NTNU – Trondheim
Norwegian University of
Science and Technology

A Price-Based Algorithm for Multi-Player Equilibrium Problems

Pål Eskild Rasmussen
Mads Falmår Wilthil

Industrial Economics and Technology Management
Submission date: June 2015
Supervisor: Rudolf Egging, IØT
Co-supervisor: Lars Magnus Hvattum, IØT

Norwegian University of Science and Technology
Department of Industrial Economics and Technology Management

Problem Description

The goal of this master thesis is to develop a solution algorithm for linearly constrained multi-objective optimization problems with equilibrium conditions. The work will focus on energy market equilibrium models.

A proof-of-concept algorithm, with corresponding model formulation, was developed as a pre-project in the fall 2014. This will serve as a starting point, as the master thesis will expand the model to handle a wider variety of energy market equilibrium models. The algorithm will be expanded to deal with added model complexity and upscaled data instances.

Algorithm performance in terms of solution speed and quality will be tested on various problems and benchmarked against other solvers (mainly PATH).

Preface

This work represents the final semester of the authors' Master of Science degrees at the Norwegian University of Science and Technology. After five years of hard work we are proud to present a thesis about solving energy market equilibrium models. This thesis is a continuation of our proof-of-concept project report written during the autumn of 2014.

We would like to thank our main supervisor Associate Professor Ruud Egging at the Department of Industrial Economics and Technology Management (IØT) for useful input, interesting discussions and helpful reviews during the course of the writing. We would also like to thank our secondary supervisor Professor Lars Magnus Hvattum for his participation in our work. Both supervisors have shown great interest and compassion in our research, making discussion sessions useful and enjoyable.

As an introduction to equilibrium modelling we received a grant from IØT to go to Infratrain 2014 in Berlin. We are very grateful for this, and believe it has helped us to understand both equilibrium modelling and the key research questions in energy market analysis.

Finally, thank you to the reader who have picked up this thesis and are ready to delve into the wonderful world of solving energy market equilibrium models. Enjoy!

Trondheim, June 11th 2015

Pål Eskild Rasmussen Mads Falmår Wilthil

Abstract

This thesis is about improving the solution time for energy market equilibrium models. Equilibrium models have seen an increased usage over the last years, and the models used for numerical analysis are growing larger and more complex. The computational solvers used to solve equilibrium problems are made for a larger class of mathematical problems and thus we believe it is possible to create more specialized algorithms that solves energy market equilibrium models faster.

A deterministic energy market equilibrium model is presented, with supplier, consumers and transmission system operators. The supplier are modelled as Cournot-players who are able to exert market power. To solve the equilibrium problem the equilibrium model is decomposed into individual and independent single-firm optimization problems with exogenous market prices. We show two different decompositions, varying in how the commodity is routed between supply and demand nodes. Both decompositions are able to keep the suppliers as Cournot-players despite their price-taking behaviour.

To solve the decomposed equilibrium model we present an iterative algorithm which searches through market prices. By acquiring the optimal decisions of the actors given one set of prices we are able to deduce how the prices should change in the next iteration. We present three different algorithm alternatives, one of which has strict requirements on the properties of the equilibrium solution to converge. The other two alternatives are more robust, but solves a computational expensive traffic equilibrium problem in each iteration.

The developed algorithm is tested and compared to the commercial solver PATH. None of the developed alternatives achieve better solution times than PATH during testing, but evidence suggests there is room for performance improvement through a more efficient handling of the traffic equilibrium problem and a more holistic search strategy. We believe that with further development a price-based algorithm can help reduce solution times of energy market equilibrium and solve models previously restricted by insurmountable solution times.

Sammendrag

I denne masteroppgaven har vi arbeidet med tiltak for å korte ned løsningstiden til likevektsmodeller. Vi har sett en økning i bruken av likevektsmodeller i analyse av internasjonale energimarked, og med økt bruk følger ofte større og mer komplekse modeller. Dette gjør at de kan ta lang tid å løse. Algoritmene som løser disse likevektsmodellene er laget for en mye større klasse av matematiske modeller, og vi tror at ved å lage en mer spesialisert algoritme vil vi være i stand til å korte ned kjøretiden for likevektsmodeller.

Vi presenterer en deterministisk likevektsmodell med tilbydere, konsumenter og transportører. Produsentene gis markedsmakt i form av en Cournot-oppførsel, mens transportmarkedet modelleres som et frikonkurransemarked. For å løse likevektsmodellen presenterer vi en dekomposisjon som lar oss løse de individuelle optimeringsproblemene uavhengig av hverandre gitt eksogene priser. Vi viser to ulike dekomposisjoner der forskjellen er hvilken aktør som håndterer ruting av energibæreren i nettverket. I begge dekomposisjonene beholder tilbyderne sin Cournot-oppførsel til tross for eksogene prisvariabler.

For å løse likevektsmodellen presenterer vi en iterativ algoritme som søker i modellens priser. Gitt et sett med priser kan vi få de optimale beslutningene fra de ulike aktørene, og disse beslutningene gir indikasjoner på hvordan prisene bør forandres til neste iterasjon. Vi presenterer tre ulike varianter av algoritmen, der en av dem har strenge krav til løsningens egenskaper i likevekt. De andre to alternativene er mer robuste, men løser et komplekst trafikklikevektsproblem i hver iterasjon.

Den utviklede algoritmen testes og sammenliknes med den kommersielle løsningsalgoritmen PATH. Ingen av alternativene viser seg raskere enn PATH på instansene brukt under testing, men vi ser et forbedringspotensiale gjennom en mer effektiv håndtering av trafikklikevektsproblemet, samt en mer helhetlig søkestrategi. Vi tror et videre arbeid med prisbaserte algoritmer kan bidra til raskere løsning av energimarkedsmodeller, og dermed løsning av modeller som tidligere har vært begrenset av kjøretid.

Contents

Chapter 1

Introduction

Section 1.1 describes the background of and motivation for our work. We examine some existing research, and relate them to the scope of our work. Section 1.2 will clearly define the objectives of our work as well as describe our limitations before Section 1.3 describes the outline of this thesis.

1.1 Background and Motivation

Decision support systems in energy markets often use one of three types of models: single firm optimization models, equilibrium models and simulation models. Ventosa et al. (2005) lists these three model classes as the main lines of development in electricity markets research and states that due to deregulation and increased competition the need for decision support systems have grown.

A single-firm optimization model is most commonly used by a single actor to investigate opportunities within their own infrastructure. The model can focus on cost-minimization in production or the economic impact of expansion opportunities.

While single-firm optimization models focuses on one firm in a more or less isolated setting, equilibrium models contain several firms and their connections. All firms are working towards their own objectives which is most often profit maximization. The connection between the firms are often of a competitive nature and some firms may be

able to exert market power to affect the end market. Equilibrium models can be used by stakeholders to model the impact of new taxes or policies, but they can also be used by single actors to investigate investment opportunities when taking decisions made by competitors and consumers into account. A drawback of using an equilibrium model compared to an optimization model is the data required: an equilibrium model may need sensitive information about all the firms in the market. There may be an extensive amount of work involved in gathering data and estimating unknown parameters. Another drawback of equilibrium models is that they can quickly grow large and complex, which in turn can make them difficult, and sometimes impossible, to solve within a reasonable amount of time.

Finally, a class of model known as simulation models can be used as decision support. A simulation model is often applied when an equilibrium model grows too complex to solve using a formal mathematical approach. Instead of using equalities and inequalities to model the actors strategic behaviour, simulation models use a set of sequential rules. This rule set can be used to estimate nearly any strategic behaviour according to Ventosa et al. (2005). The drawback of the simulation models are creating the rule set: it must be simple enough to gain a computational advantage over equilibrium models, but also represent the actors strategic behaviour in a realistic manner.Simulation models will always yield an approximation of the solution to the problem at hand compared to the equilibrium model it is based on.

Equilibrium models have become more popular in energy market analysis during the last years. An important reason for this is liberalisation. Several sectors of the western world energy markets have been liberalised over the last decades. In the early 1990s several European energy markets were monopolies. In 1996 and 1998 the European Union launched its first liberalisation directives in the electricity and natural gas markets respectively (see European Commission (2014)). With the liberalisation, several actors are able to operate in the same market. Competition and market power is becoming an increasingly interesting subject for research, and there exists a number of models which examines competition in energy markets. Some examples are Egging et al. (2010); Holz et al. (2008); Boots et al. (2003) which all analyse strategic actors in

various natural gas markets. Huppmann and Holz (2009) examines the market power exerted by OPEC in the crude oil market.

The increased use of equilibrium models have modellers expanding the models in terms of size and complexity. Modellers expand the time horizon for which the model is applied, they introduce stochasticity to a model, or they include several sectors in the same model. As a model grows larger and more complex the time needed to solve it increases, and solution times are often a restricting element in equilibrium modelling. This has created a need for reducing the solution times of equilibrium models, which will be the focus of our work.

There are several ways to decrease the solution time of an equilibrium model, some of which can be combined. Better hardware is obviously an option, but hardware is often transparent to a modeller. Hardware is easy to upgrade without considering the users, and should be combined with other measures for reducing the solution time. Another option is to approximate the model using a simulation model. Simulations have been applied to the Spanish wholesale electricity market with results close to real-world observations (see Otero-Novas et al. (2000)). Another option is to develop new solution methods which takes a shorter time to compute a solution: this is the approach we will take.

A common technique for solving equilibrium models is to reformulate them as a class of mathematical problems called mixed complementarity problems (MCPs). MCPs are a larger class of problems than equilibrium problems and for this reason the computational solvers used are quite general. The solvers will solve any MCP, and not just one arising from an equilibrium problem. When designing solution algorithms there is a often a visible trade-off between the range of models a solver can solve and how fast it will solve a given model. A solver which is able to solve many different problems must take a broad approach to ensure that an optimal solution is found to every problem regardless of parameters given by the problem description . On the other hand it is possible to develop a solution algorithm capable of solving only a single model. For specialized solution algorithms it is possible to use domain knowledge to leave out general calculations needed by a generalised solver, and solve the problem faster. There is

also an in-between solution: it is possible to create a solver capable of solving a limited range of models. The performance of such a solver should be better than the generalised algorithm, but is possibly worse than the solution algorithm constructed for only a single model.

The algorithm we develop will use domain knowledge related to the structure of energy markets and the prices they deal with. Several actors are present in an energy market (a detailed description is provided in Section 2.4.2) and they act in the market based on commodity prices, transportation prices and other prices. A single actor may have to deal with multiple prices to determine how much energy to produce, or which infrastructure investments are the most profitable.

The problem with computational inefficient methods has been assessed by others, though often by modifying the problem before it is given to the solver. Egging (2013) lists four different techniques for lowering the solution times of large MCPs: relaxation, decomposition, scenario reduction and sampling methods. Lagrangian relaxation cannot be applied to models with several competitive actors as there is no single objective function from which the Lagrangian dual can be defined.

Scenario reduction replaces a big scenario tree in a stochastic model with a smaller scenario tree. The solution to the problem using the small scenario tree should approximate the solution to the problem using the original scenario tree. Gabriel et al. (2009) applies a scenario reduction technique to a stochastic natural gas model. Two different procedures are used to estimate the size of the reduced scenario tree needed for the solution to the reduced scenario tree to approximate the solution of the full scenario tree. After the model is reduced it is solved using the same general software, but because the scenario tree is scaled down it is solvable in an acceptable time frame.

Egging (2013) presents a variant of Benders decomposition to solve a stochastic equilibrium model from the world natural gas market. Benders decomposition (see Benders (1962)) is a solution technique which removes a set of complicating variables and solves the model by blocks. An advantage for Benders decomposition over scenario reduction is that Benders decomposition finds the optimal solution to the original problem, and not an approximation.

The common factor for the decomposition-based procedures described above is that the mixed complementarity problem is decomposed. We want to avoid having to reformulate the equilibrium problem to a complementarity problem. The solver we present will solve energy market equilibrium problems based on the optimization problems of the individual actors and the connections between the actors (known as the *market clearing constraints*). All actors in an energy market is related to one or more prices. We will use this in our algorithm, and assume that if we can find equilibrium prices and force the actors to use these prices, the actors will make equilibrium decisions. This is why we call our work a **price-based algorithm for solving energy market equilibrium models**. The motivation of our work stems from international natural gas markets, but the work we present will be applicable to other sectors as well.

1.2 Objective

The main objective of this thesis is to examine if a price-based algorithm can reduce the solution times of energy market equilibrium models.

To test the solution time properties of a price-based solution algorithm we must first find a model to use as a test base. We could have used an existing energy market equilibrium model, but these are (as stated in the introduction) large, spans multiple time periods or contain stochastic parameters. Because we are building an algorithm bottom-up, the ideal situation is to have a relatively simple model. With a simple model we will be able to change a small part of the algorithm, and see the effects in the solution process. We have chosen to develop a deterministic energy market equilibrium model which is simple enough to perform solution-related analysis, but which can still be used in real-world applications.

After we have developed a model we must make sure that the actors are connected through the prices, and that the solution algorithm can use these prices to drive the solution process forwards. The model may need modifications to remove dependencies between the actors. This is the decomposition of the model.

When the decomposition is complete, we can develop the solution algorithm.

The following points summarize the objectives we must fulfill to evaluate how price-

based solution algorithms work in energy market equilibrium problems.

1. Develop a deterministic energy market equilibrium model representative of real-world applications
2. Find price-based decompositions of energy market equilibrium models
3. Develop a solution algorithm for the price-based decompositions

We will end the thesis with a test of our algorithm and a discussion on price-based solution algorithms versus existing commercial solvers. The tests will show if the specialised price-based decomposition algorithm can gain a run-time advantage over existing generalized software.

1.3 Structure of the Thesis

Chapter 2 will present the theoretical foundations for equilibrium modelling and algorithm construction. In Chapter 3 we introduce a deterministic energy market equilibrium model, which is sufficiently small to analyse and isolate effects, while still containing realistic actors and relationships. The model will include competitive actors and strategic behaviour. In Chapter 4 we present two alternative models which makes all the actors price-taking entities. This will turn out to be very important for our solution algorithm. The algorithm we have developed is presented in Chapter 5. We also discuss alternative approaches and present some details on how the algorithm is implemented. Testing and analysis is presented in Chapter 6. We do tests to analyse interesting computational aspects, show model actors behaviour and do a benchmark against commercial solvers. A conclusion is presented in Chapter 7, along with recommendations for further work.

Chapter 2

Theoretical Concepts

In this chapter we present the theoretical fundamentals for our work. We present concepts and problems relevant both for the development of our model, the decomposition and the corresponding solution algorithm. Section 2.1 introduces mathematical optimization models, equilibrium models and some game theoretic concepts.In Section 2.2 we will define algorithmic concepts which we will later use to describe our solution algorithm, and also introduce some graph theory. A traffic equilibrium problem will appear as an important sub-problem in our model and Section 2.3 introduces important definitions in this framework. To conclude the chapter, a summary on energy market models is presented in Section 2.4.

2.1 Mathematical Modelling

Stating, solving and applying mathematical models are each a science of its own, and a huge number of papers have been presented in the field of operations research. This Section aims to familiarize the reader with key concepts in optimization and equilibrium modelling.

In Section 2.1.1 we present the fundamentals of optimization and mathematical programming. The Karush-Kuhn-Tucker conditions are introduced in Section 2.1.2. Next we state some conditions for unique optimal solutions to exist in a problem in

Section 2.1.3. Finally, we look into equilibrium models in Section 2.1.4 where we also define some classes of equilibrium in a game-theoretic framework.

2.1.1 Optimization Models

There are several ways to describe an optimization problem. In its most general form the goal is to find a combination of input which yields an optimal output. An example problem is stated in expressions (2.1)–(2.2).

$$\max_x \quad f(x) \tag{2.1}$$

$$\text{s.t.} \quad x \in S \tag{2.2}$$

In expressions (2.1)and (2.2) we let x denote one or more **decision variables** and let S be the **feasible region** of x - it consists of every possible value combination that x can take. f is called the **objective function** of the optimization problem and maps the vector x onto a scalar value $f(x)$.

The formulation (2.1)–(2.2) is not very precise, and does not open for much analysis or discussion. Throughout the rest of this section, we limit ourselves to look at a more specialized feasible region represented by real-valued functions. If x has l elements and S can be represented by m functions $g : \mathbb{R}^l \to \mathbb{R}$ and n functions $h : \mathbb{R}^l \to \mathbb{R}$, we can replace formulation (2.1)–(2.2) with formulation (2.3)–(2.5).

$$\max_x \quad f(x) \tag{2.3}$$

$$\text{s.t.} \quad g_i(x) \leq 0 \quad i = 1, \ldots, m \tag{2.4}$$

$$h_j(x) = 0 \quad j = 1, \ldots, n \tag{2.5}$$

Even though we lose some generality in the replacement of constraint (2.2), formulation (2.3)–(2.5) will prove to be useful in the practical applications we consider.

Depending on the properties of f, g and h there may be several ways to solve these

problems. A specialized and well-studied instance of optimization problems are linear programming problems (LP-problems), described by Dantzig (1963). If f, g and h are affine functions and the elements of x can take continuous values, then the problem (2.3)-(2.5) is a linear problem. The simplex algorithm (described by Dantzig et al. (1955)) has proven successful in solving LP-problems. In fact, the simplex algorithm guarantees finding an optimal solution x^* if such a solution exists.

Another class of programming problems relevant for this thesis are quadratic programming problems (QP-problems). Hillier and Lieberman (2010) defines quadratic programming problems as having a quadratic objective function f, but still restricted by affine constraints g and h, as well as continuous variables x. Clearly, QP-problems share some properties with the LP-problems, but the quadratic part of the objective function introduce some complicating factors in the problem solution.

The approach of using the simplex algorithm described by Dantzig (1963) will only solve linear problems. If a problem is quadratic, or of another classification, a new approach is needed. Section 2.1.2 will show how to reformulate an optimization problem into a set of equations using the Karush-Kuhn-Tucker conditions. After this is done the system may be solved using, for instance, iterative methods described by Ortega and Rheinboldt (2000). Before introducing the Karush-Kuhn-Tucker conditions it is useful to introduce the principle of duality in linear programming.

Duality in linear programming

Duality is an important concept in optimization. Given a linear programming problem P on standard form with n variables and m constraints it is possible to state an equivalent optimization problem D with m variables and n constraints. In this context P is called the primal problem and D is the dual problem. Lundgren et al. (2010) presents an algorithm to formulate the dual problem given a primal linear programming problem. The essence is that each variable in the primal becomes a constraint in the dual, and each constraint in the primal becomes a variable in the dual.

The dual variable corresponding to a primal constraint is also known as the **shadow price** of the constraint. For a given solution, the dual variable express the change in

the objective function value by a corresponding change in the right-hand side of the constraint. Only binding constraints may have a positive dual price: a unit increase in the right hand side is worthless unless we wish to increase the left hand side as well. This property is very important in stating the Karush-Kuhn-Tucker conditions.

2.1.2 The Karush-Kuhn-Tucker Conditions

The Karush-Kuhn-Tucker was introduced by Karush (1939) and Kuhn and Tucker (1951) and are first-order conditions which lets some instances of mathematical programming problems be transformed from their "optimization-form" into sets of inequalities and equalities.

Recall the optimization problem (2.3)–(2.5). If the functions f, g and h are differentiable, it is possible to state the Karush-Kuhn-Tucker conditions for the problem. If x^* is an optimal solution to the problem (2.3)–(2.5), then there must exist m dual multipliers μ_i and n dual multipliers λ_j which satisfy expressions (2.6) trough (2.10).

$$\nabla f(x^*) = \sum_{i=1}^{m} \mu_i \nabla g_i(x^*) + \sum_{j=1}^{n} \lambda_j \nabla h_j(x^*) \tag{2.6}$$

$$g_i(x^*) \leq 0 \qquad i = 1, ..., m \tag{2.7}$$

$$h_j(x^*) = 0 \qquad j = 1, ..., n \tag{2.8}$$

$$\mu_i \geq 0 \qquad i = 1, ..., m \tag{2.9}$$

$$\mu_i g_i(x^*) = 0 \qquad i = 1, ..., m \tag{2.10}$$

Conditions (2.7), (2.9) and (2.10) can be replaced by a single **complementarity condition**, and the resulting conditions are shown in expressions (2.11) trough (2.13).

$$\nabla f(x^*) = \sum_{i=1}^{m} \mu_i \nabla g_i(x^*) + \sum_{j=1}^{n} \lambda_j \nabla h_j(x^*) \tag{2.11}$$

$$h_j(x^*) = 0 \quad j = 1, ..., n \tag{2.12}$$

$$0 \geq g_i(x) \perp \mu_i \geq 0 \quad i = 1, ..., m \tag{2.13}$$

In particular, notice that condition (2.13) formalizes the connection between a constraint and its dual variable. If the dual μ_i is positive the constraint $g_i(x)$ must equal zero, in which case it is binding. If the constraint is negative (not binding) then the dual must be 0.

When stating the Karush-Kuhn-Tucker conditions for a problem (2.3)–(2.5) the problem changes from finding a set of variables constituting an optimal decision to finding a set of variables constituting a feasible decision **if** the KKT-conditions can be shown to be both sufficient and necessary conditions for optimality.

The Karush-Kuhn-Tucker conditions as necessary conditions for optimality

The constraints (2.4) and (2.5) must meet regularity conditions for the KKT conditions to be meaningful. Gabriel et al. (2012) state several such conditions, but the *Linear Independent Constraint Qualifications* are of particular interest. If the gradients of the binding inequality constraints (2.4) and the gradients of the equality constraints (2.5) are linearly independent at a given solution x, this solution meets the regularity conditions.

Chiang (1993) shows that if $g_i(x)$ and $h_j(x)$ are affine functions, the regularity conditions holds. Thus, an optimal solution to the problem (2.3)–(2.5) is also a solution to the Karush-Kuhn-Tucker conditions, given affine constraints.

The Karush-Kuhn-Tucker conditions as sufficient conditions for optimality

If the KKT-conditions is sufficient for optimality it means that a solution to the KKT-conditions will also be an optimal solution to the original problem.

If $f(x)$ is continuously differentiable and concave and the constraints (2.4) and (2.5)

constitute a convex set, then a solution to the KKT conditions is also an optimal solution to the problem (2.3) – (2.5) (again, see Gabriel et al. (2012)). If the problem is a minimization problem, the requirement on $f(x)$ is that it is continuously differentiable and convex.

If the constraints (2.4) and (2.5) are affine, then the union of their feasible region will constitute a convex set (see Bazaraa et al. (1993)).

The model presented in Chapter 3 will show how the KKT-conditions can be used to solve an equilibrium model.

2.1.3 Uniqueness of Solution in Optimization Problems

After stating an optimization problem it is of interest if a solution to the problem exists. If a feasible solution exists, then an optimal solution must exist to the problem. That makes it interesting to determine if a single unique optimal solution exist, or if multiple optimal solutions can be found. To help determine uniqueness of solutions we use Hass et al. (2008) who defines the gradient ∇f and Hessian $\mathbf{H}(f)$ of a function $f(x_1, x_2, \ldots, x_n)$ as shown in Equation (2.14) and (2.15) respectively.

$$\nabla f = \begin{bmatrix} \frac{\delta f}{\delta x_1} \\ \frac{\delta f}{\delta x_2} \\ \vdots \\ \frac{\delta f}{\delta x_n} \end{bmatrix} \tag{2.14}$$

If a point x' solves the equation $\nabla f(x') = 0$, then x' is said to be a stationary point of the function f.

$$\mathbf{H}(f) = \begin{bmatrix} \frac{\delta^2 f}{\delta x_1^2} & \frac{\delta^2 f}{\delta x_1 \delta x_2} & \cdots & \frac{\delta^2 f}{\delta x_1 \delta x_n} \\ \frac{\delta^2 f}{\delta x_2 \delta x_1} & \frac{\delta^2 f}{\delta x_2^2} & \cdots & \frac{\delta^2 f}{\delta x_2 \delta x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\delta^2 f}{\delta x_n \delta x_1} & \frac{\delta^2 f}{\delta x_n \delta x_2} & \cdots & \frac{\delta^2 f}{\delta x_n^2} \end{bmatrix} \tag{2.15}$$

A negative definite matrix M is such that for every non-zero vector z Inequality (2.16)

must hold.

$$z^T M z < 0 \tag{2.16}$$

If the Hessian **H**(f) is negative definite at a stationary point x', then this point is a local maximum for the function f.

For an unconstrained optimization problem (2.3) to have a solution there must exist at least one stationary point x^* for which the hessian is negative definite. If only a single stationary point x^* exists and the hessian is negative definite, then this point is the unique optimal solution to the problem.

For a constrained optimization problem (2.3) trough (2.5) to have a solution there must either exist at least one stationary point x^* for which constraints (2.4) trough (2.5) are satisfied and the Hessian is negative definite, or an optimal solution exists on the border of the feasible region.

Uniqueness of solution in minimization problem follows the same argumentation as for maximization problems, though the Hessian of objective function must be positive definite, that is it must satisfy inequality (2.17) for every non-zero vector z.

$$z^T M z > 0 \tag{2.17}$$

2.1.4 Multi-Objective Optimization and Equilibrium Modeling

So far in this chapter we have discussed optimization problems with a single objective function. If several decision makers are present in an environment it is possible to assign one optimization problem to each decision maker. The decisions of one actor may affect the others, so their problems are interrelated. Gabriel et al. (2012) defines a problem with several objective functions as an **equilibrium problem**. Let K be the set of actors in the model, X the set of variables and $x_k \subseteq X$. the variables under control of actor $k \in K$. $f_k(X)$ is the objective function of actor k, while I^k is the set of constraints applying only to actor k and J is the set of shared constraints. It is possible to define $|K|$ optimization problems in expressions (2.18) trough (2.20).

$$\max_{x_k} \quad f_k(X) \tag{2.18}$$

$$\text{s.t.} \quad g_i(x_k) = 0 \quad \forall\ i \in I^k \tag{2.19}$$

$$h_j(X) = 0 \quad \forall\ j \in J \tag{2.20}$$

If it possible to formulate the objective function as $f_k(X) = f_k(x_k) + f_k(X \setminus x_k)$ and J is an empty set, the optimization problem can be decomposed into independent optimization problems. If the optimization problems are independent then each actor k can disregard the actions of other actors and the problems (2.18) trough (2.20) can be solved trough **equilibrium in dominant actions**: each actor solves his problem by maximizing $f_k(x_k)$ and disregards the actions of other actors.

In the general case the problems are not independent, and there are different ways to define an equilibrium. For the remainder of this section we will look at Cournot Games, Market Power and Nash Equilibrium.

Cournot Games and Market Power

A Cournot framework as described by Pindyck and Rubinfeld (2005) has K actors competing on selling a single commodity in a shared market. An actor makes decisions on his own production level given the production levels of competing actors.

Consider an example with two firms selling the same commodity in a single market. Both firms decide on a production level q_i for $i = \{1,2\}$. The price in the market is affected by the joint production of both producers, $Q = q_1 + q_2$. Let this price be noted $p(Q)$. If the total costs of producer i is $c_i(q_i)$ and both producers acts as a profit-maximizing unit, then the market contains 2 maximization problems, of the form given in (2.21)–(2.22).

$$\max_{q_i} \quad \pi_i = p(Q)q_i - c_i(q_i) \tag{2.21}$$

$$\text{s.t.} \quad q_i \geq 0 \tag{2.22}$$

Even though the constraints in the problem are independent for each actor, the commodity price in the objective function ties the optimization problems together. Equilibrium in dominant actions is impossible in this problem, as the production decision made by one actor affects the price seen by other actors. Acting under the assumptions of Cournot however, each player treats the others production decisions as fixed and solves the above optimization problem using the first-order condition expressed in Equation (2.23).

$$\frac{\delta \pi_i}{\delta q_i} = \frac{\delta p(Q)}{\delta q_i} q_i + p(Q) - c_i'(q_i) = 0 \tag{2.23}$$

When equations (2.23) are solved simultaneously for all firms then the solution represents a Cournot equilibrium.

In a perfectly competitive market a commodity will be priced at the producers marginal cost. If a producer is able to raise the price of a commodity trough his actions (for instance by reducing his production level) then he is able to exert **market power** (c.f. Pindyck and Rubinfeld (2005)). Cournot players have market power because they see the inverse demand curve $p(Q)$ as a function of the total supplied goods, and not just a constant price p.

Nash Equilibrium

A Nash Equilibrium is broader than a Cournot equilibrium (see Nash (1951)). Whereas a Cournot Equilibrium assumes that the production decision of others are given, Nash says that every actor in the market is doing the best he can given the decisions of the other actors. These decisions can be more than just a production decision. When a market is in a Nash Equilibrium, no actor can improve his position by changing his decisions.

In Section 2.3 we will introduce a special case of equilibrium models, namely traffic equilibrium models. We will introduce the Wardrop equilibrium, which specializes a Nash equilibrium to describe a traffic equilibrium. Later, in Chapter 3 we will introduce a deterministic energy market equilibrium model where the actors are Cournot-players.

2.2 Algorithms and Data Structures

In this section we introduce concepts of algorithms and define graphs, a very important data structure in network applications. Section 2.2.1 gives a general introduction to algorithms. Section 2.2.2 explains local search. Finally, some graph theory and relevant graph algorithms are presented in Section 2.2.3.

2.2.1 Algorithm Introduction

Cormen et al. (2009) define an algorithm as "a well-defined procedure which takes an input and transforms it into an output". Many people associate algorithms with computer programs, but a procedure written in plain English is also an algorithm. If an algorithm always yield the correct output given an input and a problem description it is called a correct algorithm. The simplex algorithm is a correct algorithm for finding the optimal variable values in linear programming problems. Given the objective function and restrictions, the simplex algorithm will always yield an optimal solution, or notify the user that an optimal solution does not exist.

Several classes of algorithms exists. Heuristics are a subset of algorithms which gives no guarantee to be correct, but fairly often finds a result which is "quite close" to the optimal values. A heuristic often have a solution time advantage compared to a correct algorithm on a given problem. Another class of algorithms are local search algorithms, which are explained in detail in the next section.

2.2.2 Local search

The **neighbourhood** of a solution x is a set of solutions $N(x)$ which can be obtained by making a small change to x. The small change can also be called a **local** change. A **local search** algorithm is an algorithm which finds an approximation of an optimal problem solution by starting with an initial feasible solution and applying local changes to move around in the search space. If the algorithm always moves to a neighbour which is better than the current solution, then the procedure is known as a hill-climbing algorithm.

2.2.3 Graph theory

Graphs are important mathematical structures for representing networks of any kind. They are used in a wide range of applications, from computer science to the social studies. Rosen (2011) describes a graph as following:

A graph is an ordered pair $G = \{V, E\}$ where V is a set of **nodes** and E is a set of **edges**. An edge connects two nodes, which means it can be written as $e = (u, v) \in E$ where $u, v \in V$. Graphs can be either directed or undirected. In a directed graph the tuple (u, v) is ordered, and the edge is said to start at u and end at v. For an undirected graph, the order of u and v has no implications on the graph properties, and the edge can be said to start at u and end at v, or vice verse.

A **path** p of length n from node u to node v is an ordered set of edges $p = \{e_1, e_2, \dots e_n\}$ where the end node of one edge is the starting node of the successive edge. A path can also be described as an ordered sequence of nodes $p = \{x_0 = u, x_1, \dots, x_{n-1}, x_n = v\}$. If a path starts and ends in the same node ($u = v$) that path is said to be a **cycle**. A path will contain a cycle if an intermediate node appears more than once. A path which does not contain a cycle is called a simple path. The number of simple paths in a graph can grow very large. If a graph is complete (every node has an edge to all other nodes) this number is $n!$ (the faculty of n).

In some graphs it is meaningful to associate a weight w_e with each edge e. The weight of a path is defined as the sum of its underlying edges: $w_p = \sum_{e \in p} w_e$. The nodes and the edges of a graph constitutes its topology. Two graphs G and G' have the same topology if their unweighted counterparts are equal.

Graphs can be used to represent networks, where a single node describes a physical unit (for example a country) and an edge describes a direct connection between the units (for example a natural gas pipeline). Weights can be used to represent either a cost, a distance, or other metrics. An example graph is shown in Figure 2.1.

The Shortest Path Problem

A well-studied problem in graph theory is to find the shortest path between two nodes u and v. For the problem to have a solution there must be no negative cycles in the

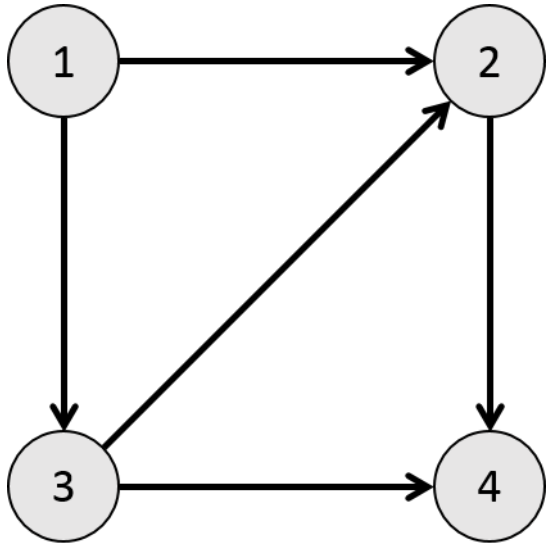

Figure 2.1: An example graph with four nodes and five edges. Note that it is directed, unweighted and contains no cycles.

graph. A negative cycle is defined as a path from node u to node u for which the weight of the path is negative. Several algorithm exist for the shortest path problem, and they have different requirements to the edge weights or network topology.

Floyd (1962) presents an algorithm for finding the shortest path from all nodes to every other node. This algorithm has no requirement for the edge weight, and therefore works on every graph which does not have a negative cycle. Because Floyd essentially used the results from Warshall (1962) in a graph-framework, the algorithm has later become known as the Floyd-Warshall algorithm. We do not restate the algorithm here, but emphasise that the shortest path problem is computationally easy compared to many other graph problems.

Depth-First Search

A depth-first search is an algorithm which can be used to find a set of nodes P_u in a graph G with starting node u for which a path exists from u to $v_i \in P_u$. Depth-first search can be implemented by a recursive procedure, which means it calls itself repeatedly. The inner workings of a depth-first search is described in Algorithm 1.

Data: A graph G and a starting node u
Result: All nodes $u_p = [v_1, v_2, \ldots, v_n]$ for which a path exists from u to v_i is marked as discovered
Label u as discovered;
foreach *edge e starting in u* **do**
  if *the end-node of e is not marked as discovered* **then**
    Call depth-first search with u = end-node of e and $G = G$
  end
end

Algorithm 1: Depth-first search.

We will see that both the Floyd-Warshall algorithm and the depth-first search will be very useful when we present our solution algorithm in Chapter 5.

We can use graphs to illustrate the difference between a general local search and a specialized hill climbing algorithm. Think of the solutions of a problem as nodes in a graph. A solution x has an edge to every solution in its neighbourhood $N(x)$. If a local search traverses the entire graph searching for the best decision then it will find a good solution, but will spend a long time searching. A specialized hill-climbing algorithm will just traverse one path, and stop when no improving neighbours are found. However, it will take a shorter time (because the search visits less nodes).

2.3 Traffic Equilibrium Models

Traffic Equilibrium Models are used to support decision-making and analysis in urban transportation networks. Graphs, introduced in Section 2.2.3 are often used to represent transportation networks. Nodes represent hubs of origin or destination, as well as intersections. Edges represent links between these hubs. The cost associated with a link can be widely defined. Many models use the time used to traverse the link, the user-cost of using the link, or the social cost of using the link.

In the following, we will consider a subset of Traffic Equilibrium Models defined by Dafermos and Sparrow (1969) as Traffic Assignment Problems (TAP). Given a demand of traffic between nodes in a network as well as cost functions for each edge the goal is to find flows of traffic which satisfy some principles of user behaviour. The common

factor in TAPs is that the cost associated with the links are experienced by the end user, and not the system as a whole.

A commonly used principle of user behaviour is described by Wardrop (1952), which states that the users of the network behave in the best way possible to themselves, while the system as a whole works optimally. The principles are formalized trough a User Equilibrium condition and a System Optimality condition:

- **User Equilibrium**: The journey times on all the routes actually used are equal, and less than those which would be experienced by a single vehicle on any unused route.

- **System Optimality**: The average journey time is a minimum.

In particular, note that the user equilibrium follows the definition of a Nash equilibrium presented in Section 2.1.4. If every other drivers route choice remains the same, a driver going from r to s has no incentive to change his route because all routes from r to s has the same travel time. He is doing the best he can **given** the decisions of the other drivers.

A traffic assignment problem can be either static or dynamic. A static TAP assumes a traveller is present at every link on his route for the entire problem. In dynamic TAPs a traveller is at a single link at a time, and moves from link to link as part of his trip.

For the rest of this section we will look at static TAPs. There are two different ways to represent the final traffic in a static TAP: route-based formulations and link-based formulations. A route-based TAP assigns traffic to a path between source and destination, and we show a complete formulation of a route-based TAP model in Section 2.3.1. A link-based TAP assigns traffic to the individual edges (or arcs) in the network, and is described in Section 2.3.2.

2.3.1 Route-Based Traffic Assignment Problems

To define a TAP we use the notation below. Let

N	be the set of nodes to model
A	be the set of links (archs) to model
$I \subseteq N^2$	be the set of origin-destination pairs to model
x_a	be the number of users on link $a \in A$
f_k^{sd}	be the number of users on route k belonging to $(s,d) \in I$
q_{rs}	be the demand of transport from r to s
$t_a(x_a)$	be the cost function of link a
δ_{ak}^{rs}	be a link-path indicator, being 1 if link a is on path k from (r,s) or 0 otherwise

It also possible to state Wardrops definition of a user equilibrium mathematically: this is shown in Equations (2.24) and (2.25).

$$f_k^{rs} > 0 \quad \Longrightarrow \quad \sum_{a \in A} \delta_{ak}^{rs} t_a(x_a) = \pi_{rs} \qquad (r,s) \in I \quad k \in R_{rs} \tag{2.24}$$

$$f_k^{rs} = 0 \quad \Longrightarrow \quad \sum_{a \in A} \delta_{ak}^{rs} t_a(x_a) \geq \pi_{rs} \qquad (r,s) \in I \quad k \in R_{rs} \tag{2.25}$$

In Equations (2.24) and (2.25) π_{rs} is the cost associated with traveling from node r to node s. A solution satisfying the requirements (2.24)–(2.25) will also satisfy the Wardrop principles of user equilibrium.

The Wardrop user equilibrium will be very important when we present the equilibrium model decomposition in Chapter 4.

2.3.2 Link-Based Traffic Assignment Problems

Link-based traffic assignment problems works dynamically from an initial solution x and iteratively changes the flow towards a target flow.

There is limited research done on link-based formulation, and in 2010 there was not

a single published real-world traffic equilibrium model which made use of a link-based formulation (according to He et al. (2010)). Watling and Cantarella (2013) suggests that link-based TAPs should be used as an extension to traditional route-based traffic equilibrium problems rather than a replacement. Link-based formulations are suited for dynamic models, and analysis which is concerned with how a traffic pattern **converges** to equilibrium as well as the equilibrium pattern itself.

2.4 Energy Market Models

This section explains the motivation for energy market models, and explain some building blocks which can be used in constructing a model. Section 2.4.1 explains some of the background of these models, and why they are useful. The actors of an energy market model are explained in Section 2.4.2.

2.4.1 Key Research Questions in Energy Markets

There are several reasons why energy market models have become increasingly popular over the last years. We introduced liberalization and competition in Chapter 1. In this section we will go a bit more in-depth about some key research questions asked in energy market analysis. Understanding the models being created today will help us present a more realistic model in Chapter 3.

A growing population and growing economies (particularly in Asia) raises the world demand for energy (see Crompton and Wu (2005)). While the use of all fuels will increase, the use of natural gas, coal and renewable sources is projected to grow the most (c.f. International Energy Agency, restated by Egging (2010)).

Natural gas has become an interesting source of energy in several markets. The shale gas boom has driven down prices in the United States, and this means Europe has become a viable market for exporting natural gas. Energy market models can highlight changes in price resulting from LNG expansions or shale gas investments, and help analyse how various energy markets will look in the distant and not-so-distant future. See Brown et al. (2010) for a thorough analysis of natural gas usage in the United States,

as well as projected usage in the coming years. Investing in LNG-infrastructure can be very profitable to facilitate the export, but location and timing may be crucial.

Russia is the sole largest supplier of natural gas to the European market. Many important pipelines run trough Ukraine, and the recent crisis in eastern Europe has raised questions on supply security for western European consumers. With Russia, Norway and Algeria providing 80 % of the gas imports to the European Union in 2001 the question of security of supply has been raised. For more details on market power in the European natural gas market, see Egging and Gabriel (2006). With current infrastructure a disruption in the Russian supply for the winter of 2014 would have forced the Baltic states to lower their consumption to 40-60 % of a regular winter (see ENTSOG (2014)). Western Europe is not as much affected by the disruption of Russian supply, as an increase in LNG import will be able to cope with the shortage. Eastern Europe does not have this option, as the pipeline capacity between the regions are limiting transmission. Energy market models can be used in numerous ways to analyse this situation. What happens if pipeline capacity is expanded? Where should LNG import terminals be build?

Current energy markets have seen an increased focus on shifting from fossil fuel sources to renewable energy sources. The efforts were intensified with the Kyoto Protocol of 1997 (see de Chazournes (1998)). In 2006, 81 % of global energy demand was covered by fossil fuels. A model of the market can help determine the incentives needed for renewable fuels to become interesting investments in the years to come.

2.4.2 The Actors of an Energy Market and their Relationships

Several actors interact in energy markets. We present a selection of actors here which we consider relevant for an energy market model. This list is not complete, and more information can be found in Egging et al. (2010).

We do not state that these actors are required in an energy market model, nor that these are the only actors in energy market models. Simply, we want to help bridge the connection between the models introduced in Section 2.1 and the model we present in Chapter 3.

A common term in energy market models is “fuel”. It denotes an energy carrier which can be transported between physical locations and quantified. Gas, electricity and bio-fuel are all examples of fuels in this setting. Models can deal with either a single fuel, or multiple fuels.

The actors of an energy market model interact with the fuel in some way. The following is a brief description of different energy market actors, with real-world examples of the different types.

- Suppliers provide fuel to the market. A supplier may be the source of a single fuel, such as a wind farm supplying electricity, or multiple fuels, which is the case if an offshore platform provides both oil and natural gas to the market.
- Consumers demand one or more fuels from the market, such as households consuming electricity and/or gas.
- Transformers converts fuels into other. A gas power plant is a transformer which turns natural gas into electricity and/or heat.
- Transmission System Operators provide means of transporting fuel between physical locations in the market. While electricity is transported by cables, gas can be pumped trough pipelines or even transported as liquefied natural gas by ship.
- Storage System Operators provide means of storing fuel between time periods. Hydro power plants can store water in mountain basins, while natural gas can be stored underground.

One particular way of organizing the actors is to group them in nodes. A node represent a market with a given demand, possibly diversified between several consumer sectors. Several producers may operate in a single node, and a single producer may operate in different nodes. Transmission of fuel between nodes are available at a cost set by the transmission system operator. Also, transmission may be limited by link capacity.

There can be several transmission system operators operating several links in the network. By assuming the market for fuel transmission functions as a freely competitive market, the operators can be aggregated to a single actor. The cost of using a link is determined in two different cases.

In case (1) there is spare capacity on the link between two nodes, and the price will be set at the lowest marginal cost of the operators on the link. Any higher price will break the assumptions of a freely competitive market.

In case (2) the capacity on the link will be fully used. Then, it is possible that the suppliers wants to send more across the link if they had been able to. An implicit auction is performed on order to determine if the TSO can add a tariff to the link. In the optimal state (for the TSO) the price is set such that the price of the link is equal to the highest evaluation among the players. In this case, the players want to send just the capacity amount across the link, but no more.

Chapter 3

A single fuel, single-period energy market model

In order to develop a solution algorithm for energy market equilibrium models, we need a reference model to specify what the algorithm should solve. To perform tests on the algorithm and assert it both in terms of accuracy and speed we have developed a simple single fuel, single-period energy market model. We have developed a limited-scale model with which testing should be clear and concise, while still keeping the most important properties of the models presented in existing literature (see Section 2.4).

Section 3.1 gives an overview of the assumptions for our model, and the notation we use throughout the rest of this thesis. The problems of the individual actors are presented in Section 3.2. Finally, Section 3.3 joins the individual problems to an equilibrium model for the energy market in question.

3.1 Model Overview

The model groups actors in nodes as described in Section 2.4.2. It is a single-period, single-fuel model including suppliers, consumers and transmission system operators (TSOs). Limiting the number of actors the model limits the number of variables and

constraints. This makes it easier to analyze how the actors parameters affect the model solution time and behaviour, and isolate cause-effect relations across the model.

The reason for choosing a single fuel is to avoid the complications of fuel substitution. We do not see this choice as controversial as several models are made to analyze a single sector of the energy market. See for instance Holz et al. (2008) and Huppmann and Holz (2009). By modelling a single sector the need to include transformers in the model is removed. It is true that transformers are still buying or selling fuel, but they can be modelled as either a supplier or a consumer.

The need for storage system operators (SSOs) are less apparent with only one time period. It can be argued that the SSOs are still present in the markets, and buys and supplies fuel depending on the price. We argue that this can be approximated using a producer and a market in the same node.

The model is formulated in an equilibrium problem framework, where the goal of each actor is expressed as an optimization problem and connected trough market clearing constraints. This allows the actors to make independent decisions and act strategically. There exists alternatives to formulating and solving an equilibrium model. One is to optimize social welfare: one example of a social welfare optimization model is Leuthold et al. (2012) which maximizes welfare for the European electricity market. However, social welfare optimization is dependent on perfectly competitive markets. We want to include strategic players in our model formulation.

Some models, such as The World Gas Model described by Egging et al. (2010) and GASTALE described by Boots et al. (2003) explicitly use traders to buy gas at the border of one node, transport the gas to a new node border and sell it. The traders are modelled to be either perfectly competitive or Cournot players. In the following model, the suppliers take the role as traders. We elaborate more on this when stating the supplier problems in Section 3.2.1.

3.1.1 Model Assumptions

The following section describe the assumptions we have made in our model, and discuss how accurate these are.

1. The suppliers act as profit-maximizing entities.
2. The suppliers see the inverse demand curve of every market.
3. The suppliers have no binding contracts.
4. The suppliers production costs can be represented by quadratic functions.
5. Consumers in a single node may be aggregated to express a single demand curve.
6. The relationship between price and demanded quantity is linear in all nodes.
7. The demand in one market is independent from the demand in other markets.
8. The transportation sector is a freely competitive market.
9. The transmission system operators can be aggregated for each direct link between two nodes.
10. The transportation costs can be represented by an independent quadratic function for each link.
11. Transportation of fuel is assumed to be lossless.
12. There is no capacity restriction on nodal throughput, only on links.

It is Assumption 2 which enables the actors to act strategically. By seeing the inverse demand function the suppliers can predict how the market price will change as the supplied commodity changes. Without this assumption, the suppliers would act as perfectly competitive entities.

Assumption 3 can be relaxed by introducing a lower bound on the variables in the model.

Assumption 8 has a very important consequence on the link price. If the flow on a link is strictly between lower and upper bound of the link, the marginal revenue for the TSO must equal the marginal cost of operating the link. Only if the link is at capacity can the marginal revenue be higher than the marginal cost. If the flow is at the lower bound then the marginal revenue can be lower than the marginal cost. We will show this after the model is stated, in Section 3.3.

Notice that Assumption 9 is a consequence of Assumption 8. Even if several transmission lines exists between two locations (such as several power lines) we assume it is possible to aggregate them to a single connection. In turn, this will greatly simplify calculations and computational considerations. For the rest of this thesis we will discuss "the link from node n to node m" and disregard the underlying physical infrastructure.

Assumption 11 is not necessarily true in a real-life system. Fuel can leak during transmission. In some networks the fuel itself is used to operate links, which may be the case with gas transmission by pipeline where the transported gas may be used to drive compressors. Still, existing literature make this assumption even in cases where it is a fact that the opposite holds (c.f. Boots et al. (2003)) because lossless transmission simplifies the network flow balance equations.

3.1.2 Model notation

The following section introduce the notation used in the model. Capital letters are used for sets and parameters, while lower case indicate variables and indices. Note that some variables are stated as greek letters. These represent duals from the independent actors problem, a connection which will become clear in Section 3.3.

Sets and Indices

Set	Indices	
N	n, m	Geographical nodes, each representing a market (e.g. countries)
P	i, k	Suppliers
$o(n)$		All nodes m with a direct transmission link from node n to node m (including n)
$i(n)$		All nodes m with a direct transmission link to node n from node m (including n)
$p(i)$		All nodes n where producer i has production facilities

Parameters

Parameter	Description
A_n	The slope of the inverse demand function in node n
B_n	The intercept of the inverse demand function in node n
CAP_{ni}^{prod}	The production capacity of supplier i in node n
CAP_{nm}^{trans}	The capacity of fuel transmission on link (n,m)
$C_{1ni}^{prod}, C_{2ni}^{prod}$	Production cost parameters of supplier i in node n, the linear and quadratic coefficient respectively
$C_{1nm}^{trans}, C_{2nm}^{trans}$	Transportation cost parameters for ink (n,m), the linear and quadratic coefficient respectively

Variables

Variable	**Description**
s_{ni}	Amount sold by producer i at node n
q_{ni}	Quantity produced by producer i at node n
f_{nmi}	Flow for producer i on link (n,m)
g_{nm}	Total flow on link (n,m)
π_n	Commodity price at node n
α_{nm}	Transportation price on link (n,m)
β_{ni}^{sup}	Dual prices for the constraint on supplier capacity
β_{nm}^{tso}	Dual prices for the constraint on TSO capacity
σ_{ni}	Dual prices for the flow conservation constraints

In some problems we will describe a variable as exogenous. An exogenous variable is treated as a parameter in a given problem because it is controlled by another actor (such as the production decision s_{nk} in objective function (3.1)) or because it is controlled by a market clearing constraint (such as the transmission price α_{nm}).

3.2 The Problems of the Individual Actors and their Connections

The supplier problems are stated in Section 3.2.1. According to our assumptions, the transportation sector is aggregated and the transportation problem is presented in Section 3.2.2. The consumers have no problem of their own, but are connected to the other problems trough market clearing constraints. The market clearing constraints are presented in Section 3.2.3. Finally, the complete equilibrium model is stated in Section 3.3.

3.2.1 The Supplier Problems

Each supplier i decides how much to produce at the nodes it has production facilities (q_{ni}), as well as sales (s_{ni}) in the markets and transportation (f_{nmi}) between the different nodes. There may be several suppliers in each node n, and a single supplier can have production in several nodes. They all maximize their own profit.

$$\max_{s_{ni},q_{ni},f_{nm}} \quad \sum_{n\in N}[(B_n - A_n \sum_{k\in P} s_{nk})s_{ni} - (C^{prod}_{1ni} + C^{prod}_{2ni} q_{ni})q_{ni}] - \sum_{n\in N}\sum_{m\in N} \alpha_{nm} f_{nmi} \tag{3.1}$$

$$\text{s.t.} \quad 0 \le q_{ni} \le CAP^{prod}_{ni} \qquad (\beta^{sup}_{ni}) \quad \forall n \in p(i) \tag{3.2}$$

$$s_{ni} - q_{ni} + \sum_{m\in o(n)} f_{nmi} - \sum_{m\in i(n)} f_{mni} = 0 \qquad (\sigma_{ni}) \quad \forall n \in N \tag{3.3}$$

$$s_{ni} \ge 0 \qquad \forall n \in N, i \in P \tag{3.4}$$

$$f_{nmi} \ge 0 \qquad \forall n, m \in N, i \in P \tag{3.5}$$

Objective function (3.1) is the revenue from selling fuel across all the nodes less the production costs for producing the fuel and the transportation costs across all links. Note that α_{nm} is not determined by the supplier and can be viewed as a parameter

at this point. Constraint (3.2) ensures that the production is non-negative and within capacity across all nodes. The constraint (3.3) is the node-balance equation for each node, ensuring that all commodities in a node is either sold in that node or routed to be sold in other nodes. Finally, constraints (3.4) and (3.5) are non-negativity constraints on the amount of fuel sold and link transmission.

The supplier is given full Cournot market power by seeing the slope of the inverse demand curve A_n in all markets. If the supplier would be acting as a perfect competitive player, the price π_n of each market replaces the expression $B_n - A_n \sum_{k \in P} s_{nk}$ in the objective function (3.1).

We have chosen to keep all the sales variables s_{ni} non-negative. This hinders a supplier to buy fuel in one node, pay for transmission to a different node and sell it there. While the ability to buy fuel seems like a good idea from the perspective of having the suppliers act as traders, it will never happen with our current assumptions. All suppliers (traders) have the same transmission costs and sales revenues which means that for one supplier to increase his profits buying fuel, another supplier lowers his profit. This goes against Assumption 1, so we can conclude that restricting the sales variables s_{ni} to be non-negative does not affect the trading behavior of the suppliers.

As discussed in the model assumptions (Section 3.1.1) we can enforce contracts for supplier i in node n by adding a lower bound on s_{ni}, effectively changing restriction (3.4).

Uniqueness of Solution in the Supplier Problems

Stating the gradient of the objective function of supplier i's maximization problem yields a vector where the highest-order entries are linear. The gradient is shown in Equation 3.6. The first set of rows are associated with the sales variables s_{ni}. The second set of rows are associated with the production variables q_{ni}. The third and final set of rows are associated with flow routing variables f_{nmi}. Rows in set one and two have a linear and a constant term, while the final set has constant values in each row.

$$\nabla PROFIT_i = \begin{bmatrix} -A_1 s_{1i} + (B_1 - A_1(\sum_{k\in P} s_{1k})) \\ -A_2 s_{2i} + (B_2 - A_2(\sum_{k\in P} s_{2k})) \\ \vdots \\ -A_{|N|} s_{|N|i} + (B_{|N|} - A_{|N|}(\sum_{k\in P} s_{|N|k})) \\ -2C_{21i}^{prod} q_{1i} - C_{11i}^{prod} \\ -2C_{22i}^{prod} q_{2i} - C_{12i}^{prod} \\ \vdots \\ -2C_{2|N|i}^{prod} q_{|N|i} - C_{1|N|i}^{prod} \\ \alpha_{11} \\ \alpha_{12} \\ \vdots \\ \alpha_{|N||N|} \end{bmatrix} \tag{3.6}$$

The Hessian of the problem of supplier i is shown in Equation (3.7). It has non-zero entries only on the diagonal resulting from the quadratic term in the sales profit expression and the production cost expression. The flow-routing terms are linear in the supplier objective function (3.1), so these entries are zero in the Hessian.

$$\mathbf{H} = \begin{bmatrix} -2A_1 & 0 & \dots & \dots & \dots & \dots & \dots & \dots & 0 \\ 0 & \ddots & \vdots & \dots & \dots & \dots & \dots & \dots & \vdots \\ \vdots & \dots & -2A_{|N|} & \dots & \dots & \dots & \dots & \dots & \vdots \\ \vdots & \dots & \dots & -2C_{21i}^{prod} & 0 & \dots & \dots & \dots & \vdots \\ \vdots & \dots & \dots & 0 & \ddots & \dots & \dots & \dots & \vdots \\ \vdots & \vdots & \dots & \dots & \dots & -2C_{2|N|i}^{prod} & 0 & \dots & \vdots \\ \vdots & \dots & \dots & \dots & \dots & 0 & 0 & \dots & \vdots \\ \vdots & \dots & \dots & \dots & \dots & \dots & \dots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & \dots & \dots & \dots & \dots & 0 \end{bmatrix} \tag{3.7}$$

Assuming that the quadratic production cost parameter is strictly positive for all suppliers ($C_{2ni}^{prod} > 0 \forall i \in P$) **and** that every market inverse demand function has a strictly

positive slope ($A_n > 0 \forall n$) the sub-matrix regarding supply and production is negative definite. This means that if we were to remove the flow variables, the problem of supplier i would have a unique solution in sales and production, but with the linear flow terms we have no such guarantee, and there might be multiple solutions to the supplier problems given market prices.

Uniqueness With Production in One Node

If a supplier i has production in a single node $n^i \in N$ it is possible to replace the production variable q_{ni} with the sum of sales $\sum_{n \in N} s_{ni}$.

If we ignore what nodes the flow is routed through, and only look at how the transportation costs affect sales and production, we can show that the flow variables f_{nmi} becomes redundant as well: As the supplier is maximizing its profits, it will route flow in the cheapest possible way. In equilibrium, all paths from production to sale will be priced based on the Wardrop principle, i.e. all used paths between given source and destination nodes will have the same path transmission cost, and all unused paths will have a cost higher than or equal to this cost. This means we can set transportation costs per unit flow equal to the cost of the cheapest paths between the production node and the nodes where commodities are sold. As we are ignoring what nodes the flow is routed through, this means that the flow variables can be removed and substituted by a linear sales cost term with cost parameters equal to the aggregated link prices of one of the cheapest paths between the production node and the sales nodes.

With both the production and flow variables replaced by sales terms, the resulting Hessian of supplier i is shown in Equation (3.8). This matrix is negative definite, which means that the optimal solution to the maximization problem of supplier i is *unique in terms of sales and production when suppliers have production in only one node.* There might be several solutions in terms of how the flow is routed, but the production and sales will not be affected by this as the transportation costs will be the same due to the Wardrop principle.

$$\mathbf{H} = \begin{bmatrix} -2A_1 - C^{prod}_{2n^i i} & 0 & \dots & 0 \\ 0 & -2A_2 - C^{prod}_{2n^i i} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & -2A_{|N|} - C^{prod}_{2n^i i} \end{bmatrix} \tag{3.8}$$

3.2.2 The Transmission System Operator Problem

The transportation sector is, in contrast to the suppliers, modelled as perfectly competitive. This enables us to model this sector as one transmission systems operator (TSO) that maximizes the value of the transportation services. The TSO will then decide on supplied transportation services (g_{nm}) in order to maximize it's profits.

$$\max_{g_{nm}} \quad \sum_{n\in N} \sum_{m\in o(n)} [\alpha_{nm} g_{nm} - (C^{trans}_{1nm} + C^{trans}_{2nm} g_{nm}) g_{nm}] \tag{3.9}$$

$$\text{s.t.} \quad g_{nm} \leq CAP^{trans}_{nm} \qquad (\beta^{tso}_{nm}) \quad \forall n \in N, m \in o(n) \tag{3.10}$$

Expression (3.9) describes the objective of the TSO, which is maximizing profit earned from transporting fuel in the network. Revenues consist of the flow quantities times the price α_{nm}, the dual of the market clearing conditions (3.13).

Since the TSO only sees the price as a constant term, and thus is ignorant on his effect on the price, he acts as perfectly competitive. The only constraint on the TSO is that the flow g_{nm} allowed on the link between n and m must not exceed the capacity of the link. This is expressed in Constraint (3.10).

Uniqueness of Solution in the TSO Problem

The Hessian of the TSO objective function is shown in Equation (3.11). It is negative-definite for every decision the TSO makes, which means that the TSO problem has one unique solution given market prices.

$$\mathbf{H} = \begin{bmatrix} -2C_{211}^{trans} & 0 & \dots & 0 \\ 0 & -2C_{212}^{trans} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & -2C_{2|N||N|}^{trans} \end{bmatrix} \tag{3.11}$$

An Unconstrained Alternative of the TSO Problem

So far we have assumed a fixed constraint on capacity enforced by a finite number. An alternative approach to restricting link flow is to have asymptotically growing cost curves. For such a cost curve to be effective it is likely that the complexity of the objective function must be increased. However, the only constraint in the TSO problem will be gone, resulting in an unconstrained optimization problem. We do not present an unconstrained alternative of the TSO problem here, but merely present the concept. We will get back to this in Section 3.3.

3.2.3 Market Clearing Constraints

To describe the dependencies between the different players and the markets they act in, we need market clearing constraints. These constraints are not part of any players problems, but ensure market balance. One set of constraints describes the commodity price in each market. These are described in Equations (3.12).

$$\pi_n = B_n - A_n \sum_{i \in P} s_{ni} \quad (\pi_n) \quad \forall n \in N \tag{3.12}$$

By setting the price equal to the inverse demand curve we ensure that all commodity markets clears. This is required in an equilibrium: no market can have a price which deviates from the inverse demand curve.

In addition to the commodity market balance constraint we require the flow over each link to equal the sum of all the suppliers flows. This is shown in Equations (3.13).

$$g_{nm} - \sum_{i \in P} f_{nmi} = 0 \quad (\alpha_{nm}) \quad \forall n \in N, m \in N \tag{3.13}$$

As with the Equation (3.12) these constraints ensures that the transportation market clears, i.e. aggregated transportation demand set by the suppliers ($\sum_{i \in P} f_{nmi}$) equals transportation services set by the TSO (g_{nm}). Setting the transportation price α_{nm} as the dual in Equations (3.13) ensures that the price represent the highest evaluation of violating the constraint by one unit among all the players (see Section 2.1.1). By doing this we get transportation prices that corresponds to what the supplier with the highest evaluation of flow over that link is willing to pay, while at the same time satisfying the TSO problem.

3.3 An Equilibrium Model for the Whole Market

Looking at the objective functions for the the suppliers (3.1) and for the TSO (3.9) we can acknowledge that they are continuously differentiable and concave. All constraints, both in the players' problems and in the market clearing constraints, are affine. This means that the joint feasible region of all constraints are convex and that the Karush-Kuhn-Tucker conditions for them are both necessary and sufficient. Then, by finding a solution satisfying the joint KKT conditions for the actor problems including the market clearing constraints, we will find a Nash-Cournot equilibrium for the market: every player is making the best decision he can given the decisions of other players.

Equations (3.14)–(3.22) shows the joint KKT conditions for all actors along with the market clearing constraints, and represents our complete model. Models presented as aggregated complementarity constraints like this are called mixed complementarity problems (MCP).

Complementarity Conditions derived from the Supplier Problems

$$0 \leq -(B_n - A_n \sum_{k \in P} s_{nk}) + A_n s_{ni} + \sigma_{ni} \perp \quad s_{ni} \geq 0 \quad \forall n \in N, i \in P \tag{3.14}$$

$$0 \leq C^{prod}_{1ni} + 2C^{prod}_{2ni} q_{ni} + \beta^{sup}_{ni} - \sigma_{ni} \perp \quad q_{ni} \geq 0 \quad \forall n \in N, i \in P \tag{3.15}$$

$$0 \leq \alpha_{nm} + \sigma_{ni} - \sigma_{mi} \perp \quad f_{nmi} \geq 0 \quad \forall n \in N, m \in o(n), i \in P \tag{3.16}$$

$$0 \leq CAP^{prod}_{ni} - q_{ni} \perp \quad \beta^{sup}_{ni} \geq 0 \quad \forall n \in N, i \in P \tag{3.17}$$

$$0 = s_{ni} - q_{ni} + \sum_{m \in o(n)} f_{nmi} - \sum_{m \in i(n)} f_{mni} \perp \quad \sigma_{ni}, free \quad \forall n \in N, i \in P \tag{3.18}$$

Complementarity Conditions derived from the TSO Problem

$$0 \leq -\alpha_{nm} + C^{trans}_{1nm} + 2C^{trans}_{2nm} g_{nm} + \beta^{tso}_{nm} \perp \quad g_{nm} \geq 0 \quad \forall n \in N, m \in o(n) \tag{3.19}$$

$$0 \leq CAP^{trans}_{nm} - g_{nm} \perp \quad \beta^{tso}_{nm} \geq 0 \quad \forall n \in N, m \in o(n) \tag{3.20}$$

Complementarity Conditions derived from the Market Clearing Constraints

$$0 = \pi_n - B_n + A_n \sum_{i \in P} s_{ni} \perp \quad \pi_n, free \quad \forall n \in N \tag{3.21}$$

$$0 = g_{nm} - \sum_{i \in P} f_{nmi} \perp \quad \alpha_{nm}, free \quad \forall n \in N, m \in o(n) \tag{3.22}$$

A Note on the Transmission Price α_{nm}

After having stated the equilibrium model it is possible to show the marginal cost pricing on the link transmission cost discussed in Section 3.1.1, assumption 8.

If the flow across link (n, m) is strictly below capacity then complementarity condition (3.20) will force

$$\beta^{tso}_{nm} = 0$$

Further, if the link flow g_{nm} is positive, complementarity condition (3.19) will force

$$\alpha_{nm} = C^{trans}_{1nm} + 2C^{trans}_{2nm} g_{nm}$$

which is indeed the marginal cost of the TSO. Only if the flow g_{nm} is at capacity is β^{tso}_{mn} allowed to be positive, and α_{nm} will increase accordingly. If g_{nm} is zero, then complementarity condition (3.19) will allow

$$\alpha_{nm} \leq C^{trans}_{1nm}$$

If an unconstrained alternative of the TSO problem is used then a link will always be priced at marginal cost. With asymptotically growing cost curves the marginal cost grow asymptotic as well.

Chapter 4

Decomposing the Model

To develop a solution algorithm we wish to decompose the equilibrium model into independent individual actor problems that uses exogenous market prices. The actor problems presented in the previous chapter have several dependencies between them beyond the market prices. In this chapter we describe the dependencies between the actors, the influence these dependencies have on the solutions of the actor problems and suggest reformulations that removes the dependencies to create independent problems.

Section 4.1 presents the two dependencies.In Sections 4.2 and 4.3 we analyze the dependencies and present reformulations which removes them from the model.

4.1 Independent Actor Problems with Exogenous Prices

We will decompose the model by making the actors' problems independent of each other. One way to do this is having market prices as exogenous parameters to the problems of the individual actors. This will enable us to determine what decisions the actors would make given a set of market prices. Philosophically this makes sense, as this resembles how many non-cooperative markets functions in real life: actors making decisions given market prices.

Take care to not associate the exogenous prices discussed in this chapter with a

Bertrand assumption on the suppliers strategic behaviour. Our purpose is to reformulate the problem to have all prices as parameters, but let the suppliers still retain their Cournot behaviour: they choose their own production which maximize profits given the production of other actors. Section 4.2 will show that we are able to keep this strategic aspect while still reformulating the suppliers to be price-taking entities.

A price-based decomposition separates the problems of finding equilibrium market prices and equilibrium actor decisions. We will later utilize this in an algorithm for finding a market equilibrium, where the algorithm will suggest the prices and the actors will make decisions based on these prices.

Looking at the model stated in Chapter 3, we see that the constraints binding the different actors together are the two market clearing conditions. Condition (3.12) binds together all the suppliers' problems. We call this the Commodity Dependency, as it is related to the commodity price. Condition (3.13) binds together the suppliers problems and the TSO problem. We call this the Transportation Dependency.

In Section 4.2 we describe the commodity dependency and present a reformulated suppliers problem to show how the dependency can be handled. In Section 4.3 we describe the transportation dependency, and discuss how to handle it in a decomposed framework.

4.2 The Commodity Dependency

The inverse demand function of (3.12) aggregates supplier sales to find the commodity prices.

$$\pi_n = B_n - A_n \sum_{i \in P} s_{ni}$$

The right-hand-side of the expression is also used as the price term of the suppliers' objective function (3.1). This renders one suppliers' decisions dependent on the decisions made by all other suppliers. We want to make the supplier problems independent of each other and only rely on exogenous prices. When we present a set of commodity prices and transportation prices to a supplier, we want him to make an optimal production decision given that set of prices. Across all suppliers we then find the total supply

to each market. Comparing this supply to the demand generated in the markets by the same set of prices allows us check whether or not the set of prices represents equilibrium prices. If supply equals demand in every market then the price set is a candidate for equilibrium prices.

In other words we want to make the suppliers decide equilibrium production and sales independently when given equilibrium prices.

4.2.1 Removing the Commodity Dependency

As discussed in Section 3.2.1 we cannot simply use the inverse demand function to substitute the price π_n into the supplier objective function. This would result in perfectly competitive players, violating the assumptions we made in Section 3.1.1. To make a proper decomposition, we need to keep information about the markets (A_n and/or B_n) in the supplier objective function while removing the decisions of the other players (s_{nk}).

By looking at the MCP formulation given by conditions (3.14)–(3.22) in Section 3.3, we see that condition (3.14) contains the inverse demand curve. Because market clearing condition (3.21) must hold (and the price π_n corresponds to that of a cleared commodity market n) we can substitute the inverse demand curve with the corresponding commodity price π_n, rendering the equivalent formulation shown in condition (4.1).

$$0 \leq -\pi_n + A_n s_{ni} + \sigma_{ni} \perp s_{ni} \geq 0 \forall n \in N, i \in P \tag{4.1}$$

Acknowledging this enables us to reformulate the original maximization problem of the suppliers as well, and their new objective functions are shown in expression (4.2).

$$\max_{s_{ni}, q_{ni}, f_{nmi}} \quad \sum_{n \in N} [(\pi_n - \frac{A_n}{2} s_{ni}) s_{ni} - (C^{prod}_{1ni} + C^{prod}_{2ni} q_{ni}) q_{ni}] \quad - \sum_{n \in N} \sum_{m \in o(n)} \alpha_{nm} f_{nmi} \tag{4.2}$$

Here the original revenue term $(B_n - A_n \sum_{k \in P} s_{nk}) s_{ni}$ is exchanged with $(\pi_n - \frac{A_n}{2} s_{ni}) s_{ni}$. Taking the partial derivative with respect to s_{ni} yields $(B_n - A_n \sum_{k \in P} s_{nk}) - A_n s_{ni}$ and $\pi_n - A_n s_{ni}$. Again, because of the market clearing condition (3.21), these are equiv-

alent. Because the Karush-Kuhn-Tucker conditions are the same, and necessary and sufficient for the problem at hand, the solutions found by the two formulations have to be the same. Thus, the suppliers decisions will not change from the reformulation.

With this reformulation, we have a formulation of the supplier problems where their sales and production decisions are independent of each other. The flows are still influenced by the capacity constraints and cost function of the TSO (the Transportation Dependency), but if we can give equilibrium prices π_n and α_{nm} as parameters, then the suppliers will make independent equilibrium decisions on their sales and production.

Given equilibrium prices the gradient and the hessian of the objective function is unchanged. This means we can conclude that the suppliers problem with the new objective functions can be solved independently with exogenous prices and render equivalent production and sales decisions as they would in the original model. Sadly, the routing decisions is not guaranteed to be unique if multiple paths exists between a production node and consumption node. Due to the Wardrop principle all used routes from origin to destination must be equally priced in an equilibrium, and the supplier has no incentives for splitting his flow correctly across the routes. This is related to the transportation dependency, so we explain and analyze this in Section 4.3.1.

4.3 Transportation Dependency

Recall the dependency between the supplier flow decisions f_{nmi} and the total link transmission g_{nm} given by (3.13). The transportation market must clear so the aggregated transportation demand set by the suppliers must equal the transportation supply set by the TSO. This renders a dependency on the flow variables of the TSO and all the suppliers.

The original formulation of the supplier problems and the TSO problem (presented in Chapter 3) has the suppliers decide on the routing of the flow. This means each supplier decides on what links the flow from a supply node to a market node is sent through, given a price for each link. An alternate approach would be to let the supplier see a price of sending flow between supply and demand nodes, but let the TSO decide how to route the flow.

In this section we will see how the suppliers and TSO problem can be reformulated in such a way that the routing decisions are left to the TSO, and how this corresponds to the equilibrium model stated in Section 3.3. Model assumptions, parameters and parameters are kept the same unless specified otherwise.

Before we present the alternate problem formulations, we discuss the problem of having the suppliers route the flow when trying to decompose the model to get independent actor formulations.

4.3.1 The Problem with the Suppliers choosing Routes

The suppliers problem stated in Chapter 3 does not have a unique solution when it comes to the flow variables, i.e. there are several optimal routing decisions for the suppliers when given link prices. There are two problems with this:

1. As the transportation costs for the suppliers (given by the transportation prices α_{nm}) are linear, the marginal cost of transportation is constant, leaving no incentive for the suppliers to split its flow along several paths. This is a problem since the link price is dependent on the total link flow, and due to the quadratic component of the TSO cost, sending flow from a source to a destination along two or more distinct paths can be cheaper than sending all the flow along just one.

2. The suppliers are not aware of the capacity restrictions on link flow. This is not a problem when all flow from supply to demand are routed along the same path, as the equilibrium transportation price should be high enough to keep the flow below or at capacity, but there could be several paths from supply to demand where one or more paths are restricted by capacity causing a need to split the flow along several paths. In the latter case the supplier only seeing the linear price has no incentive to split its flow, and will most likely send all of the flow along one of the paths.

These two problems show that solving the formulation of the supplier problems presented in Chapter 3 with exogenous prices (with the commodity dependency removed as discussed above) does not necessarily give equilibrium flow decisions when

there is a need for suppliers to split their flow along several paths. If the network is sparse enough to have at most one path between supply and demand nodes these problems will not come into play. Many energy market networks are not sparse however, so having to rely on this property to solve the model is not wanted. We wish to prepare for the general case, and the next sections will suggest a reformulation of both the suppliers problems and the TSO problem to work around the transportation dependency by moving routing decisions from the suppliers to the TSO: the suppliers does not care about the routing of the flow, but rather the price they must pay.

4.3.2 Alternate Suppliers Problem

In this alternate formulation the supplier decides on where produced commodities should be sold, not on what links it flows through. Thus the node-balance equation in each node (3.3) will not be part of the suppliers problem. The suppliers must only ensure that they produce as much as they sell.

We introduce two new indices to our model. s and d are used to reference nodes in N, but are used to describe supply to demand node variables and parameters. When describing a direct link between two nodes, we will still use n and m.

Reformulated Variables

Variable	Description
f_{sdi}	Transportation demand from supply node s to demand node d set by producer i
α_{sd}	Transportation price from supply node s to demand node d

After changing the definition of f and α we can state the alternate supplier problems in Equations (4.3) trough (4.8).

$$\max_{s_{ni},q_{ni},f_{sdi}} \quad \sum_{n\in N}[(\pi_n - \frac{A_n}{2}s_{ni})s_{ni} - (C^{prod}_{1ni} + C^{prod}_{2ni}q_{ni})q_{ni}] - \sum_{s\in N}\sum_{d\in N}\alpha_{sd}f_{sdi} \tag{4.3}$$

$$\text{s.t.} \quad 0 \le q_{ni} \le CAP^{prod}_{ni} \qquad (\beta^{sup}_{ni}) \quad \forall n \in p(i) \tag{4.4}$$

$$\sum_{d\in N} f_{sdi} \le q_{si} \qquad (\sigma_{si}) \quad \forall s \in N \tag{4.5}$$

$$s_{di} \le \sum_{s\in N} f_{sdi} \qquad (\sigma_{di}) \quad \forall d \in N \tag{4.6}$$

$$s_{di} \ge 0 \qquad \forall d \in N, i \in P \tag{4.7}$$

$$f_{sdi} \ge 0 \qquad \forall s,d \in N, i \in P \tag{4.8}$$

We have replaced constraint (3.3) with two new constraints. Inequality (4.5) ensures outflow from a node does not surpass production in that node. Inequality (4.6) ensures sales in a node does not surpass inflow to that node. Note that if supplier i produces and sells in the same node n then the transportation demand f_{nni} will be positive.

Uniqueness of Solution in the Alternate Supplier Problems

By looking at the objective function of the suppliers problem, we see that the flow variables only have a linear term giving zeroes along the diagonal of the Hessian (in the same way as the original formulation described in Section 3.2.1). This means that the Hessian is not negative definite, giving no guarantee of a unique solution to the suppliers problem. However, the change in the flow variable still removes the transportation dependencies described in Section 4.3.1 as the supplier no longer routes the flow itself, but instead decides on a transportation demand for the TSO to route according to the Wardrop principle and capacity constraints.

Also, if the supplier only have production in one node $n^i \in N$ the we can see that

there will be a unique solution for all variables: the production variable can be replaced by the sum of sales, and the transportation demand variables can be replaced by the corresponding sales variables. This yields the Hessian shown in 4.9 which is negative definite meaning the alternate supplier problem has one unique solution when the supplier only has production in node node.

$$\mathbf{H} = \begin{bmatrix} -A_1 - C^{prod}_{2n^i i} & 0 & \dots & 0 \\ 0 & -A_2 - C^{prod}_{2n^i i} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & -A_{|N|} - C^{prod}_{2n^i i} \end{bmatrix} \tag{4.9}$$

4.3.3 Alternate TSO Problem

The TSO now has to route the flow between origin and destination as requested by the solution to the suppliers problems. As mentioned in Section 2.3 this corresponds to solving a transportation equilibrium problem for a given set of transportation demand. We use a path based formulation because of the limited experience with link-based formulations in traffic equilibrium modelling.

New Sets and Indices

Set	Index	
K	k	All paths in the network
$K^{sd} \subset K$		All paths starting in node s and ending in node d
$K^{nm} \subset K$		All paths going through link (n, m)

New Variables

Variable	Description
g_{nm}	Link flow from n to m
f_{sdi}	Flow demand from node s to node d set by supplier i
p_k	Flow along path k

$$\min_{g_m} \quad \sum_{n\in N}\sum_{m\in M}(C^{trans}_{1nm} + C^{trans}_{2nm} g_{nm}) g_{nm} \tag{4.10}$$

$$\text{s.t.} \quad g_{nm} = \sum_{k\in K^{nm}} p_k \qquad \forall n, m \in N \tag{4.11}$$

$$g_{nm} \leq CAP^{trans}_{nm} \qquad \forall n, m \in N \tag{4.12}$$

$$\sum_{k\in K^{sd}} p_k = \sum_{i\in P} f_{sdi} \qquad \forall s, d \in N, s \neq d \tag{4.13}$$

The alternate TSO problem has the actor minimize the transportation costs while still fulfilling the source-destination demands from the suppliers. The objective function (4.10) is the sum of costs across all links in the network. Constraint (4.11) ensures that the flow across a link is equal to the sum of flows on all paths utilizing that link. Constraint (4.12) makes sure the link flow does not surpass its capacity. Finally, Constraint (4.13) ensures that the transportation demand given by the suppliers are fulfilled.

In the following Section, we will prove that every feasible instance of the Alternate Transmission System Operator Problem has a unique equilibrium solution in terms of link flow.

4.3.4 Equilibrium in the Alternate TSO Problem

As stated in Section 2.1.4 there may be none, one, or several equilibria to a model. In the following we state some conditions for an equilibrium to exist in our model and under which conditions there exists an unique equilibrium to the TSO problem.

We will prove that the problem described by Equations (4.10) through (4.13) is either infeasible or has a unique equilibrium solution $\tilde{g} = [g_{nm} \quad \forall (n \in N, m \in o(n))]$ given the following conditions:

- $C^{trans}_{2nm} > 0 \quad \forall n \in N, m \in o(n)$
- $CAP^{trans}_{nm} > 0 \quad \forall n \in N, m \in o(n)$

If a link (n, m) has a capacity of 0 it is not usable. If a data set has $CAP^{trans}_{nm} = 0$ for

at least one link (n, m) we can create an equivalent problem where the link is removed, solve it, and apply the solution to the original problem.

Let S be the feasible region of link flows.

$$S = \{g_{nm} \mid 0 \leq g_{nm} \leq CAP_{nm}^{trans} \quad \forall n \in N, m \in o(n)\} \tag{4.14}$$

Let D be the set of link flows which satisfies the demand a_{sd}.

$$D = \{g_{nm} = \sum_{k \in K^{nm}} p_k \mid \sum_{k \in K^{sd}} p_k \geq f_{sd} \quad \forall (s,d) \in N \times N\} \tag{4.15}$$

Let $\tilde{g}$ be an equilibrium, which satisfies the Wardrop principles stated in Section 2.3: the cost of all used routes between s and d must be equal, and lower than any unused routes. Additionally, the system cost is a minimum. Notice that $\tilde{g}$ must be an element of D in order to be an equilibrium: if the link flows do not satisfy the transportation demand, then it is by definition not an equilibrium. Notice also that $\tilde{g}$ must be an element of S because a link-flow scheme that violates the link capacities does not fulfill the requirements of an equilibrium.

$$\tilde{g} \in D \cap S \tag{4.16}$$

Let $c : S \rightarrow R_+^{n*n}$ be a function mapping the flows of a link to the cost incurred.

$$c = \{c_{nm} = C_{1nm}^{trans} g_{nm} + C_{2nm}^{trans} g_{nm}^2 \quad \forall n \in N, m \in o(n)\} \tag{4.17}$$

Existence of Equilibrium

Smith (1979) states that if (1) c is continuous, (2) D is a closed subset of R_+^{n*n} and (3) D is a subset of S, then a Wardrop equilibrium $\tilde{g}$ must exist in the intersection between D and S.

(1) is true for every possible transmission cost parameters. The link cost functions are quadratic with respect to the transmitted quantity.

(2) is true in all cases as well, assuming that the capacity CAP_{nm}^{trans} is finite. If a link

(n,m) has an infinite capacity then we can create an equivalent problem with

$$CAP_{nm}^{trans} = \sum_{s\in N}\sum_{d\in N} f_{sd}$$

This will not restrict the original problem, so a solution to the equivalent problem is also a solution to the original problem.

(3) is the feasibility condition, and states that the demanded flows D is a subset of the supplied link capacities S. Sadly we cannot guarantee that the flows requested by the suppliers constitute a feasible solution to the TSO problem. However, in our context that does not matter. If a feasible solution does not exists, then we know that the transportation prices α_{nm} is set too low for at least one link. We will use this in our algorithm: we know that if we can find the equilibrium values of α_{nm} for all links, then the TSO problem will have a feasible solution.

Uniqueness of an Equilibrium

Theorem 3 by Smith (1979) states that a Wardrop equilibrium $\tilde{g}$ is unique if the dot product between the cost difference vector and the flow difference vector is strictly positive for any distinct link-flow vectors x and z as shown in Inequality (4.18).

$$[c(x)-c(z)]\cdot(x-z) > 0 \tag{4.18}$$

Using definition (4.17) to write this requirement in our terminology yields the following expression:

$$[c(x)-c(z)]\cdot(x-z) \tag{4.19}$$

$$= \sum_{n\in N}\sum_{m\in o(n)} [(C_{1nm}^{trans}x_{nm} + C_{2nm}^{trans}x_{nm}^2) - (C_{1nm}^{trans}z_{nm} + C_{2nm}^{trans}z_{nm}^2)][x_{nm}-z_{nm}] \tag{4.20}$$

$$= \sum_{n\in N}\sum_{m\in o(n)} [C_{1nm}^{trans}(x_{nm}-z_{nm}) + C_{2nm}^{trans}(x_{nm}^2 - z_{nm}^2)][x_{nm}-z_{nm}] \tag{4.21}$$

x and z are both defined as link-flow vectors with elements greater than or equal to 0 defined by Equation (4.14). Recall also that we assume strictly positive cost parameters

C_{2nm}^{trans}. Looking at Expression (4.21) we can associate each link (n,m) to one of three cases shown in Expressions (4.22) trough (4.24).

$$x_{nm} > z_{nm} \implies \begin{cases} C_{1nm}^{trans}(x_{nm} - z_{nm}) + C_{2nm}^{trans}(x_{nm}^2 - z_{nm}^2) > 0 \\ x_{nm} - z_{nm} > 0 \end{cases} \tag{4.22}$$
$$\implies [C_{1nm}^{trans}(x_{nm} - z_{nm}) + C_{2nm}^{trans}(x_{nm}^2 - z_{nm}^2)][x_{nm} - z_{nm}] > 0$$

$$x_{nm} < z_{nm} \implies \begin{cases} C_{1nm}^{trans}(x_{nm} - z_{nm}) + C_{2nm}^{trans}(x_{nm}^2 - z_{nm}^2) < 0 \\ x_{nm} - z_{nm} < 0 \end{cases} \tag{4.23}$$
$$\implies [C_{1nm}^{trans}(x_{nm} - z_{nm}) + C_{2nm}^{trans}(x_{nm}^2 - z_{nm}^2)][x_{nm} - z_{nm}] > 0$$

$$x_{nm} = z_{nm} \implies [C_{1nm}^{trans}(x_{nm} - z_{nm}) + C_{2nm}^{trans}(x_{nm}^2 - z_{nm}^2)][x_{nm} - z_{nm}] = 0 \tag{4.24}$$

Because the link patterns x and z must be distinct, at least one link-pair (x_{nm}, z_{nm}) will belong to either (4.22) or (4.23). Thus the sum over all link pairs must be positive and inequality (4.18) holds, allowing us to conclude that if an instance of the alternate TSO problem (4.10) trough (4.13) has a feasible solution, it is unique.

4.3.5 The Connection between Link-Flow Uniqueness and Path-Flow Uniqueness

After showing that every feasible instance of the Alternate TSO problem has a unique solution at the link-flow level it should also be assessed if this has any implications on the market equilibrium solution. We will first consider the implications of a unique link-flow in terms of routing variables f_{nmi} and secondly in terms of the link transmission price α_{nm}.

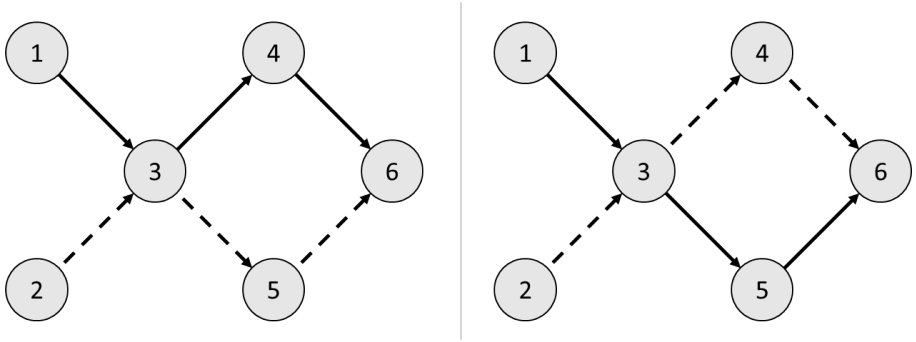

Figure 4.1: Two routing of flows in a congested network with equal cost. Two suppliers in node 1 and 2 produce and sell the same amount of fuel in node 6. In the left network, supplier 1 uses the route 1-3-4-6, while supplier 2 uses 2-3-5-6. An equal routing with regards to the transportation cost and the origin-destination demand is shown in the right network. Supplier 1 uses 1-3-5-6, and supplier 2 uses 1-3-4-6. The Wardrop principle states that the two routes between node 3 and node 6 must be equally priced, and so the suppliers do not care where their flow is routed.

Unique Link-Flow and Routing Properties

We can now show that a unique link flow solution does not imply a unique routing for the suppliers. It is easy to construct an instance where the link flow variables g_{nm} are unique, but the flow f_{nmi} is not uniquely routed. This is shown and explained in Figure 4.1.

Even if there are several routing options for the suppliers, it will not matter for properties of the equilibrium solution other than for how a supplier distributes produced quantities between the paths it's routed along. In fact, if we can show that multiple cost-equal routing decisions are possible in a network, then several equilibrium solutions will exist for the problem/model presented in Chapter 3. All actors' profits will be the same, and the quantity sold in each end market is equal.

Thus, given the equilibrium prices an optimal (hence feasible) solution of the path flow variables can easily be decomposed into a solution for the suppliers flows.

Unique Link-Flow and Link Pricing

Recall that the congestion rent β^{tso}_{nm} on a link (n,m) is the premium which the TSO can charge above the link marginal cost. This is discussed in Section 3.3.

If two or more links on a single path have a positive congestion rent, then an infinitely number of equilibria can exist to the problem. This is shown by Egging (2010). The sum of the congestion rent will have a fixed value but the distribution of the rent across the congested links does not matter.

4.3.6 Adding the Congestion Rent to the TSO Cost Curve

A possible reformulation of the TSO problem is to add the congestion rent and the linear coefficient of the TSO cost curve. Let γ_{nm} be the congestion rent on link (n,m) defined as

$$\gamma_{nm} = max(\alpha_{nm} - C^{trans}_{1nm} - 2C^{trans}_{2nm} g_{nm}, 0)$$

Consider the alternate TSO objective function shown in Expression (4.25).

$$\min_{a_{nm}} \quad \sum_{n\in N} \sum_{m\in M} (C^{trans}_{1nm} + \gamma_{nm} + C^{trans}_{2nm} a_{nm}) a_{nm} \tag{4.25}$$

An advantage to this is that the transportation prices become equal to the TSOs marginal costs. Given this reformulated problem it is possible to solve the alternate TSO problem given by expressions (4.10) trough (4.13) without the capacity restriction (4.12). This is due to the Wardrop principles: Given equilibrium congestion rents and transportation demand, no links will be used beyond capacity because the congestion rents will make this more expensive. If it were cheaper to route flow above capacity, the congestion rents would not have equilibrium properties.

Chapter 5

Algorithms for Solving Energy Market Equilibrium Models

This chapter describes a price-based algorithm for solving the equilibrium model given in Chapter 3. We utilize the decompositions described in Chapter 4 to create independent actor optimization problems given market prices as exogenous parameters, and develop a search strategy in order to find a set of equilibrium prices and corresponding actor decisions given by the optimization problems.

We base our algorithm on adjusting the market prices in the directions indicated by the actors' decisions, i.e. after solving the actor problems for a set of prices we look at the actors of the commodity and transportation markets to see how the supply side of the market corresponds to the demand side of the market. In cases of over supply we adjust prices down, and in cases of under-supply we adjust prices up. If supply equals demand (the market clears) there are no incentives for the actors to change prices or quantities, so the market is in equilibrium. This response-based adjustment represents a local search because we start in an initial solution and iteratively move to an improving solution.

Section 5.1 will briefly describe existing techniques for solving equilibrium models. Section 5.2 provides an overview of the algorithm developed, and Sections 5.3

trough 5.5 presents three different alternative algorithms differing in both what decomposition is used as well as search strategy. Finally, Section 5.6 will describe some important implementation details of the algorithm as a C++ program.

5.1 Current Solution Techniques

After having formulated our model in Chapter 3 we proved that the Karush-Kuhn-Tucker conditions are both sufficient and necessary to find an optimal solution to the equilibrium problem. Many equilibrium models share this property, and as we mentioned in Chapter 1 a widely used method to solve energy market equilibrium problems are to state the Karush-Kuhn-Tucker conditions for the problem, reformulate it into a mixed complementarity problem (MCP), and use an MCP solver to find a solution. Two such solvers are PATH (see Ferris and Munson (1998)) and MILES (see Rutherford (1993). PATH includes more options to solve larger models when compared to MILES, such as a user-defined starting point and a self-restarting mechanism.

PATH and MILES share the same basic functionality. For all complementarity conditions one side is fixed to 0. The resulting sub-problem is then solved and this gives an indication on which sides of the complementarity conditions to fix for the next sub-problem. With enough iterations, the solution to the sub-problem will typically converge to the equilibrium solution for the complementarity problem. The sub-problems are solved using an iterative algorithm as well. In complementary solver jargon, the iterations needed to solve a sub-problem is called minor iterations, while solving a single sub-problem is called a major iteration.

The main advantages of using solvers like PATH and MILES are the broad range of models they can solve. In fact, if the Karush-Kuhn-Tucker conditions are meaningful for the problem then a complementarity solver can solve the problem. Because the solvers apply to a broad range of models they are robust, and much effort have been made to optimize the underlying algorithms.

General solvers like PATH or MILES also has some disadvantages. Basing the model solution on the Karush-Kuhn-Tucker conditions puts some limitations on the model properties. The modellers must be able to prove that the KKT-conditions are mean-

ingful for the problem they are modelling. Proving this is not always trivial. Also, if the KKT-conditions must be meaningful in order for a modeller to solve his model, this may limit the scope of the model as functionality that breaks the conditions for the KKT conditions to be meaningful must be omitted.

Recall the discussion we had in Chapter 1 about generality of a solver and the solution time. Because PATH and MILES apply to a broad range of problems, there is probably a solution time advantage to gain from using more specialized software. In the rest of this Chapter we will present an algorithm which does not depend on the Karush-Kuhn-Tucker conditions, but where the parameters of the actors are input to the solution algorithm.

5.2 Algorithm Overview

The algorithm presented in this chapter is an iterative algorithm which uses the decomposition presented in Chapter 4. The actors problems are connected through a set of prices. We will let the search algorithm control the prices and when these prices are presented to the actors they will make independent decisions. Based on the decisions of the actors the algorithm determines how to change the prices in order to get closer to equilibrium. An equilibrium solution is found when all market clearing constraints holds (supply equals demand in all commodity markets and transportation prices are equal to TSO marginal costs plus congestion rents that ensure all links have a flow less than or equal to the link capacity).

We like to use the analogy of an auction to describe our algorithm. An auctioneer (the algorithm) will call out a set of prices and the participants (the model actors) will give their responses. These responses gives the auctioneer an indication of what prices he should call out next. This continues until the auctioneer calls a price for which the markets clear, i.e. no actor has incentive to change the prices.

5.2.1 Importance of unique solution for individual actors

In Chapters 3 and 4 we stressed the property of unique solutions to the actor problems given market prices. This is important in the context of our algorithm because we base the price adjustment on the solutions to the actor problems. If the actors have multiple solutions given market prices, and only a subset of the solutions correspond to the equilibrium solution of the model, the algorithm might adjust the prices towards a non-equilibrium solution. If a set of prices gets a solution to an actor problem that pulls the prices in one direction, and a slightly adjusted set of prices gets another solution pulling the prices in another direction, the algorithm run the risk of changing prices back and forth and not converge towards a set of equilibrium prices. With current optimization solvers we only get one of possibly multiple solution, so the algorithm must be able to know if the solutions given by the solver are solutions that will pull the prices towards equilibrium. If not, the algorithm must know how to adjust the actor problem solutions to a set that will.

5.2.2 Three Alternative Algorithms

We have developed three different alternative algorithms. Alternative 1 uses the original formulation of the TSO and supplier problems described in Chapter 3, with the modification of the supplier problems described in Section 4.2.1 to remove the commodity dependencies. This decomposition still has the transportation dependencies described in Section 4.3.1 between the TSO and suppliers, and the idea is to handle the transportation dependencies in the search algorithm rather than through the decomposition. Alternative 2 and 3 is based on the alternative formulations of the TSO and supplier problems presented in Section 4.3, and thus have the dependencies described in Section 4.3.1 removed.

Similar to the commercial solvers presented in Section 5.1 our algorithm consist of a major iteration and a minor iteration. Unlike the commercial solvers however, the major and minor iterations are related to the actors problems. For this reason, we call the iterations “TSO iterations” and “commodity iterations”. In a commodity iteration the commodity prices are adjusted with fixed transportation prices until supply equals

demand in all markets. In a TSO iteration the commodity market are cleared is cleared for the current transportation prices (using commodity iterations) and then the transportation prices are adjusted depending on the resulting quantities set by the supplier problems.

5.3 Alternative 1

This section is about Alternative 1 of our solution algorithm. Section 5.3.1 presents the algorithm outline, and Section 5.3.2 describes how the commodity price is adjusted. Recall that Alternative 1 has the suppliers routing their flows, and the dependencies discussed in Section 4.3.1 needs to be handled explicitly; we describe the mechanism which does this in Section 5.3.3. Finally, Section 5.3.4 describes how to adjust the transportation prices.

5.3.1 Algorithm Outline

Pseudocode for the main steps of Alternative 1 are shown in Algorithm 2. Alternative 2 and 3 will share the same basic structure. Note that lines 2 through 11 represents a TSO iteration, and lines 11 through 8 represents a commodity iteration.

```
   Data: Problem description
   Result: Equilibrium commodity and transportation prices and quantities
 1 Set initial prices;
 2 repeat
 3 |   repeat
 4 |   |   foreach supplier do
 5 |   |   |   Determine supplier bids given prices;
 6 |   |   end
 7 |   |   Adjust commodity prices to lower gap between supply and demand
   |   |   [Algorithm 3];
 8 |   until supplied quantities equal demand in each market;
 9 |   get bids with link capacity constraints enforced[Algorithm 4];
10 |   adjust transportation prices according to TSO problem [Algorithm 5];
11 until transportation prices satisfy TSO problem ;
```

Algorithm 2: Alt 1 - Main algorithm

Line 1: In order to start the search, the algorithm need some initial prices. These can

be a guess on equilibrium prices from some heuristic function, some random values or even zero for all prices. In order to make a guess likely to be close to the equilibrium price, we acknowledge that for market n the price will be between 0 and B_n, which is the intercept of the inverse demand curve. As an initial estimate, the midpoints of the inverse demand curves are used as initial commodity prices, that is $\pi_n = B_n/2$. Based on similar reasoning, the initial transportation prices are set to the TSO marginal costs with flows at half capacity, that is $\alpha_{nm} = C^{trans}_{1nm} + CAP^{trans}_{nm} C^{trans}_{2nm}$.

Line 5: With initial prices, the algorithm starts by letting the suppliers place bids on their production, sales and transportation variables, i.e. the quadratic optimization problem consisting of objective function (4.2) and constraints (3.2) through (3.5) is solved for all the suppliers with the current prices given as exogenous parameters.

Line 7: Having the suppliers bids we can look at the gap between supplied and demanded quantities for each market node, and adjust the commodity prices in order to close the gaps. Knowing what direction the commodity prices need to be changed is easy with a linear demand curve: the price should go down if we have over-supply and up if we have under-supply. How much the prices should be changed is more difficult, and we will get back to this in Section 5.3.2 where we discuss the commodity price adjustments in detail. The algorithm iterates between suggesting new commodity prices and evaluating the resulting bids until the gaps between supply and demand are *sufficiently small.* The notion of sufficiently small is due to computational reasons, and we elaborate in this in Section 5.6.3. The transportation prices are kept constant during lines 3 – 8.

Line 9: Next we look at the situation for the TSO. Before the TSO prices can be adjusted, the fact that all supplier bids up to this point are placed unaware of link capacities needs to be addressed (recall the problems with supplier routing flow from Section 4.3.1). This transportation dependency is handled by enforcing link capacity restrictions on the suppliers and noting how their bids are affected. The process of enforcing link capacities on suppliers is described in detail in Section 5.3.3.

Line 10: Since the TSO is modelled as perfectly competitive and only has capacity restriction we can omit solving the corresponding quadratic problem, as we know the

properties the transportation prices have in equilibrium by Equation (3.19). For each link nm the price will be set by one of three rules depending on the total link flow g_{nm}.

1. The flow is 0, and the price is lower than or equal to the linear coefficient of the cost function. ($\alpha_{nm} \leq C_{1nm}^{trans}$)

2. The flow is nonzero and below capacity with a price at marginal cost of the flow. ($\alpha_{nm} = C_{1nm}^{trans} + 2C_{2nm}^{trans} g_{nm}$)

3. The flow is at capacity with a price greater than or equal to the marginal cost at capacity. The premium above marginal cost is what we refer to as the congestion rent. ($\alpha_{nm} \geq C_{1nm}^{trans} + 2C_{2nm}^{trans} CAP_{nm}^{trans}$)

Using these properties the algorithm can adjust the prices directly without solving the TSO problem and look at market clearing gaps. Section 5.3.4 provides details for how the transportation prices are adjusted.

Line 11: If the transportation prices needed adjustments, a new iteration begins at line 3. Otherwise, we know that all current prices correspond to equilibrium prices because there are no gaps between supply and demand in commodity markets (by line 8) and the transportation prices satisfy the equilibrium properties (no change was needed, implying that the transportation prices correspond to properties 1, 2 and 3).

This concludes the search, and the algorithm has found an equilibrium solution.

5.3.2 Adjusting Commodity Prices

Pseudocode for the adjustment of one commodity price is shown in Algorithm 3. The prices for each node are adjusted independently of each other.

Line 2: If this line is reached, it means that the current price is equal to that of the inverse demand function given current supply bids. Equivalently, supplied quantities equals demand for the current prices, so the market clears for the current price. There is no incentive to change, so the new price is set equal to the current price. In the implementation, some slack can be tolerated in the equality on line 1. We elaborate more on the algorithm slack in Section 5.6.3.

Data: Supply quantity bids and price from this and previous iteration. Inverse demand function f^{-1}. Step length parameters $0 < step_1 < 1$ and $step_2 > 1$
Result: Adjusted commodity prices

```
1  if f^-1 (current supply) == current price then
2  |  new price = current price;
3  else
4  |  if previous price == current price then
5  |  |  new price = current price + step_1*(f^-1(current supply) - current price);
6  |  else if current price < f^-1 (current supply) then
7  |  |  if previous price < current price then
8  |  |  |  new price = step_2*(current price - previous price);
9  |  |  else
10 |  |  |  new price = step_1*(current price + previous price);
11 |  |  end
12 |  else
13 |  |  if previous price > current price then
14 |  |  |  new price = step_2*(current price - previous price);
15 |  |  else
16 |  |  |  new price = step_1*(current price + previous price);
17 |  |  end
18 |  end
19 end
```

Algorithm 3: Commodity Price Adjustment

Line 4: If this line is reached the current price does not satisfy the equilibrium condition on Line 1. This means that there is a gap between supply and demand in the market, and the price must be changed in order to close this gap. The algorithm starts by checking for equality between this and the current price. If there is equality, Line 2 was reached in the previous iteration, implying that the price at that point satisfied equilibrium conditions. This means that some other price change (commodity price for a different market or a change in the transportation prices) has caused imbalance in the market, and we can't look at the previous price and supply bid for this market to make a decision on how to adjust the current price. Therefore we look at the gap between the price given by the inverse demand function and the current price, and adjust the price with a fraction ($step_1$) of this gap. This occurs on line 5.

Line 6: If the price was changed in the last iteration, we change the price again according to whether or not the price was adjusted too much (line 10 and 16) or to little

(line 8 and 14) in the last iteration. If the previous price change was too small, indicated by the price being lower/higher than the inverse demand functions in both this and the previous iteration, the algorithm change it with a higher amount than it did last time. This is in order to speed up the search when we are far away from meeting supply (we see that the step size increases exponentially when we keep having under or over supply over several iterations). The increase in price change is given by the parameter $step_2$. If the price was changed too much in the previous iteration, i.e. we had over-supply and now have under-supply or vice versa, the new price is given a value somewhere between the current and previous price. How much the price is changed towards the previous price is given by the parameter $step_1$.

5.3.3 Handling Transportation Dependencies

In this section we tackle the problem with the suppliers choosing routes described in Section 4.3.1. The problem is that the suppliers does not have incentives to split their flows along several paths when it is necessary due to capacity restrictions or in order to minimize the transportation costs. The following two sections will explain the problems further and in the case of capacity restrictions suggest a procedure for modifying supplier bids in order to meet capacities.

Enforcing Link Capacities

Before the transportation prices can be adjusted, the transportation capacities must be taken into account: the suppliers are "blind" to the transportation capacities, so they will send all their commodities from supply node n to market node m along the cheapest path $p \subset N \times M$ in the network. We denote the link in p with the lowest capacity $l_c \in p$, and its capacity c. Let us assume the flow from n to m is higher than c. Seeing this, one is tempted to conclude that the price of flow over l_c should go up, i.e. the congestion rent of l_c wasn't high enough. But this is not necessarily true, because there might be another path $p' \neq p$ from n to m priced equal to p. If $l_c \notin p'$ and the unit flow cost over p' is the same as that of p, the supplier could just redirect the flow above capacity c over p' to avoid violating the capacity constraint for l_c. So there was no reason

to raise the price of l_c after all, we just needed to redirect some of the flow along another path. This situation is shown in Figure 5.1.

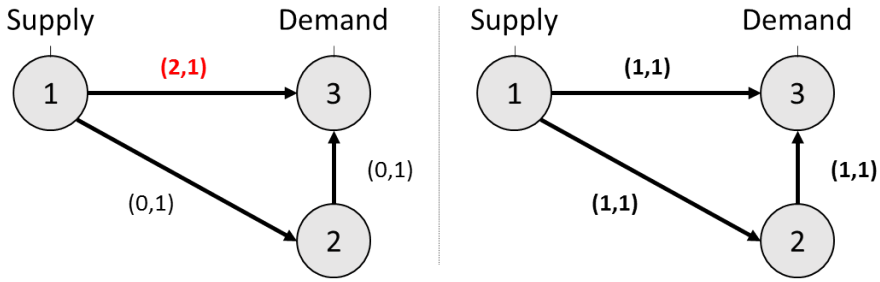

Figure 5.1: Two possible flow routing decisions. Touples (x,y) gives (flow,capacity) for each link. The network on the left shows a situation with flow bid on link (1,3) by the supplier in node 1 above capacity. If the unit price of flow along links (1,2) and (2,3) is equal to that of link (1,3), i.e. the two paths from node 1 to node 3 are equally priced, the supplier is not affected by redirecting some of it's flow along (1,2) and (2,3) rather than all through (1,3), thus satisfying flow capacities. This situation is shown on the right.

Our approach for accommodating this is to arrange a new round of bidding where the algorithm iteratively enforces capacity constraints over occurring capacity violations. Pseudo-code for this process i given in Algorithm 4.

The algorithm starts with a set of flow bids satisfying the commodity market clearing constraints. If there are flow capacity violations, one of the links with flow above capacity is chosen as link l. For all the suppliers contributing to the capacity violation of l, i.e. all the suppliers with positive flow through l, the algorithm lets them place new bids - now with the flow through l constrained by Equation (5.1).

$$f_l \leq \frac{f_s}{f_t} * c_l \tag{5.1}$$

The reasoning behind the value $\frac{f_s}{f_t} * c_l$ is two-fold: First, the total flow of the new bids will be restricted to the capacity of the link. Second, having the fraction of capacity assigned to each supplier equal their original fraction of the total flow in the capacity violating bid has the property that they get capacity proportionate to their previous bid, thus giving more to those who wanted most in the first place and vice verse. After getting the new bids from the suppliers, the dual price of the newly added constraint (5.1)

Data: Supplier bids without link capacity restrictions. Link capacities
Result: Supplier bids satisfying link capacity restrictions. Links needing a higher transportation price due to capacity restrictions

```
while there are capacity violations in bids do
    l = link with flow bids above capacity;
    foreach supplier s with positive flow on link l do
        f_s = flow bid on link l from supplier s;
        f_t = total flow on link l;
        c_l = capacity of link l;
        add f_l ≤ (f_s / f_t) * c_l to supplier's problem;
        let s place new bid;
        d = dual of added constraint;
        if d ≠ 0 then
            Mark link l as needing higher transportation price;
        end
    end
end
remove added capacity restrictions;
```

Algorithm 4: Capacity enforcing

is checked. As mentioned in Section 2.1.1, the dual price represents the value added by relaxing the constraint by one unit. In our case, the dual price of constraint (5.1) corresponds to the supplier's evaluation of being allowed to send one more unit of commodity flow through link l. If this dual is equal to 0 for all the suppliers we see that the suppliers have simply rerouted the flow to other paths in the network with no additional cost. This solves the problem in figure 5.1! In the other case, where the dual is non-zero for one or more of the suppliers, the algorithm can conclude that the price of link l should be higher. This is due to the fact that transportation prices in equilibrium correspond to the evaluation of the suppliers, i.e. when congested the price should have a congestion rent set high enough so that no supplier has incentive to rise the flow above capacity. A positive dual of a flow capacity constraint on the supplier means that it has added value if the constraint is relaxed, thereby the supplier has incentive to have a higher flow. Links needing a higher price are marked, so this can be taken into account in the procedure for transportation price adjustment.

This procedure for enforcing the link capacity restrictions on the suppliers will give a set of supplier bids were none of the capacity constraints are violated, but it does

not guarantee an optimal distribution of the link capacity, as constraint (5.1) is only a guess on what the equilibrium distribution is. How this affects the search convergence remains to be seen in Chapter 6.

Splitting Flow Based on Quadratic Costs

If we go back to Figure 5.1, we can construct an example where the flow from node 1 to node 3 needs to split along the two paths in order to minimize transportation cost. Lets assume the following cost parameters for the TSO:

- $C_{113}^{trans} = C_{213}^{trans} = 2$
- $C_{112}^{trans} = C_{123}^{trans} = C_{212}^{trans} = C_{223}^{trans} = 1$

Lets also assume that the transportation demand from node 1 to node 3 is 2, that there are no restricting capacity constraints, and the following transportation prices: $\alpha_{13} = 4$, $\alpha_{12} = \alpha_{23} = 2$. This means the routing of 2 units through link (1,3) is equal to that of routing the flow via node 2 from the suppliers perspective, and the solution to the suppliers problem will most likely send all the flow along one of these paths. If so, the marginal cost for the path flow will be 6 ($C_{113}^{trans} + 2 * C_{213}^{trans} = C_{112}^{trans} + 2 * C_{212}^{trans} + C_{123}^{trans} + 2 * C_{223}^{trans} = 6$). Using this routing to adjust the transportation prices, we would conclude that the price of the link(s) in the path used must go up, and the price of the link(s) not being used should go down. If, on the other hand, the flow was split equally along both the paths, their marginal costs would be 4, and by the Wardrop principle we know that this is the most cost efficient way to route the flows (all paths with positive flow are equally priced). This routing would render the current link prices equal to marginal cost and therefore yield equilibrium properties.

This example shows how the splitting of path flows can be crucial in order to find an equilibrium solution. As of now we have no mechanism in place to force this splitting of flow. Solving it perfectly will relate closely to solving the traffic equilibrium problem of Section 4.3.3 as we want to route flow according so the Wardrop principles holds. We solve the traffic equilibrium problem in the other two alternatives of the algorithm, but we believe there might be heuristic approaches to this as well. How the lack of enforcing

the Wardrop principles will affect the performance of Alternative 1 remains to be seen in Chapter 6.

5.3.4 Adjusting Transportation Prices

Like the commodity prices, the transportation prices are adjusted independently of each other. We also utilize the response to previous price changes in the same way we did for the commodity prices, i.e. if we need to take make a change in the same direction as we did in the previous iteration we take a larger step, and if we changed too much in the previous iteration we set the new price between the current and previous price.

The difference when comparing adjustment of commodity prices against that of the transportation prices lies in the necessary equilibrium conditions for the transportation prices, as described in Section 5.3.1. We also introduce a new step length parameter: $step_0$ is used to adjust a price in the cases where the price was not changed in the previous iteration (as opposed to $step_1$ being used on commodity prices in the same situation). $step_0$ is considerably smaller than $step_1$ and is used to avoid large price changes in said cases, as all commodity markets are cleared for each change of the transportation prices. Clearing commodity markets can be costly for big transportation price changes, and it also affects the transportation demand set by the suppliers. Therefore, in the cases of transportation prices not being changed in the previous iteration, we want to start with low changes in order to let the commodity markets adapt to the new change, and avoid sudden disrupts in transportation demand caused by large changes in the commodity markets. Hence the smaller step parameter $step_0$.

Pseudocode for adjusting the transportation price for a link is given in Algorithm 5.

Lines 1-8: The algorithm begins by increasing the price if the link was marked by Algorithm 4. This is done to reduce the suppliers transportation demand down to the capacity of the link. As Algorithm 4 has enforced the link capacity constraint, we don't know how much the price needs to be adjusted in order to get the transportation demand down to the capacity, so we do not have a clear target for how much the link price need to be incremented. The highest dual of the capacity constraints enforced on sup-

Data: Flow bid and price from this and previous iteration. Marginal cost and revenue functions MC and MR. Link capacity. Step length parameters $0 < step_0 < 1$, $0 < step_1 < 1$, $step_2 > 1$ and $0 < f_{scale} \leq 1$
Result: Adjusted flow price

1 **if** *link was marked as needing higher price by Algorithm 4* **then**
2 **if** *current price > previous price* **then**
3 new price = current price + $step_2 *$ (current price - previous price);
4 **else if** *current price < previous price* **then**
5 new price = current price + $step_1 *$ (previous price - current price);
6 **else**
7 new price = current price + $step_0 * f_{scale} * MC$(current flow);
8 **end**
9 **else if** *flow == capacity* **then**
10 **if** *MR(current price)* $\geq$ *MC(capacity)* **then**
11 new price = current price;
12 **else**
13 **if** *current price > previous price* **then**
14 new price = current price + $step_2 *$ (current price - previous price);
15 **else if** *current price < previous price* **then**
16 new price = current price + $step_1 *$ (previous price - current price);
17 **else**
18 new price = current price + $step_0 * (MC$(capacity) - current price);
19 **end**
20 **end**
21 **else**
22 **if** *current price == MC(current flow)* **then**
23 new price = current price;
24 **else if** *current price == previous price* **then**
25 new price = current price + $step_0 * (MC$(current flow) - current price);
26 **else if** *MC(current flow) > current price* ***and*** *current price > previous price* ***or*** *MC(current flow) < current price* ***and*** *current price < previous price* **then**
27 new price = current price + $step_2 *$ (current price - previous price);
28 **else**
29 new price = current price + $step_1 *$ (previous price - current price);
30 **end**
31 **end**

Algorithm 5: Transportation price adjustment

pliers by Algorithm 4 would give the needed increment, but in practice we have found this dual to be very volatile due to how Algorithm 4 distributes capacity between the suppliers is only a guess which often turn out wrong compared to the equilibrium solution. This has made us use the increment given on Line 7 in stead, which increases the

price with a fraction of the marginal cost of the link. The fraction is given by the initial step parameter $step_0$ and a new parameter f_{scale}. f_{scale} is representing the approximate distance between the current prices and the equilibrium prices, i.e. a value close to 1 when the distance is large (early part of the search) and closer to 0 as the search progresses. This makes the increment on Line 7 bigger when we need bigger changes, and smaller when we are closer. What values of f_{scale} are used is related to the slack tolerance scheme of the algorithm which is discussed in Section 5.6.3.

Line 9: Next we check for the case of the link flow being at capacity (and not marked as needing a higher price). If the marginal revenue is greater than or equal to the marginal cost, the price satisfy equilibrium conditions, so no changes are made. If marginal revenue (the current price) is less than the marginal cost, the price is increased.

Line 22: Last is the case of the flow being below capacity. In this case we want a price that makes marginal revenue equal to marginal cost. The search strategy becomes very similar to that of the commodity prices: a price is increased (or decreased) exponentially until it is too high (or low), in which case the algorithm start searching between the current and previous price.

Note that the algortithm does not look for prices below marginal cost when flows are zero. This means that the prices found for links with zero flow ($\alpha_{nm} = C^{trans}_{1nm}$) are just an upper bound on the equilibrium link price found when solving the MCP formulation of the model. The reason we do not search for prices below marginal cost is first of all that prices of links not evaluated high enough by suppliers to be used are not that interesting. Secondly finding these prices when we are still adjusting prices for links with positive flow adds unnecessary complexity. It is much easier to find them when we know the other true equilibrium prices, as we can lower them iteratively just until the point that the flow gets positive. Finally, prices below marginal costs are a consequence of the mathematical modelling used to find the equilibrium rather than representing the real world we are trying to model: in order to get congestion rents be equal to the suppliers willingness to pay, the transportation prices are set to the dual of the market clearing constraint (3.13). This has the consequence that prices can be lower

than marginal cost as well, but a better representation of the world would be to price links not being used at marginal cost, as this is the price suppliers would have to pay for it in order to use it, rather than the suppliers willingness to pay for that link. By this reasoning we have omitted trying to find suppliers willingness to pay for transportation links not being used, as we in these cases think the marginal cost is a better value for the transportation price.

With the adjustment of transportation prices presented, this concludes the description of Alternative 1.

5.4 Alternative 2: Let The TSO Handle Routing of Flows

Our second alternative uses the alternate formulations of the TSO and suppliers problems presented in Chapter 4 to handle transportation dependencies. This shifts the routing of the flows from the suppliers to the TSO, and changes the transportation prices the suppliers see from individual link prices to aggregated cost from supply nodes to demand nodes. In turn the suppliers puts bids on transportation in terms of units transported from supply nodes to demand nodes rather than along individual links.

The overall algorithm retains the same structure as that of Alternative 1 (see Algorithm 2), except for the mechanism for enforcing link capacities presented in Section 5.3.3 (Line 9) which becomes redundant and is removed from this version of the algorithm. All other differences compared to Alternative 1 lies in how the transportation prices are adjusted. Pseudocode for adjustment of the transportation prices is given in Algorithm 6.

Line 1: The TSO problem is solved with congestion rents added to the cost term of link flow and the capacity constraint removed as discussed in Section 4.3.6. Recall that if the transportation demand and the congestion rents are equal to that of an equilibrium solution, this formulation will give path and link flows that correspond to the equilibrium solution. Also, when using congestion rents as cost, the capacity restrictions become redundant for equilibrium transportation demand and congestion rents. We have dropped the capacity restrictions here as this makes for easier adjustment of the link prices. The congestion rents used are those that were determined by the previ-

Data: Supplier bids on SD transportation demand. Current and previous link prices. Current congestion rents
Result: New SD prices
1 Solve uncapacitated TSO problem given transportation demand and current congestion rents;
2 Adjust link prices with modified Algorithm 5;
3 **if** *link prices where adjusted* **then**
4 | set new congestion rents according to new link prices;
5 | set new SD prices equal to the shortest paths given new link prices;
6 **end**

Algorithm 6: Alternative 2 - SD price adjustment

ous iteration of the algorithm, and for the first iteration they are set to zero.

Line 2: The solution of the TSO problem gives the algorithm optimal link flow values given the current congestion rents and transportation demand. This provides a basis for adjusting the link prices in the same way we did in Alternative 1, so a modified Algorithm 5 is used here. The only changes are on Lines 1 and 7. On Line 1 capacity violations are found by the link flow being above capacity rather than being marked by Algorithm 4. On Line 7 and the initial step taken is scaled with the factor $(currentflow-capacity)/capacity$ rather than the f_{scale} parameter. By scaling with the new factor we are able to adjust the link price by how much the capacity restriction is violated, and thus making a better guess for the change needed to get the flow down towards capacity (more flow above capacity means a larger price increase is needed than for less flow above capacity).

Line 3: This line checks whether any of the link prices needed adjustment. If so we need to update the supply node to demand node (SD) transportation prices and congestion rents accordingly. If there was no need for adjustment we know that the current prices has equilibrium properties, and the algorithm is done.

Line 4: Reaching this line means the algorithm has adjusted the link prices. The congestion rents used in the TSO problem needs to be adjusted accordingly so that the TSO and suppliers place bids based on the same congestion rents in the next iteration. The congestion rents are set to the premium above marginal cost at capacity using the

formula

$$\gamma_{nm} = max(\alpha_{nm} - C_{1nm}^{trans} - 2C_{2nm}^{trans} CAP_{nm}^{trans}, 0)$$

Line 5: As the link prices have been adjusted the algorithm needs to update the SD prices to give the suppliers. As these prices are equal to the cheapest paths from supply nodes to demand nodes, we find solve a shortest path problem on the network using the current link prices as weights. The shortest paths are found using the Floyd-Warshall algorithm described in Section 2.2.3. We now have new SD prices to give to the suppliers and start a new iteration.

5.5 Alternative 3: Transportation Pricing Without Link Prices

Alternative 1 adjust link prices and presents them directly to the suppliers. Alternative 2 uses an underlying link price network to give a basis for the SD prices given to the suppliers. In this third and final alternative we bypass the link prices, and adjust SD prices directly based on the TSO solution and congestion rents. Other than the transportation pricing, this alternative is equal to that of Alternative 2. Algorithm 7 shows the procedure for adjusting the transportation prices.

```
   Data: Supplier bids on SD transportation demand. Current and previous SD
         prices and congestion rents
   Result: New SD prices
1  Solve uncapacitated TSO problem given transportation demand and congestion
   rents;
2  Adjust SD prices towards marginal costs of TSO problem including congestion
   rents;
3  Adjust congestion rents according to TSO solution;
4  foreach link l with adjusted congestion rent do
5  |  foreach SD with positive flow through l do
6  |  |  SD price += congestion rent adjustment of l;
7  |  end
8  end
```

Algorithm 7: Alternative 3 - SD price adjustment

Line 1: The TSO problem is solved in the same way as that of Alternative 2.

Line 2: On this line, the SD prices are adjusted towards the marginal costs of TSO

problem including congestion rents. In equilibrium the SD prices should be equal to this value. The SD marginal costs are found using the Floyd-Warshall algorithm with link marginal costs (still including congestion rents) given by the flow values of the TSO problem. Note that the duals of the transportation demand constraint contains the same marginal costs, but the Floyd-Warshall algorithm is used as the solver used in the implementation does not set dual values for SD pairs not having transportation demand above 0. The price adjustments are made using previous prices in the same way we adjusted towards marginal cost in the other Alternative 1 and 2.

Line 3: The congestion rents are adjusted according to whether or not they have equilibrium properties, i.e. up if the flow through a link is above capacity and towards zero if it's below. The congestion rents are adjusted using the previous congestion rents in the same fashion as the SD prices, and the initial step used when there was no change in the previous iteration is $step_0 * marginalcost * (flow - capacity)/capacity$, similar to how we adjusted link prices due to congestion in Alternative 2.

Line 6: On this line the SD prices are adjusted to reflect the adjustments made to the congestion rents. All SD routes with positive flow through a link with adjusted congestion rent has their SD price adjusted with the same amount the congestion rent was adjusted with. Note that this is an estimation of how changing a congestion rent affects SD prices: As a change in congestion rent can cause a change in how the TSO routes the flow, the portion of the link marginal cost decided by the quadratic cost parameter can also change. This means that in order to get the exact effect on SD prices occurring from a change in congestion rent we would have to solve the TSO problem again. This is omitted as solving the TSO problem is one of the most expensive operation throughout a search iteration, so we think it's more efficient to estimate the effect the congestion rent changes has on the SD prices and see how this in turn affects the transportation demand in the next iteration, rather than finding the exact change in SD price due to an adjustment in congestion rent in the current iteration. With the SD prices adjusted according to adjustments in congestion rents, we conclude the adjustment of SD prices for Alternative 3 and start a new iteration if adjustments to either the SD prices or the congestion rents where needed.

By not having a underlying link price layer, and rather change the SD prices directly from the TSO costs, Alternative 3 can drop a layer of complexity compared to Alternative 2 and only focus on what matters in terms of what affects supplier bids. On the other hand, Alternative 2 adjusts what drives the SD prices, which are the link prices, so what will turns out to be most effective remains to bee seen in Chapter 6.

5.6 Implementation

The alternative algorithms are implemented in C++ using Mosek Version 7 (ApS (2015)) to solve the different optimization problems (suppliers and TSO problems). The Mosek solver is set to use the previous solution as a starting point when solving the problems, and besides this it is set to its standard settings. The supplier problems are solved in parallel using the OpenMP interface (Chapman et al. (2007)).

5.6.1 Finding the path variables for alternate TSO problem

As the alternate TSO problem uses path variables, we have implemented a modified DFS algorithm (see Section 2.2.3) that finds all relevant paths in the network. The paths have two restrictions: They can not contain cycles (because we have positive link costs it's never economically efficient to send flow on a round-trip), and the total path costs can never be higher than the maximum possible revenue generated in the destination node quantified by the intercept of the demand slope. The recursive call of the modified DFS is shown in Algorithm 8. The algorithm finds all relevant paths between an origin and a destination node, so it is ran on all SD pairs to find all the relevant path variables for the TSO problem.

In Section 2.2.3 we noted that the number of possible acyclic paths in a network grows very quickly with increasing network sizes, and in the worst case is equal to the factorial of the number of nodes. With factorial increase in path variables, there is a risk of running out of physical memory on the computer running the algorithm, as was the case for us when we tried Algorithm 8 on a 30 node network. As the number of path variables affects the run-time of the TSO problem as well as memory usage, reducing

Data: Origin node o and destination node d. Current node c and path from origin until this current node p_{oc}. Demand intercept in destination node (B_d). Minimum cost of p_{oc}, $MinC_{oc}$
Result: Paths from current node to destination
foreach *neighbouring node n not visited and with* $MinC_{oc} + C_{1cn}^{trans} < B_d$ **do**
  if *neighbour == destination* **then**
    add path $p_{oc} + d$;
  else
    Recursive call with current path = $p_{oc} + n$;
  end
end

Algorithm 8: DFS call to find path variables

the number of path variables is crucial in order to solve the traffic equilibrium model in reasonable time. Heuristic procedures could be applied here to reduce the number of path variables. For instance setting a max number of links pr paths would help reduce the number of paths, but that has the risk of excluding paths used in the equilibrium solution. We think reformulating the TSO problem to a link-based model as discussed in Section 2.3.2, and thus remove the need for path variables, may be necessary for the algorithm to perform well on bigger networks.

5.6.2 Step Length Parameters

In Algorithm 3 and Algorithm 5 we introduced several step length parameters: $0 < step_0 < 1$, $0 < step_1 < 1$, $step_2 > 1$ and . We will here discuss possible values for these parameters, and how different values affect convergence for the algorithm.

To recap, $step_1$ is used by both transportation and commodity price adjustments when the new price needs to lie between the current and previous price. I.e. $step_1$ represents where on the line between the current and the previous price the algorithm moves to. When searching for a value known to be in a given interval (here the current and the previous price), and the interval is sorted (by increasing price) a binary search is the most effective. This consists of splitting the interval in two and checking in which half the true value lies in. (Readers unfamiliar with binary search may look to Cormen et al. (2009) for a detailed explanation). Inspired by this, we have chosen 0.5 as a value

for the $step_1$ parameter. We will not get exactly binary search behavior because other price changes can influence the direction we want to move, as well as the fact that the algorithm increases the step size when a step in one direction was not enough, but we still think that the going to the mid point when a value is assumed to be in a given interval is reasonable.

The $step_2$ parameter represents the step-length relative to the previous price-change when the algorithm decides to move in the same direction as it did the last time. A value equal to 1 gives the same step, while $step_2 > 1$ gives an exponential increase in the steps taken when the algorithm keeps being on the same side of the wanted price. The reasoning for having exponential increase in step length is to speed up the change when the current price is very far from the wanted price. Some caution is warranted as we risk jumping far above the wanted price when step sizes grow, but luckily this is where the $step_1$ balances things out by reducing the next step lengths exponentially. Still, as bigger steps increase the influence on actors decisions, thus increasing the effect on other prices, we should be careful with the exponential factor. A value of 1.5 (steps increase by 50 %) is considered fitting in this regard.

Another aspect is the relation between $step_1$ and $step_2$, because these risk cancelling each other out. Values .5 and 2 is an example of this, and the occurring looping effect is shown in figure 5.2. Lesson learned: avoid $step_1 = \frac{1}{step_2}$.

$step_0$ is only used when adjusting transportation prices, and represents a "small initial step" when we know what direction the price needs to go, but there was no change in the price in the previous iteration, e.g. if there was no change in the price in the previous iteration (indicating it had equilibrium properties) but it now needs to change towards a higher marginal cost. In these situations influences from other price changes tends to be high, as the price had equilibrium properties in the previous iteration some other price change has created the need to change the price. This causes the algorithm to not have good assumptions on what properties the link will have in equilibrium (e.g. if it will have positive congestion rent or have flow at all), and therefore we want $step_0$ to be small in order to let the other prices stabilize before changes on the current link grow large. Based on this we have set the value of $step_0$ equal to 0.01.

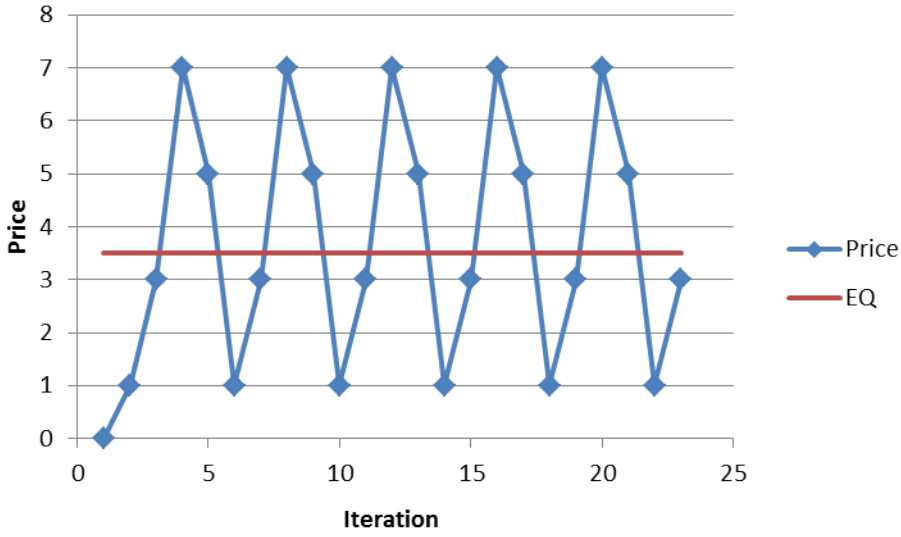

Figure 5.2: A search with $step_1 = 0.5$ and $step_2 = 2$. We see that we keep moving between same set of prices, not getting closer to the equilibrium price.

When taking the influence prices has on each other into account, we realize that different price changes can cancel each other out. In this case we risk that the algorithm can start looping between a fixed set of prices and simply get stuck (similar to the situation in Figure 5.2). Adding a random or pseudo random element to the step parameters could accommodate this, and should be considered in cases of looping behaviour. At this point we have not taken such concerns, but it might prove necessary in further development.

5.6.3 Slack tolerance

As noted in Section 5.3.1 the algorithm allows some slack when evaluating the equilibrium properties of the commodity and transportation prices.

For each commodity market n we allow a slack between the price suggested to the suppliers π_n and the price given by the inverse demand function on supply bids. We denote ϵ^{com} as the parameter for allowed slack. The algorithm sees the supplied and demanded quantities as equal if inequality (5.2) holds. Note that s_{ni} is found by the

suppliers problem given the price π_n.

$$-\epsilon^{com} \leq \pi_n - B_n + A_n \sum_{i \in P} s_{ni} \leq \epsilon^{com} \tag{5.2}$$

In the same way the algorithm allows slack ϵ^{TSO}_{price} when comparing the transportation price α_{nm} to the marginal cost of the TSO. Equations (5.3) and (5.4) shows the conditions for the transportation prices being considered equal to TSO marginal cost for the link and SD level respectively. δ^{demand}_{sd} denotes the dual of the demand constraint (4.13).

$$-\epsilon^{TSO}_{price} \leq \alpha_{nm} - (C^{trans}_1 + 2C^{trans}_2 g_{nm}) \leq \epsilon^{TSO}_{price} \tag{5.3}$$

$$-\epsilon^{TSO}_{price} \leq \alpha_{sd} - \delta^{demand}_{sd} \leq \epsilon^{TSO}_{price} \tag{5.4}$$

When comparing flow to capacity the algorithm allows slack as a portion of the capacity, i.e. flow is considered equal to capacity if it is within ϵ^{TSO}_{cap} percent of the capacity. The equality check is given in Equation (5.5). Note that this equality check differs from that of the commodity and transportation prices in that the allowed slack (+- $\epsilon^{TSO}_{cap} * CAP^{trans}_{nm}$) is relative to the capacity value, while in the price equality checks the allowed slack is absolute. The use of relative slack values has the allowed slack varying with the link capacity, which we think is fitting from a modelling perspective as you want less slack on capacity restrictions on links with small capacity than on links with a bigger capacity.

$$-\epsilon^{CAP}_{rel} * CAP^{trans}_{nm} \leq \frac{g_{nm} - CAP^{trans}_{nm}}{CAP^{trans}_{nm}} \leq \epsilon^{CAP}_{rel} * CAP^{trans}_{nm} \tag{5.5}$$

$$-\epsilon^{CAP}_{abs} \leq g_{nm} - CAP^{trans}_{nm}) \leq \epsilon^{CAP}_{abs} \tag{5.6}$$

Both relative capacity slack (5.5) and absolute capacity slack (5.6) were tested on all three alternatives of the algorithm, and had insignificant effects on algorithm performance, except for Alternative 1 where the algorithm performed considerably worse for relative capacity slack than with absolute capacity slack, and in some cases models solved with absolute capacity slack where not solved using relative slack. Why this

had such an impact on performance for Alternative 1 is a small mystery to the authors, but we suspect that negative effects of the transportation dependencies not being handled is somehow enforced by use of relative capacity slack. The difference in how the alternatives adjusts transportation prices up in cases of capacity violations might also explain the difference in performance. Either way we decided to use absolute capacity slack for Alternative 1 and relative capacity slack on the others on the basis of algorithm performance.

The use of relative slack tolerance were considered on prices as well. Using relative slack on prices allows more slack on more expensive items and less on cheaper items. In the case of transportation prices, relative slack would allow lower slack on links with low marginal costs (indicating low flow) and higher slack tolerance on prices with high marginal costs (indicating high flow). This is not wanted as we want higher accuracy on the prices of "active" links that in turn has a bigger influence on both supplier and TSO behaviour, rather than on the links not being of interest to suppliers. Thus we decided on using absolute slack on the prices.

The algorithm starts with a set of relatively high slack tolerance parameters ϵ^{com}, ϵ^{TSO}_{price} and ϵ^{TSO}_{cap}. After prices with equilibrium properties within the given slack tolerance is found, the procedure decreases the slack parameters and restarts with the previous solution as a starting solution. This is repeated until the solution found are within wanted slack levels.

This is done mainly for solution time considerations. The suppliers problems requires a substantial amount of the computational efforts in each iteration. Limiting the number of solver calls should therefore be of high priority when trying to reduce the time required for the algorithm to find an equilibrium solution. Using small commodity price slack tolerance means the algorithm will use more iteration to clear commodity markets for each change of transportation prices than if bigger commodity slack tolerance were used. Using large commodity slack parameters requires less iterations to clear commodity markets, but gives less accurate commodity prices and in turn less accurate supplier transportation demands given the current transportation prices. Less

accurate transportation demand reduces the chance of adjusting the TSO prices in the right direction. So we have a trade-off between reducing the time used for clearing commodity markets per transportation price adjustment, and increasing the chance of the transportation prices being adjusted correctly leading to faster convergence towards equilibrium prices. To balance this trade-off the commodity price slack tolerance should represent the current distance the TSO prices is from equilibrium, and we think that starting with overall high slack tolerance and iteratively reducing them is the best way of achieving this balance, and thus increasing performance by reducing the number of solver calls.

By the same reasoning the slack tolerance parameters ϵ^{com}, ϵ^{TSO}_{price} and ϵ^{CAP}_{rel} or ϵ^{CAP}_{abs} should be set to initial values that are balanced in terms of their influence on the equilibrium properties of the prices. We have decided on the following initial slack parameter values:

Parameter	**Value**
ϵ^{com}	10% of average midpoint of demand curves for all commodity markets
ϵ^{TSO}_{price}	10% of average TSO marginal cost with flow at half capacity
ϵ^{CAP}_{abs}	10% of average link capacity
ϵ^{CAP}_{rel}	0.1

When the algorithm has found prices with equilibrium properties given the current slack parameters, we divide all slack parameters by 10 and restart until until prices with equilibrium properties within desired slack levels are found. Desired solution slack levels are free to choose by the user of the algorithm, and depends on wanted accuracy. The limit for possible slack levels are dependent on the precision of the Mosek solver, as well as the algorithms ability to converge.

With the slack parameter scheme presented, we can get back to the f_{scale} parameter presented in Section 5.3.4. Recall that this parameter is used to scale the step taken on Line 7 of Algorithm 5, i.e. when a link price needs to be incremented due to congestion and there was no change in the price in the previous iteration. We argued that the value of f_{scale} should represent an approximation of the distance between the current

prices and the equilibrium prices, and with the slack scheme we should now have a good approximation of this using by knowing what slack levels we have found equilibrium prices for. f_{scale} starts with a value of 1, and then as slack levels are decremented we decrement f_{scale} by the same factor. By doing this we decrease the steps taken with f_{scale} as we get closer to the equilibrium prices, thus adapting the search to continuously get closer to the equilibrium values. Note that f_{scale} is only used by Alternative 1, and that Alternative 2 and 3 uses the amount of flow above capacity to scale the step length instead.

Chapter 6

Test Run Results and Discussion

In this chapter we will test the solution algorithm on nine different data sets. Section 6.1 will describe the data sets used. The testing environment and algorithm parametrisation is given in Section 6.2. The test results in terms of speed and accuracy of the solutions are presented in Section 6.3, before Section 6.4 discusses our findings and analyze some interesting test results.

6.1 About the Data sets

We have ran tests on a total of nine data sets. The first three sets have a simple structure, and are used to show the inner workings of the algorithm and test specific problems described in Chapter 5. The last six data sets were created by modelling three different regions of the global natural gas market. Two versions of each region are created, one version with congested links in the equilibrium solution and one where the flow through all links are below capacity. As mentioned in Section 3.2.2 the transportation capacity restrictions could be modeled by an asymptotic cost curve, so we think it is valuable to check the impact the capacity restrictions have on algorithm performance. While the first three data sets are created in order to highlight specific aspects of algorithm performance, the last six data sets are designed to resemble typical instances in energy market equilibrium modelling.

All data sets are based on real-world data taken from the global natural gas market (see Egging et al. (2010) for a description of the data source). This done to ensure that the relative size between different cost and capacity parameters resemble that of instances used in practical applications. The prices are in US dollars per thousand cubic meter of natural gas (USD/kcbm) and the quantities are in thousand cubic meters (kcbm) of natural gas. The commodity prices (π_n) at equilibrium tend to be in an order of magnitude of 100 to 400 USD, while transportation equilibrium prices (α_{nm}) are in an order of magnitude of 20 trough 300 USD. Equilibrium quantities (both sales s_{ni}, production q_{ni} and flows g_{nm}) are between 500 and 150 000 kcbm.

For all the data sets, each supplier only have supply in one node. As noted in Section 4.3.2, suppliers with production in several node might have multiple solutions for the same set of prices. This could keep the algorithm from converging, as there exists a transportation dependency not handled by the algorithm, yielding the following **suggestion for further work:** Investigate the effects suppliers having production in more than on node has on algorithm performance.

Data Set 1 is a three-node network with a single supplier. The topology of Data Set 1 is shown in Figure 6.1. All nodes have a fuel demand and the supplier is located in node 1. When the markets clear there is no congestion on the links. Data Set 1 is for an overall comparison of the different algorithm alternatives.

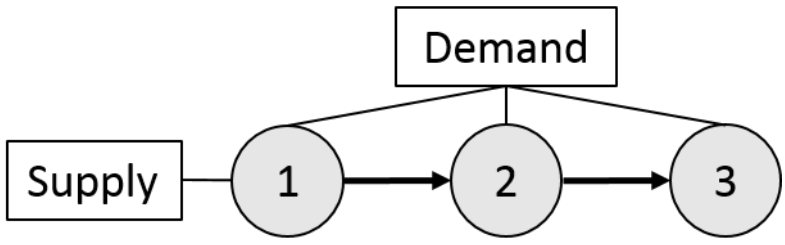

Figure 6.1: The topology of Data Set 1.

Data Set 2 is a three-node network with two suppliers in node 1 and 2 delivering fuel to an attractive market in node 3. The network topology is shown in Figure 6.2. The transportation link between node 1 and node 2 has limited capacity in the equilibrium

solution and supplier 1 will choose to route some of his sales through node 2. We will examine how the three algorithm alternatives handles splitting of flows due to capacity constraints in the network, continuing the discussion we started in Section 5.3.3.

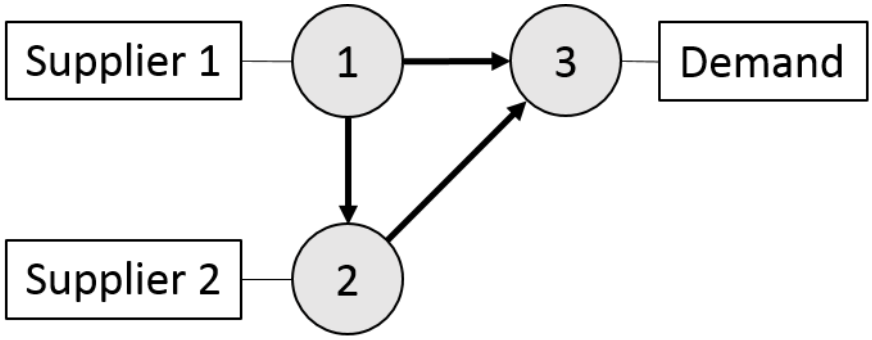

Figure 6.2: The topology of data sets 2 and 3. Supplier 1 will split his flow to node 3 and send fuel both directly through link (1,3) and by node 2 to maximize his profit.

Data Set 3 has the same topology as Data Set 2, but none of the links are constrained by capacity restrictions in the equilibrium solution. In equilibrium the link between node 1 and node 2 is expensive, so supplier 1 will choose to route some of his fuel by node 2. We will use this data set to examine how the three algorithm alternatives handle splitting of flows due to link costs, also discussed in Section 5.3.3.

Data Set 4 is a five-node network based on countries in central Europe. All nodes have suppliers and consumers acting in them. There is a total number of 10 links in the network.

Data Set 5 has the same topology as Data Set 4, but without congested links in the equilibrium solution. This is achieved by increasing the quadratic cost component C_{2nm}^{trans} until the link is no longer constrained by the capacity restriction.

Data Set 6 is an eight-node network based on South America. It contains a total of 17 links, and one supplier is present in every node.

Data Set 7 has the same topology as Data Set 6, but no links are congested in the equilibrium solution.

Data Set 8 is a 13-node network based on northern America. 12 of the nodes have a production facility each belonging to a different supplier. Data Set 8 has 43 links, 7 of

which are congested in the equilibrium solution.

Data Set 9 has the same topology as Data Set 8, but without congested links in the equilibrium solution.

6.2 Computational Specifications

The tests were done on a Lenovo Yoga 2 Pro laptop with a dual core Intel i7-4500U CPU. All data sets are tested using all three alternatives of our algorithm and the solver PATH. PATH was used on the MCP formulation of the model stated in Section 3.3 programmed in the GAMS modeling system (Rosenthal (2015)). We did not configure PATH any differently from the default settings as its user guide (Ferris and Munson (1998)) states it will adapt to the problem size.

The algorithm terminates when when prices with equilibrium properties with the following slack parameters is found:

- $\epsilon^{com} = \epsilon^{TSO}_{price} = 10^{-3}$
- $\epsilon^{CAP}_{abs} = 10^{-2}$
- $\epsilon^{CAP}_{rel} = 10^{-7}$

Recall that the price slack parameters ϵ^{com} and ϵ^{TSO}_{price} sets the allowed difference between commodity prices and the inverse demand curve given supplied quantities and the difference between transportation prices and TSO marginal costs, respectively. With prices being in unit US dollars per 1000 cubic meters gas, the price slack tolerance parameters represent the "marginal incentive" to change a price for each actor in the model in US dollars per 1000 cubic meters gas. We think that said incentives below 10^{-3} are small enough to ignore from a modeling standpoint, and thus a fitting price slack parameter limit. With link capacities averaging around 10 000 and the largest being in the 100 000 range, the relative capacity slack limit $\epsilon^{CAP}_{rel} = 10^{-7}$ is approximately as strict as the absolute capacity slack limit $\epsilon^{CAP}_{abs} = 10^{-2}$ for the links with high capacity. For the lower capacity links, the relative slack parameter will be stricter than the absolute, but

in both cases we think that the limits are fitting as capacity violations of 10^{-5}% of capacity, or 10^{-2} kilo cubic meters of gas are small enough to be considered insignificant from a modeling standpoint.

Note that the slack parameters only limit the incentives to change and not the error on solution prices and quantities. With the slack parameter values stated above we aim for prices with errors less than 0.01 (1 cent). The PATH solver with its default settings has a convergence tolerance of 10^{-6}. The convergence tolerance parameter sets the maximum value for one of the sides of the complementarity conditions MCP formulation to be considered at 0. If we look at the MCP formulation of our model stated in Section 3.3 we see that the convergence tolerance of the PATH solver is comparable to the value of the slack parameters in our algorithms, and as such the PATH solver has a stricter slack tolerance than our algorithms. We argue that we can set higher slack parameter values as we can take what each slack parameter restrict in the solving of the model into account and set these to values we consider fitting from a modelling standpoint. The fact that the PATH solver has stricter slack tolerance also lets us compare the solutions using the difference between those of our solution and that of path as an error term. At the end of the next section we will compare the solutions given by our algorithms to the solutions of the PATH solver.

The step length parameters have the values stated in Section 5.6.2, i.e. $step_0 = 0.01$, $step_1 = 0.5$, $step_2 = 1.5$ and f_{scale} starts at 1 and reduced to 10% of its value when slack parameters are reduced (the slack parameters are also reduced by a factor of 10).

6.3 Results

An overview of our results in terms of solution times and number of iterations is shown in Table 6.1. The columns describe the following properties of the search:

- **Data Set:** Which of the nine data sets were solved.
- **Alternative:** Which alternative of the algorithm to be used. The three alternatives are described in Chapter 5.

- **TSO Iter.:** The number of times the transportation prices have been adjusted. For alternatives 2 and 3 this corresponds to the number of times the TSO Problem has been solved.

- **Commodity Iter.:** The total number of times the commodity prices have been adjusted. Note that the commodity prices are adjusted to clear the commodity markets *for each adjustment of the transportation prices* and that this number is the sum of all commodity price adjustments throughout the search.

- **Solution Time:** The duration of the run, in seconds. This is the time used by the algorithm to find equilibrium prices, and excludes time spent loading the data instances and setting up the algorithm e.g. the DFS run for finding path variables (Algorithm 8). The solution times are in wall clock time, not CPU time.

We see that Alternative 1 only manages to solve three of the nine instances. It is significantly faster than the other two alternatives on Data Set 2, which was designed to test the procedure described in Section 5.3.3 that splits flow due to capacity restrictions. The fact that Alternative 1 outperformed the other two alternatives on this instance indicates that Alternative 1 handles the link capacity restrictions well. That Alternative 1 failed on Data Set 3 was no surprise, as this data set is designed to test how the algorithm handles splitting of flow due to cost considerations described in Section 5.3.3. As Alternative 1 does nothing to handle this splitting of flow, it is not surprising that it fails to converge to find equilibrium prices in this instance. The reason Alternative 1 fails on the other instances (except Data Set 7) could be due to the same need to split flow based on cost consideration as that of Data Set 3.

Alternative 2 solves the model for all instances. We see that it takes 26404 transportation price adjustments for it to solve Data Set 2, roughly 500 times more adjustments than needed for Data Set 8, which is surprising considering Data Set 2 only has three links, while Data Set 8 has 43. In general Alternative 2 spends more time searching for solutions in instances with congested links than in those without.

Alternative 3 solves all the data sets. It is slower than Alternative 2 on all sets except Data Set 2 and Data Set 6.

Data set	Alternative	TSO iter.	Commodity iter.	Solution Time
Set 1: Simple supplier	1	40	3	0.029
Set 2: Capacity split	1	502	245	0.797
Set 3: Cost split	1			
Set 4: 5-node congested	1			
Set 5: 5-node uncongested	1			
Set 6: 8-node congested	1			
Set 7: 8-node uncongested	1	66	204	1.218
Set 8: 13-node congested	1			
Set 9: 13-node uncongested	1			
Set 1: Simple supplier	2	40	3	0.035
Set 2: Capacity split	2	26404	137	18.101
Set 3: Cost split	2	47	19	0.040
Set 4: 5-node congested	2	176	427	0.945
Set 5: 5-node uncongested	2	77	70	0.261
Set 6: 8-node congested	2	1230	10541	35.525
Set 7: 8-node uncongested	2	79	251	1.153
Set 8: 13-node congested	2	178	829	40.930
Set 9: 13-node uncongested	2	121	1172	30.087
Set 1: Simple supplier	3	49	3	0.039
Set 2: Capacity split	3	25521	164	21.199
Set 3: Cost split	3	51	18	0.056
Set 4: 5-node congested	3	182	226	0.759
Set 5: 5-node uncongested	3	90	77	0.317
Set 6: 8-node congested	3	215	291	2.723
Set 7: 8-node uncongested	3	113	163	1.277
Set 8: 13-node congested	3	161	374	44.833
Set 9: 13-node uncongested	3	221	1135	60.574

Table 6.1: The results from the test runs. Empty entries means the algorithms did not find a solution.

There is a big span in the average number of commodity iterations per TSO iteration. The variance is very large, and Alternative 2 spans from 0.005 commodity iterations per TSO iteration on Data Set 2 to 9.69 commodity iterations per TSO iteration on Data Set 9. We will get back to this when we discuss the relationship between commodity iterations and other data set property in Section 6.4.5.

Table 6.2 shows the solution times for the PATH algorithm for all instances. The solution times are the time spent solving the model (RESOURCE USAGE in the GAMS solve summary) and excludes the time spent setting up the problem, in the same way as that of the solution times in Table 6.1.

Data set	PATH Solution Time
Set 1: Simple supplier	0.016
Set 2: Capacity split	0.031
Set 3: Cost split	0.031
Set 4: 5-node congested	0.140
Set 5: 5-node uncongested	0.188
Set 6: 8-node congested	0.156
Set 7: 8-node uncongested	0.141
Set 8: 13-node congested	2.907
Set 9: 13-node uncongested	1.296

Table 6.2: The solving times of the PATH solver for all data instances. All times are in seconds.

We see that PATH solves the models quicker than any of the alternative algorithms for all data sets. The smallest time difference between PATH and the best of the alternative algorithms is Alternative 2 using 29 % more time than PATH on data set 3. The biggest difference is on data set 8 where Alternative 2 uses 40 times longer than PATH to find a solution.

Table 6.3 show the maximum and average differences between the solutions found by our algorithms and the solutions from the solver PATH. In general, we see that the algorithms has converged towards the equilibrium solutions and that the errors correspond to the slack parameter values used.

Error		Commodity Prices	Link Prices	Sales	Link Flow
Absolute	Average	0.0019	0.0026	0.0091	0.0070
	Max	0.0049	0.0059	0.2975	0.4645
Relative	Average	0.0011%	0.0197%	0.0016%	0.0011%
	Max	0.0041%	0.2823%	0.1433%	0.3640%

Table 6.3: The average and maximum deviations from PATH solutions among all data sets and all algorithm alternatives. The prices are in USD/kcbm and the quantities in 1000 cubic meters. Relative values are relative to PATH variable values.

The absolute errors on the prices are higher than the price slack parameters (10^{-3}), but they are still within 1 cent of the PATH solution. By comparing the errors on transportation (link) prices to that of commodity prices we see that the transportation prices have higher errors, especially for the relative errors. This can be explained by the commodity prices being of higher values than the link prices, and since the slack parameters

used on both prices are the same the relative error becomes bigger for the smaller prices (the link prices).

6.4 Test Run Discussion

From the results above there are several ways to analyze the algorithms performance. We will first take a look at the search mechanisms in Section 6.4.1. We then examine each algorithm alternative individually in Section 6.4.2 trough 6.4.4. In Section 6.4.5 we take a closer look at how the algorithm and data set properties affects the number of commodity iterations. Section 6.4.6 compares the solving of congested versus uncongested networks, and Section 6.4.7 discusses the algorithms sensitivity to different search parameters. The discussion is closed by Section 6.4.8 which compares our algorithm to PATH and does some analysis on scalability and run time. Finally, we do a ranking of the three algorithm alternatives in Section 6.4.9.

Some of the plots shown in this Section have had their values scaled in order to show the trends rather than the absolute development. A variable x is scaled according to the formula

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

which means that the highest value in the series will be scaled to 1 and the lowest value in the series will be scaled to 0.

6.4.1 A closer look at the Search Strategy

In Data Set 1 we see the basic search mechanic behaving as we expect. Figure 6.3 shows some statistics from the run of Data Set 1 using Alternative 1: the commodity price, the consumed quantity and the transportation price on the link from node 1 to node 2. Note that Iteration 0 represents the initial values of the commodity price and the transportation price. The transportation price grows exponentially during the first 10 iterations. Simultaneously the quantity sold in node 2 drops exponentially. The increased transportation price gives supplier 1 incentives to sell less in the market. After the transportation price peaks in iteration 10 it is adjusted down, and the supplier responds by

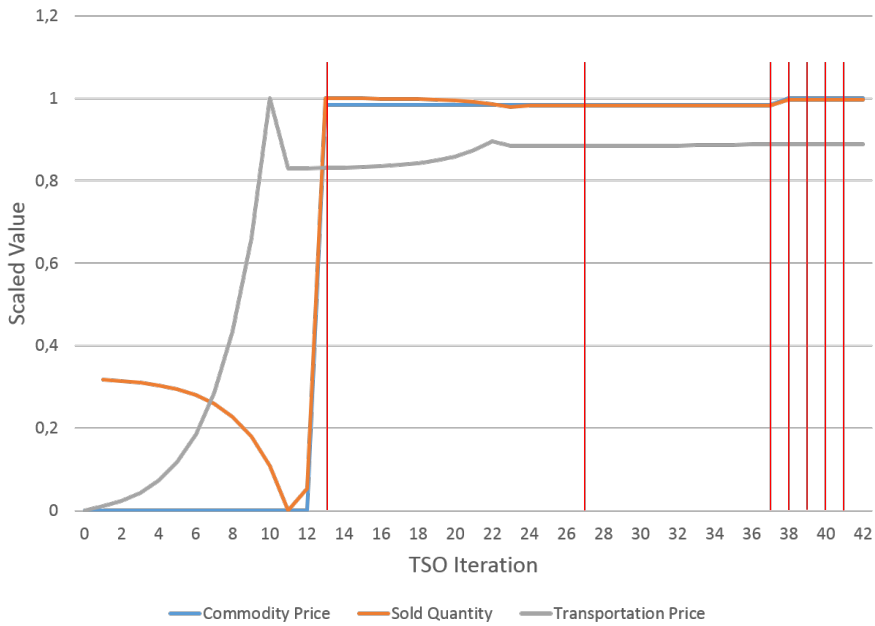

Figure 6.3: Scaled commodity price, transportation price and consumed quantity from the run of data set 1, Alternative 2. The slack level decrements are marked with vertical red lines.

slightly increasing his sales. In iteration 12 the commodity price is increased, which gives the supplier an incentive to sell more. In iteration 13, the algorithm decreases the slack level for the first time, and the effect of the transportation price slack is clearly visible here: even if the transportation price of link (1,2) is not at its equilibrium value, it is sufficiently close for the search to finish given initial slack values. After iteration 13 (and the first slack level decrement) the transportation price slowly increases, reaching a local maximum in iteration 23, before it converges to its equilibrium price.

Just after the third slack decrement in Figure 6.3 the commodity price is increased slightly, resulting in a corresponding increase in the suppliers sales. The algorithm only spends one iteration to correct the prices, and the following slack decrements happen on every iteration until the search is finished. This is a result of the algorithm design: by finding an equilibrium solution with high slack levels, the search converges faster with lower slack level.

Alternatives 2 and 3 show almost the exact same development as seen in Figure 6.3. This leads us to the conclusion that the search strategy is fundamentally the same in all alternatives, and that it achieves expected performance: if a price is far away from its equilibrium value it is adjusted exponentially. If the algorithm overshoots the equilibrium price, the binary search-inspired mechanism quickly finds the equilibrium solution. The differences between the three alternatives is more apparent with increased complexity in the network topology, multiple paths between source and destination, and positive congestion rents.

From the test runs we can also see a consequence of adjusting market prices individually. In some data sets we observe that a price is wrong for several iterations which makes the search converge slower.

Consider Figure 6.4 which plots the commodity price in node 1 and the transportation prices on links (1,2) and (1,5) from Data Set 6, Alternative 2. Initially, the commodity price is flat before it is adjusted upwards. The transportation prices are adjusted smoothly up, but past their equilibrium values. Only in iteration 12 do they start to fall, converging slowly to their equilibrium values. The reason why the transportation prices was increased to far is because the commodity price was too low. When the market in node 1 was less attractive, the supplier in node 1 produced more for other markets, and less for the market in node 1. This made the outgoing links congested, which in turn raised the transportation price. Only when the commodity price in node 1 is increased does the supplier produce more for the market in node 1, and less for other markets. This shift in sales reduces the link congestion, and the transportation price is decreased.

The reason why the commodity price was not adjusted until iteration 7 is because the of slack values. One solution to this is to start the search with lower slack values. However, we can be smarter. The ideal situation for the prices shown in Figure 6.4 is if the commodity price for node 1 had seen a smooth rising price in the first iterations simultaneously to the links by realising that the market would become more attractive when increasing the outgoing link prices. As a consequence the transportation prices would not have been increased to a value above the equilibrium price, and the search

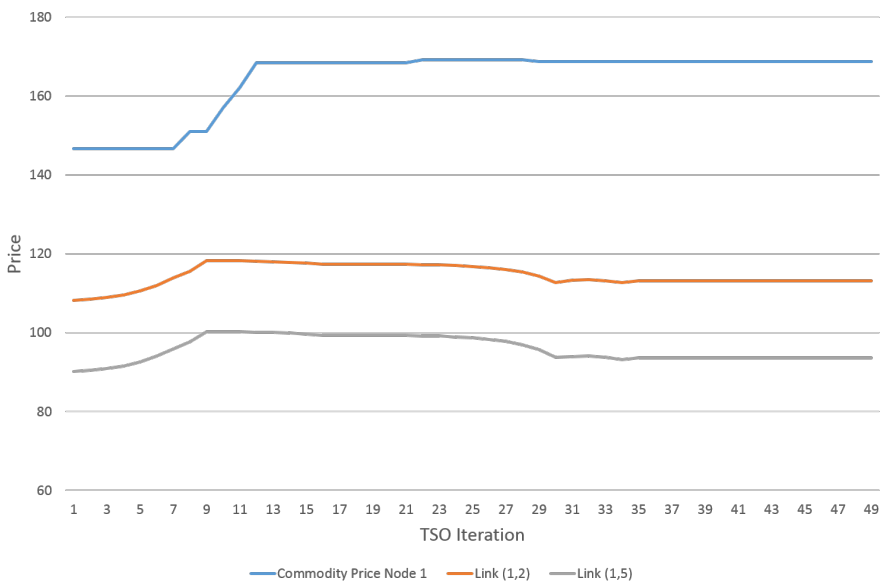

Figure 6.4: The commodity price in node 1 and the price on links (1,2) and (1,5) for Data Set 6, Alternative 2.

might have converged faster.

The slowed convergence seen by adjusting prices individually leads to the following **suggestion for further work:** Quantify the correlation between commodity prices and transportation prices to create a more holistic search strategy.

6.4.2 A closer look at Alternative 1

Recall that Alternative 1 of our algorithm has the suppliers routing their flow. We identified two problems with this in Section 4.3.1. One of the problems is related to suppliers having to reroute flow due to capacity constraints, which occurs in Data Set 2 where the capacity restrictions are forcing the supplier to split his flow along two paths. The mechanism for enforcing the capacity constraints presented in Section 5.3.3 works as intended: when Alternative 1 searches for equilibrium prices the suppliers reroutes their flow along the two paths, and pulls the congestion rents toward the equilibrium values that removes incentives for the suppliers to break the capacity restrictions. This is reflected by the increase in the prices for the congested links (1,3) and (2,3) (see Fig-

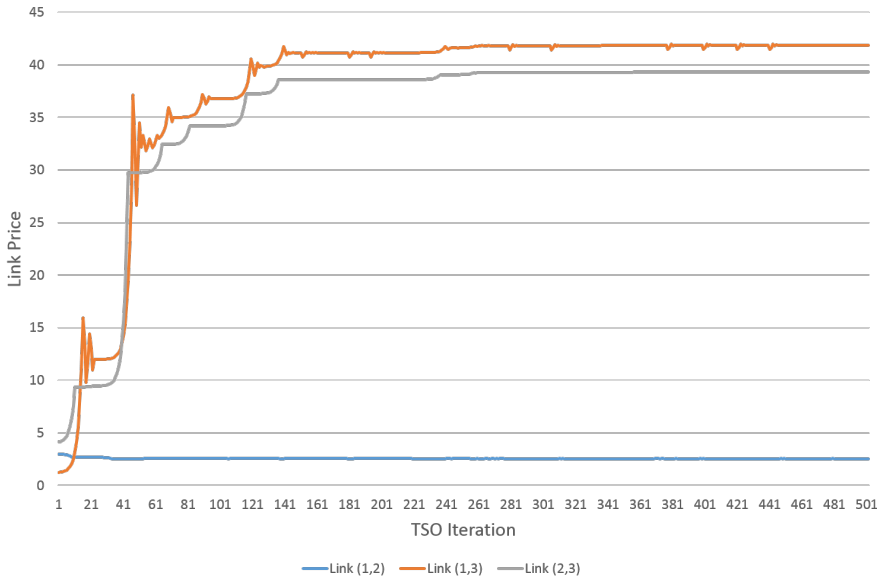

Figure 6.5: The link price development of Data Set 2, Alternative 1. Note that the prices of congested links (1,3) and (2,3) increase to remove incentives for the suppliers to break the capacity restrictions.

ure 6.2 for the network topology) shown in Figure 6.5.

Alternative 1 is unable to solve Data Set 3, where splitting the flow due to cost considerations is crucial in the equilibrium solution. As discussed in Section 5.3.3 Alternative 1 has no mechanism to enforce this type of rerouting, and the suppliers will most likely route all their flow along one of the cheapest paths rather than splitting along several paths even if that is more cost efficient for the TSO. This is exactly what happens when Alternative 1 is used on Data Set 3. Figure 6.6 shows the flow routed through the two different paths from node 1 to node 3 by supplier 1 throughout the search. In equilibrium supplier 1 has to split its flow along both paths from node 1 to node 3 in order to keep the transportation costs down. As shown in Figure 6.6 supplier 1 will alternate between sending all its flow along the two paths. This alternation of flow in turn has the marginal costs of the links alternating between a higher value than in equilibrium when the links are used, and a lower value than in equilibrium when the links are not being used. Since the link prices will be adjusted towards the TSO marginal costs, they

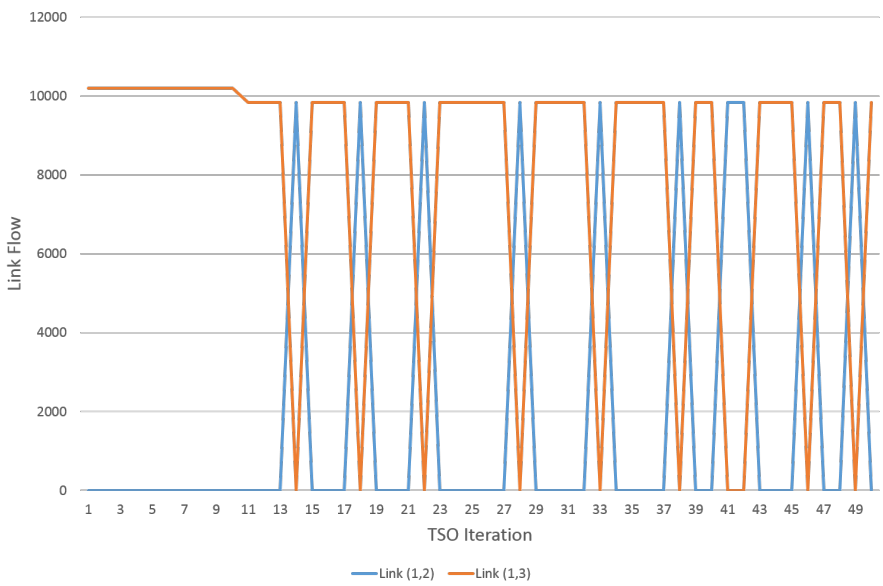

Figure 6.6: The decisions of supplier 1 in Data Set 3, Alternative 1. Notice how he keeps changing between the two paths depending on which is cheaper.

will alternate between small adjustments up and down and not converge towards their equilibrium values. Alternative 1 keeps on doing this for the entire search and the algorithm fails.

The alternating flow routing by the suppliers that caused Alternative 1 to fail on Data Set 3 is apparent in all other data sets for which Alternative 1 does not find a solution. An example is shown in Figure 6.7 which shows the routing decisions for supplier 1 in Data Set 4. This shows that a there is a need to split flows along several paths due to cost consideration in all of these data sets, and that a mechanism to enforce this must be established in order to make Alternative 1 useful in practical applications. **Suggestion for further work:** Develop a mechanism for Alternative 1 to split flow due to TSO cost considerations

To gain some insight in when Alternative 1 does work we should also analyze how Alternative 1 behaves on Data Set 7. Three link prices from the run of Data Set 7, Alternative 1 are shown in Figure 6.8. Notice in particular how nicely the price of all links falls on the 8 first iterations. This is close to the optimal development of link prices given our

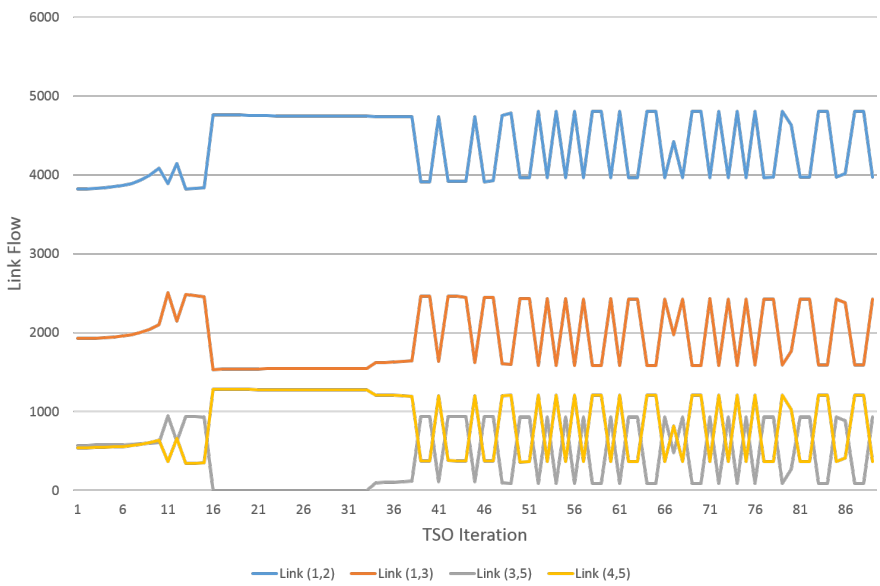

Figure 6.7: Link flow decisions for Supplier 1 for the first TSO iterations of Alternative 1 on Data Set 4. Notice the alternating flow decisions, similar to that of Figure 6.6.

algorithm. The link prices are set higher than the equilibrium prices at the start of the search, and take steps of increasing size towards the equilibrium price. The stop in iteration 8 represents the first slack decrement in the run. Later in the run, a correction is seen in iteration 19 for all links. This correction is unrelated to a slack decrement, but happens because the link prices pass below marginal cost, and the algorithm adjusts the link prices up.

It is a bit surprising that Alternative 1 manages to solve an uncongested data set like Data Set 7. If we know up front that all links are uncongested in the equilibrium solution we know for a fact that no supplier will split his flow due to capacity constraints. If a supplier choose to split his flow he will do so because of link cost: the exact situation we now know Alternative 1 is not able to solve. This results in the conclusion that no supplier splits his flow in Data Set 7, and all suppliers can route their flow along the shortest path from source to destination nodes in the network.

For Alternative 1 to have any use in practical applications, we must know that the equilibrium solution does not encompass suppliers splitting their flow because of link

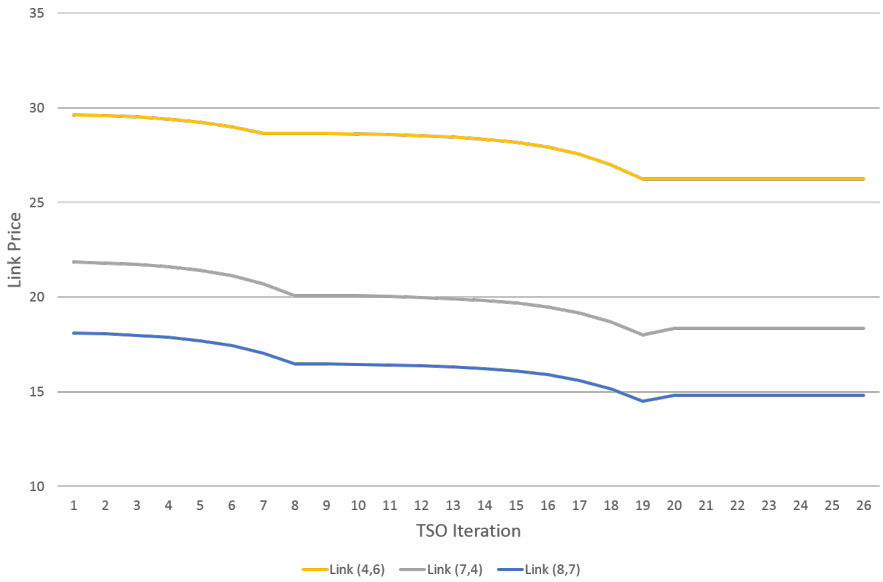

Figure 6.8: The development of three link prices from Data Set 7, Alternative 1. Only the first 24 iterations are shown.

transportation costs. If we know that every supplier routes his sales by the shortest path for all source-destination pairs in the network, **or** the suppliers split their flows due to capacity constraints, then Alternative 1 may be able to find a solution.

6.4.3 A closer look at Alternative 2

Using Alternative 2 on Data Set 2 results in some interesting observations. The solution time is much higher than we expected for a small 3-node network. Examine Figure 6.9 which shows both the commodity price and the transportation prices from Data Set 2 using Alternative 2. The price on link (1,2) quickly rises close to the equilibrium price. Link (1,2) is the only link in the network which is not congested, and so it is priced at marginal cost. The other two links have a positive congestion rent, and takes many additional iterations to get close to the equilibrium price. Notice also in Figure 6.9 that there is a clear correlation between how the commodity price is increased and how the transportation price is increased.

A highly probable reason why the price ascent is so slow is because there is a big

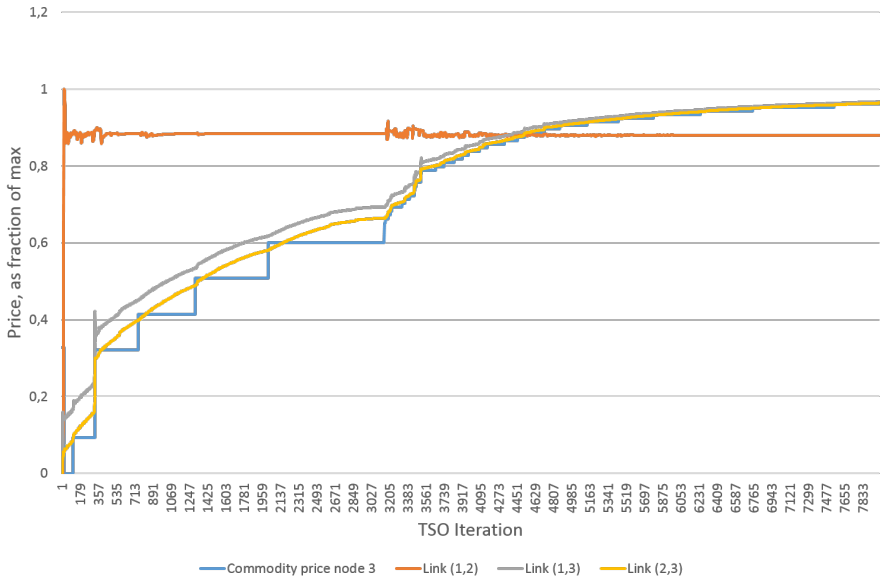

Figure 6.9: A plot showing the commodity price in node 3 and all link prices from the run of data set 2, alternative 2. The prices are feature scaled between 0 and 1.

difference between the link marginal cost at capacity and the equilibrium link price. In fact, the congestion rent is approximately 32 times larger than the marginal cost for link (2,3). The market in node 3 is clearly very attractive for the suppliers, and the TSO is able to charge a high price for the transportation services.

A big difference between the link marginal cost at full capacity and the equilibrium transportation price results in slow convergence by the following argument: first, the commodity price is adjusted by a low increment. When the commodity price increases slightly, the market becomes more attractive to the suppliers. When the suppliers want to sell more in node 3 the TSO is able to charge a higher congestion rent on the link. The following increase in transportation price is slow because this iteratively happens in small steps: recall from Algorithm 5, lines 2 trough 3 that the step length is relative to the gap between the previous link price and the current link price. The gap is small because the node price is adjusted in small increments as well: the transportation price will stabilize to match a given commodity price, and does not know that the commodity price will be increased further.

Still, Alternative 2 uses a lot of TSO iterations to increase the congestion rents for each adjustment of the commodity price. As we said, since the increments of the commodity price are small, the initial increase in the transportation prices are small. But with the "take bigger steps in the same direction" strategy, we should get an exponential growth in the transportation prices as the links should continue to be congested. Figure 6.9 shows that the opposite is happening, the increase in transportation prices slowing down as the search progresses! This unexpected behaviour is explained by the fact that there are two congested links along distinct paths from the supply node 1 to the demand node 3. As a capacity violation on the *first* link is detected, the congestion rent of this link start to increase. As the price increments get bigger, a higher fraction of the transportation demand from 1 to 3 gets routed through the *second* congested link up to a point where the flow along the *first* link is at capacity and the algorithm will stop changing it. This will stop the increase in the steps taken and when it starts to increase again it will be with the small initial step given by the $step_0$ parameter. In other words the two congested links will alternate between which has the biggest fraction of the transportation demand routed through them, and as such stop each other from gaining momentum towards the needed congestion rents. The change in transportation price from the previous iteration shows this clearly, and a plot is shown in Figure 6.10.

If we look at Data Set 6 we see that Alternative 2 uses over thirty times more time to solve this than the uncongested counterpart Data Set 7. Upon inspection we saw that Alternative 2 quickly reaches the first slack levels, but spends 1068 of the total 1230 TSO iterations on the final two slack parameters. Figure 6.11 shows the development of the link price for link (2,1) from TSO iteration 19 to TSO iteration 300. This link is congested in equilibrium with the congestion rent making up over half the link price. We see that the link price gets close to its equilibrium value after 90 iterations, but then has a series of downward spikes taking it further away from the equilibrium and then spending some iterations on getting closer again. These downward spikes are explained by how Alternative 2 adjusts link prices: the spikes occur when the link flow changes from being considered at capacity to below capacity. In the case of a link being below capacity, Alternative 2 adjusts the link price towards the marginal cost of the flow (Line 25

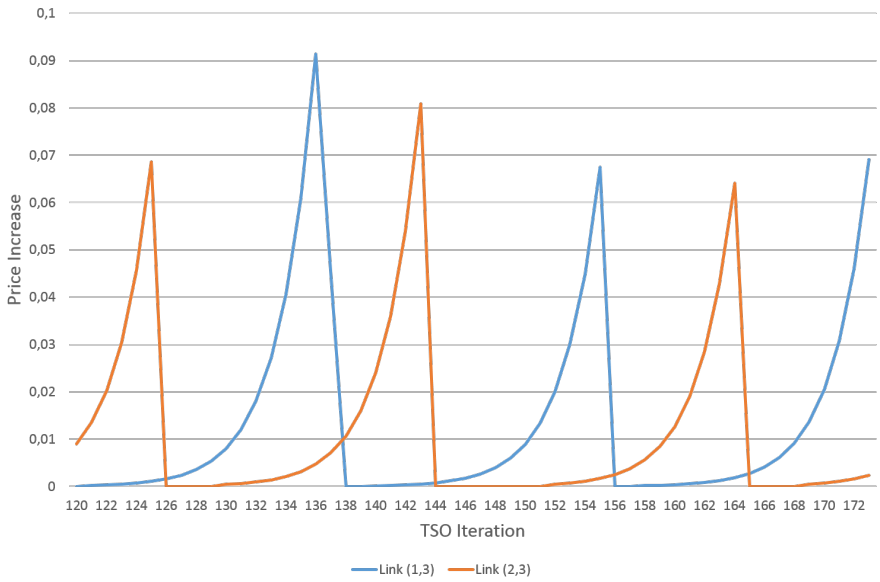

Figure 6.10: A plot showing the link price increase of the congested links (1,3) and (2,3) from the run of Data Set 2, Alternative 2. Notice how the prices increase to a point and then starts with small changes again.

of Algorithm 5). Since the congestion rent is high compared to the link price the distance from marginal cost is big and therefore the initial step taken becomes big. This causes a large change of price away from the equilibrium (the algorithm overshoots) and thereby slowing down the search. Hence Alternative 2 struggles with low slack requirements when the congestion rents are high relative to the marginal cost values. We believe the search strategy can be refined to handle this situation better, for example by the following **suggestion for further work:** When adjusting link prices, decrease link prices with non-zero congestion rents differently than those with prices lower than the marginal cost of flow at capacity.

6.4.4 A closer look at alternative 3

Alternative 3 has the same problem as Alternative 2 when solving Data Set 2. In general, the performance of Alternative 3 is similar to the performance of Alternative 2.

If we look at the performance of Alternative 3 on Data Set 8 we see that it has less

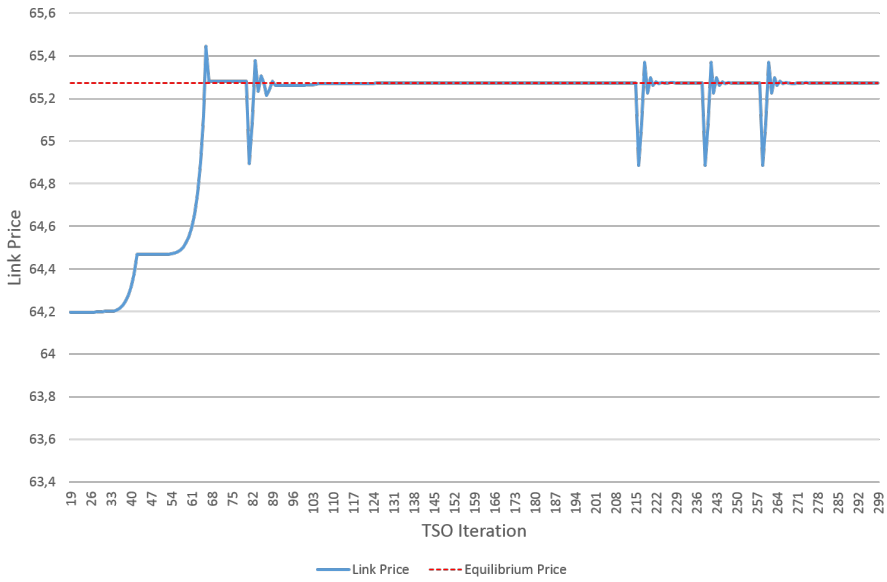

Figure 6.11: A plot showing the link price for link (2,1) in Data Set 6, Alternative 2. The blue line gives the prices set by the algorithm and the dotted red line gives the equilibrium price.

TSO iterations and commodity iterations than Alternative 2 (161 vs. 178 and 374 vs. 829 respectively), but still takes a longer time solving the problem (44.8 seconds vs. 40.9 seconds). Although the search mechanism of Alternative 3 finds equilibrium prices with fewer adjustments than Alternative 2, it uses more time which means that each iteration is more time consuming in Alternative 3 than Alternative 2. The only difference in terms of time consuming tasks per iteration for the two alternatives is that Alternative 3 needs to iterate over the SD pairs for each change in a congestion rent to adjust the SD prices accordingly (Line 6 of Algorithm 7). We believe this extra task is not of high enough cost to cause such a big difference in terms of time per iteration as we see between Alternatives 2 and 3. With no other differences between the two alternatives, there must be a significant time per iteration change caused by how the transportation prices are adjusted. If we look at the variance of the transportation demand set by the suppliers throughout the search, Alternative 3 has a 6.3% higher average variation in transportation demand per SD pair than Alternative 2. As the Mosek solver used to

solve the TSO problem is set to use the previous solution as a starting point, a higher variation in transportation demand throughout the search leads to the Mosek solver spending more time solving the TSO problems and thereby causing more time spent per TSO iteration. The same argument can be used for solving the suppliers problems: a higher variation in the commodity sales causes more expensive commodity iterations. This helps explain how Alternative 3 uses more time solving Data Set 8 than Alternative 2 even though it uses fewer iterations. In terms of optimal search strategy this also means there is an incentive to keep the variations of transportation demand and commodity sales low to decrease the solution time and thereby increase the algorithms performance.

We also see that Alternative 3 outperforms Alternative 2 on Data Set 6. When discussing Alternative 2 we saw that the poor performance came from the large congestion rents relative to marginal costs of the TSO. In Alternative 3 these problems are not present, and this is explained by the fact that the congestion rents are adjusted as its own entity, rather than through adjustments of the total link prices as in Alternative 2. To be more specific, when Alternative 3 sees a link with a positive congestion rent and flow below capacity it reduces the *congestion rent* based on the distance from capacity, rather than the *total link price* based on its *distance from the marginal cost* as in Alternative 2. This makes the initial steps taken when in the previous iteration the flow was considered on capacity much smaller for Alternative 3 than Alternative 2 when congestion rents are large, causing less disruption in transportation prices and faster convergence for Alternative 3 on Data Set 6.

6.4.5 The Number of Commodity Iterations vs. TSO Iterations

Figure 6.12 shows the average number of commodity iterations per TSO iteration for all data sets using Alternatives 2 and 3. The number of commodity iterations per TSO iteration seems to grow with the number of nodes in the network. Additionally, Alternative 2 seems to use more commodity adjustments than Alternative 3, where Data Sets 6 and 9 uses significantly more iterations that the other data sets.

Data Set 2, Alternative 2 has the lowest number of commodity iterations per TSO

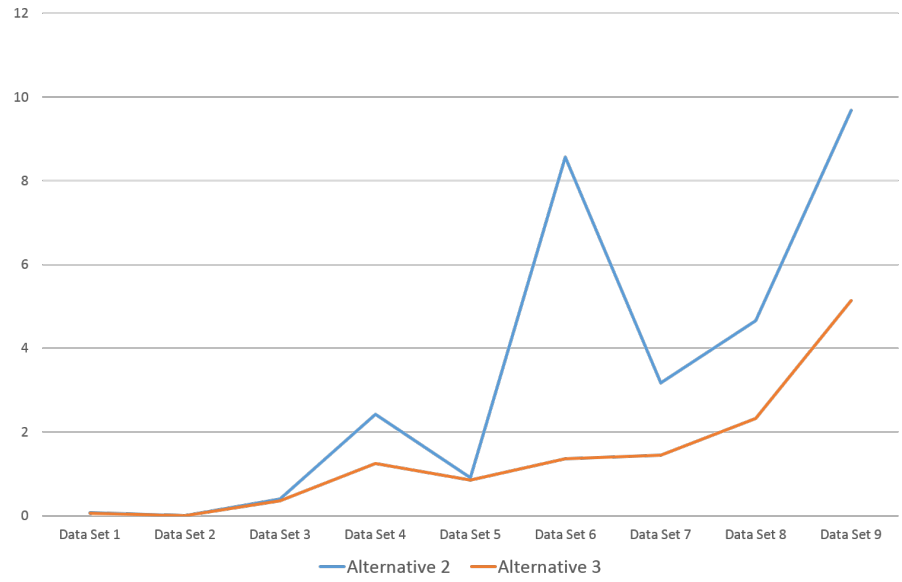

Figure 6.12: The average number of commodity iterations per TSO iteration for all the data sets, for Alternatives 2 and 3.

iteration with only 0.005 commodity iterations per TSO iteration. This is not due to a low number of commodity iterations, but a large number of TSO iterations (explained in Section 6.4.3). The same behaviour is seen by Alternative 3 on Data Set 2, which has 0.006 commodity iterations per TSO iteration.

In Data Set 6, Alternative 2 uses on average 8.57 commodity iterations per TSO iteration. In fact, the number of commodity iterations used by Alternative 2 on Data Set 6 is larger than all the other data sets combined! We have already discussed why Alternative 2 spends a long time solving Data Set 6: recall from Section 6.4.3 that it is related to a link where the congestion rent is relatively large compared to the marginal cost at link capacity. Figure 6.11 can also be used to explain the number of commodity iterations: when the link price takes a large initial step the commodity price will need adjustment as well. Because the link price takes a long step it is likely that the commodity price needs to take a long step as well, to be correct given current link prices. Upon inspection of the commodity price for node 1 in Data Set 6, Alternative 2 similar price movements is clearly visible: this is shown in Figure 6.13.

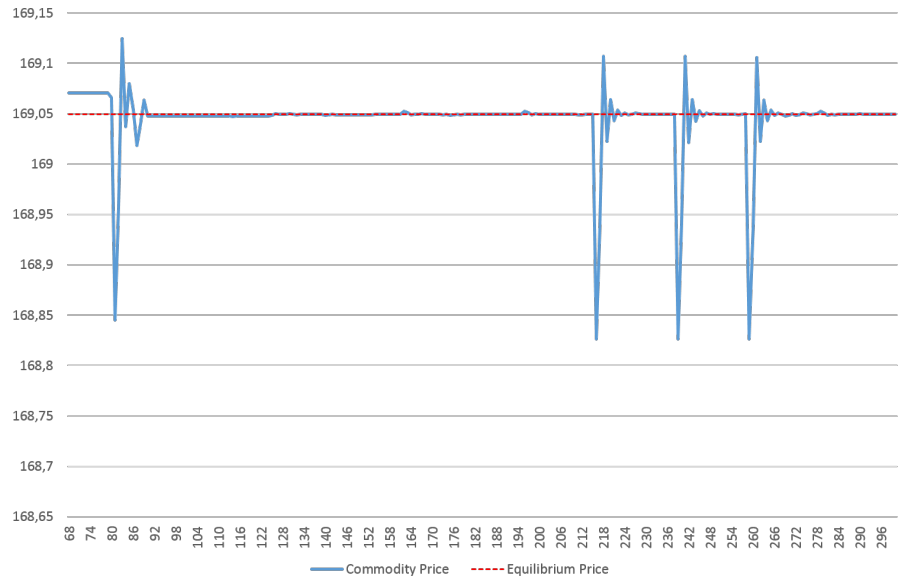

Figure 6.13: The commodity price in node 1 from the run of Data Set 6, Alternative 1. Notice the similarities to Figure 6.11.

Figure 6.12 also shows a difference between Alternatives 2 and 3. Even when disregarding Data Set 6, Alternative 2 uses in average 35 % more commodity iterations than Alternative 3. In comparison, Alternative 2 uses only 6.5 % more TSO iterations on average. We believe this difference in commodity iterations are related to how the two alternatives adjusts transportation prices as discussed in Section 6.4.4. Because Alternative 2 has a tendency to make larger steps when doing an initial correction of link prices, the number of iterations needed to correct the commodity price increases as well.

The average number of commodity iterations per TSO iterations increase with the number of nodes in the data set. If adjustments are made to several link prices, then it is more likely that the commodity price needs adjustment afterwards as well. In smaller network, it is possible that the commodity prices can remain unchanged for several price change adjustments.

In general, we can conclude that if an algorithm alternative is prune to a higher variation in the transportation prices it is likely that the number of commodity iterations

per TSO iteration will be larger, creating an incentive for making smaller adjustments in the transportation prices. This is the same incentive found in Section 6.4.4 to avoid making the solving of actor problems more time consuming. **Suggestion for further work:** Examine how the search strategy can be modified to reduce the variation of the market prices. This will improve performance by decreasing the number of commodity iterations needed as well as reducing the time used pr. both TSO and commodity iteration.

6.4.6 Congested Networks vs. Uncongested Networks

Based on the test results it seems that congestion is a time sink for alternatives 2 and 3. Alternative 2 always takes longer to solve the congested version of a data set than the uncongested version. Alternative 3 is faster on the congested 13-node set than the uncongested, but shows the same trends as Alternative 2 on the 5-node and 8-node set.

As noted in Section 6.4.3, we believe that the poor performance of Alternative 2 on the congested Data Set 6 can be improved by small corrections in the search strategy. We also believe that it should be possible to develop a mechanism that speeds up the growth of the congestion rents in Data Set 2, for example by the following **suggestion for further work:** Pre-process the total transportation capacity between supply and demand nodes (by solving a max-flow problem on the network) and check the SD transportation demand towards the SD capacities to identify needs to increase congestion rents. The effect of such improvements remains to be seen, and there might be other aspects of the algorithm and search strategy that makes it struggle with congested data sets.

The fact that Alternative 3 solves the congested 13-node set faster than the uncongested set indicates that the presence of congestion does not exclusively have bad influence on algorithm performance. The fact that PATH uses more than twice as much time to solve the congested 13-node set compared to that of the uncongested counterpart might in fact indicate that solving congested data sets is one of Alternative 3s stronger suits.

We conclude that there are a few aspects of congested networks that the algorithm

alternatives are currently struggling with, but the test base is too small to say anything decisive about whether congested sets will be harder to solve than uncongested sets.

In Section 3.2.2 we introduced a version of the TSO problem where asymptotically growing cost curves are used to avoid flow above link capacity instead of link capacity restrictions. It would be interesting to do a comparison between a congested data set and a cost-capacitated data set. On one hand, enforcing link capacity trough the link cost curve simplifies the transportation price adjustments by always adjusting towards marginal cost. On the other hand, the marginal costs of link flow would be very sensitive to transportation demands close to capacity (small changes in transportation demands would yield big changes in marginal cost), which might cause disruption in the transportation prices and thus slowing down the search. By this and the above reasoning, we think that developing the current approach using capacity restrictions seems like a better approach than switching to asymptotic cost curves in terms of algorithm performance.

6.4.7 Algorithm Parameter Sensitivity

The step length and initial slack parameters used in the tests are the ones we have found to give overall best performance for all the data sets.

The value of step length parameter $step_0$ has an impact on the performance of the algorithms. Recall that this value decides the length of the step taken when a transportation price needs adjustment but there was no change in the price in the previous iteration. A low $step_0$ value was set because influences from other price changes tends to be high in situations where the parameter is used. What we have seen when experimenting with different $step_0$ values is that when set too high the search slows down as the larger steps causes disruption among the other prices which in turn slows the search down, and when set too low the search slows down due to the the prices taking more iterations to get towards the wanted values. Table 6.4 shows solution times for all alternatives with $step_0$ values 0.1, 0.01 (the one used in the other results) and 0.001. We see that in some cases the alternative $step_0$ values performs better than the results previously shown (especially for the higher $step_0$ value on Data Set 2, which makes sense

considering the discussion in Section 6.4.3), but on average $step_0$ equal to 0.1 gives 18% longer solution times than $step_0$ equal to 0.01 and $step_0$ equal to 0.001 gives 48% longer solution times.

Similar variations to the other algorithm parameters ($step_1$, $step_2$ and the initial slack parameters) gives consistently worse performance for all alternatives.

6.4.8 Comparison to PATH

In this Section we will compare the algorithm developed in Chapter 5 to the commercial solver PATH, and also discuss how the different algorithm alternatives will scale on larger data sets. Because Alternative 1 only managed to solve 3 of the 9 data sets we have left it out of this discussion.

Alternatives 2 and 3 has comparable performance, but both are slower than PATH on every data set. Figure 6.14 shows a logarithmic plot of the run time of Alternatives 2 and 3 as well as PATH across all data sets. The sets are ordered by what we consider increasing difficulty, i.e. increasing in node size with the uncongested versions before their congested counterparts. The trends for Alternative 2 and 3 are very similar. Alternative 3 seems to grow a little slower than Alternative 2, but the testing base is too small to draw any decisive conclusions on differences in scalability. Looking at the trend line for PATH in Figure 6.14 it is clearly visible that the solution times of PATH grows slower than the two alternatives. We see that the difference between the two alternatives and PATH grows by a constant factor, and since the plot is logarithmic this indicate that the difference in solution time will grow exponentially. This is an unfortunate indication, but again we stress that the number of Data Sets is too small to make decisive conclusions about the run-time growth of either solution algorithm. We are also convinced that the algorithm we have implemented has potential for improvement. Especially in terms of scalability, as the current path-based formulation of the TSO problem has an extreme growth in number of path variables for increased number of links in the data set. We believe that reducing the number of path variables by removing unnecessary path variables in Algorithm 8 or switching to a link-based formulation will help the TSO problem scale a lot better. **Suggestion for further work:** Improve Algorithm 8

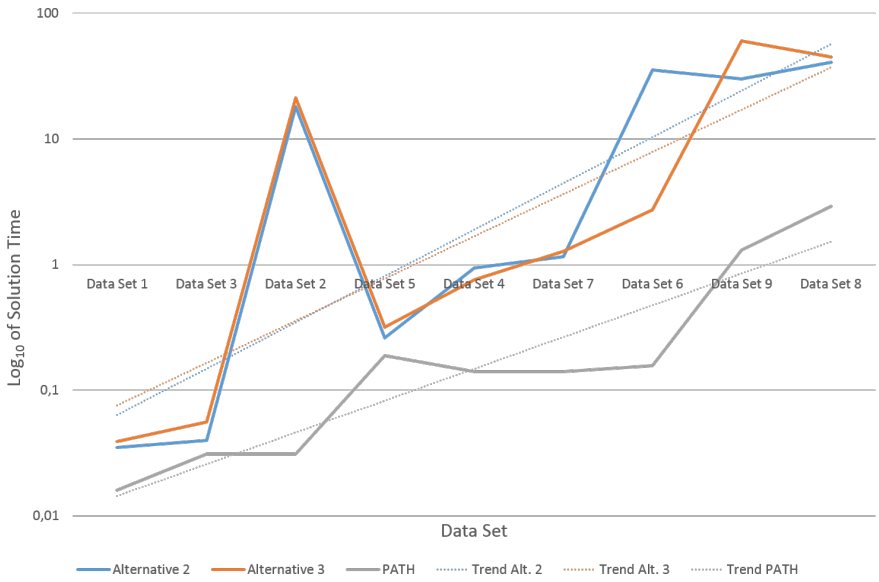

Figure 6.14: A logarithmic plot of the data set run times. The data sets are sorted by increasing node size, and the uncongested version before the congested version.

to remove unnecessary path variables or switch to a link-based formulation of the TSO problem. The analysis in this chapter have also exposed several aspects of the search strategy which can be altered to improve performance, indicating that the performance gap between the alternative algorithms and PATH can be greatly reduced.

6.4.9 Ranking the three Alternatives

As discussed in Section 6.4.2 we know that Alternative 1 has strict requirements to the equilibrium solution for it to be usable. Even if we can determine that a data set has this property at its equilibrium solution, Alternative 1 does not seem to gain any solution time advantage over neither Alternative 2 nor Alternative 3 (except for data set 2, which we already has discussed in Section 6.4.3). Because alternatives 2 and 3 are usable on every data set, we state that Alternative 1 performs worse than both other alternatives.

Alternatives 2 and 3 are harder to compare, and based on the discussions in Sections 6.4.3 and 6.4.4 we are unable to order them by performance. Alternative 2 handles uncongested networks *slightly* better than Alternative 3, but Alternative 3 is more ro-

bust to Data Sets where the congestion rents are large compared to marginal costs at link capacity.

	Alternative 1			Alternative 2			Alternative 3		
Data Set	**0.1**	**0.01**	**0.001**	**0.1**	**0.01**	**0.001**	**0.1**	**0.01**	**0.001**
Set 1: Simple supplier	0.054	0.029	0.042	0.044	0.035	0.049	0.041	0.039	0.055
Set 2: Capacity split	0.350	0.797	3.431	5.931	18.100	47.741	7.289	21.199	43.136
Set 3: Cost split				0.044	0.040	0.069	0.036	0.043	0.075
Set 4: 5-node congested				1.846	0.945	0.919	0.719	0.759	1.041
Set 5: 5-node uncongested				0.363	0.261	0.521	0.397	0.317	0.585
Set 6: 8-node congested				21.383	35.525	10.754	3.514	2.723	5.943
Set 7: 8-node uncongested	1.442	1.218	1.057	1.604	1.153	1.635	2.114	1.277	2.025
Set 8: 13-node congested				276.342	40.930	67.071	52.713	44.833	84.411
Set 9: 13-node uncongested				17.856	30.087	42.054	44.805	60.574	76.107

Table 6.4: Solution times for $step_0 = 0.1$, $step_0 = 0.01$ and $step_0 = 0.001$

Chapter 7

Summary and Conclusion

This chapter concludes our thesis. We present a summary of our work in Section 7.1 before we finally draw some conclusions in Section 7.2.

7.1 Summary

We have investigated the possibilities for creating a price based algorithm for solving energy market equilibrium models. Chapter 2 presented the necessary fundamentals of optimization, energy markets, algorithms and data structures that were utilized in the later chapters. Of particular importance were the Wardrop principles, which states the properties that must hold for the transportation market in an equilibrium solution.

Chapter 3 presented a single-period, single fuel, energy market equilibrium model with three types of actors: suppliers, consumers and transmission system operators. The goal of the model was to be representative of the type of models used in current research while not having a very high complexity level that would make analysis of a solution algorithm difficult. We believe the model is complex enough to be give decision support in energy markets. The suppliers in the model were given market power while the transmission system operators were perfectly competitive. This assumption on perfect competition allowed us to model the transmission system operators problems as a single transportation problem.

Chapter 4 presented different decompositions of the model presented in Chapter 3 to separate the different actor problems, allowing us to solve each actors problem as a single-firm optimization model with exogenous market prices. Two decompositions were found: one having the suppliers route the flow and another where suppliers request source-destination flows and the routing is left to the TSO. The decomposition where the suppliers route their flow is found to have dependencies between the suppliers and the TSO beyond the market prices. A solution algorithm based on this decomposition would have to take these dependencies into consideration when finding equilibrium solutions. The other decomposition has most of these dependencies removed at the cost of a more expensive TSO problem solving a traffic equilibrium model to route the transportation demand set by the suppliers.

Chapter 5 presented three alternative solution algorithms for solving the decomposed model presented in Chapter 4. All alternatives are based on giving the different actors in the model the same set of prices and get the optimal actors decisions given these prices. Based on the decisions of the actors it is possible to assess the different incentives to change the prices, and make adjustments to the prices in order to get them closer to a set of prices that will give equilibrium behaviour among the actors. All algorithm alternatives follow the same basic search: the commodity price is adjusted until commodity markets are cleared for a set of transportation prices and these transportation prices are then adjusted given the transportation demand set by suppliers after clearing commodity markets. This is repeated iteratively until both the commodity and transportation markets are cleared. The three alternative algorithms vary in terms of which decomposition they use and how the prices are adjusted. Alternative 1 uses the decomposition where the suppliers route the flow, and adjusts link prices to clear the transportation market. Alternative 2 uses the decomposition that has the TSO route the flow, and adjusts link prices to clear the transportation market in a similar way as Alternative 1. Alternative 3 has the TSO route the flow, but does not adjust link prices: it searches in total supply node to demand node transportation prices directly given the TSO decisions.

Chapter 6 provides test results of all three alternatives of the algorithm. The results

are analyzed and compared to the commercial solver PATH. Alternative 1 proves to be unable to handle data sets where the suppliers must split their flows due to link costs. Alternatives 2 and 3 both show advantages and disadvantages for different data sets, but both alternatives are able to solve all test data sets we provide. Neither alternative is able to solve a data set faster than the commercial solver PATH, and they seem to scale worse with increasing data set sizes.

7.2 Conclusion

We have shown that it is possible to decompose an energy market equilibrium model into several individual optimization problems with exogenous market prices. Even if the actors are responding to prices, the suppliers are able to keep their Cournot-competitive behaviour. Of the two decompositions presented, the one having the TSO problem formulated as a traffic equilibrium problem has proven to be the best to use in solving the energy market equilibrium model. The decomposition that has the suppliers routing the flow needs a mechanism for splitting flow due to transportation costs to be of use in practical modeling. We believe that it should be possible to create such a mechanism, for example by adapting heuristic methods for solving traffic equilibrium models.

The search strategy developed converges to the equilibrium market prices for all the data sets used in testing. The market prices are adjusted independently of each other, only considering the properties of the the link or commodity in question. We have described some aspects of the search strategy which can be improved, and in general a more holistic search strategy that consider the impact price changes has in a broader sense will greatly reduce the number of price adjustments needed to find equilibrium solutions.

In its current state, the price-based algorithm is slower than PATH and seem to scale worse on bigger data sets as well. We believe that the improvements such as those discussed above will help close the gap between PATH and the price-based algorithm, and eventually the price-based algorithm may be able to solve energy market equilibrium models that PATH and similar other solvers are unable to solve in reasonable time. The

TSO traffic equilibrium problem scales particularly bad with network size, and an effort should be done to reduce the solution time of this sub-problem. We suggest either restraining the number of paths generated or considering a link-based formulation could improve the overall algorithm performance particularly with regards to scalability.

The model used in this thesis includes the most important actors of energy market equilibrium models research, but is not comprehensive of all actors and functionality used in practice. In order for the model to be of better use as decision support new actors such as a storage system operators should be added. How the algorithm will handle more complexity in the model remains to be seen, both in terms of finding a price-based decomposition for the new actor and increase in solution times. Currently the algorithm clears one market before adjusting the prices in the other market, and adding a new market (e.g. storage market) in the same way means solving clearing the two first markets before adjusting prices in the new market. This would mean solving the current algorithm for each iteration of the new market added, implying a very high increase in solution times. On the other hand, the added complexity could be more integrated in the structure of the current algorithm and as such not increase solution times to such a high degree.

In this thesis we have focused on energy market equilibrium models, but both the model and solution algorithm can be applied to other multi-player equilibrium problems with the same structure.

All things considered, we believe the work of this thesis has contributed positively towards solving energy market equilibrium models faster. The price-based algorithm enables a more specialised solution approach than solvers currently used, and should be able to trade generality for better performance. More development remains before the algorithm can be of use in practical modelling, but the work of this thesis has showed that a price-based algorithm may help reduce solution times of energy market equilibrium models, enabling modellers to solve models previously restricted by insurmountable solution times.

Bibliography

ApS, M. (2015). *The MOSEK C optimizer API manual Version 7.1 (Revision 31)*.

Bazaraa, M. S., Sherali, H. D., Shetty, C., et al. (1993). *Nonlinear programming, theory and applications*, volume 610. John Wiley & Sons, New York,.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 4(1):238–252.

Boots, M. G., Rijkers, F. A., and Hobbs, B. F. (2003). Gastale: An oligopolistic model of production and trade in the european gas market. *Energy Research Centre of the Netherlands (ECN), ECN*.

Brown, S. P., Gabriel, S. A., and Egging, R. (2010). Abundant shale gas resources: some implications for energy policy. *Backgrounder. Washington, DC: Resources for the Future*.

Chapman, B., Jost, G., and Pas, R. v. d. (2007). *Using OpenMP*. MIT Press.

Chiang, A. C. (1993). *Elements of dynamic optimization*. McGraw-Hill.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introdution to Algorithms*. The MIT Press, 3rd edition.

Crompton, P. and Wu, Y. (2005). Energy consumption in china: past trends and future directions. *Energy economics*, 27(1):195–208.

Dafermos, S. C. and Sparrow, F. T. (1969). The traffic assignment problem for a general network. *Journal of Research of the National Bureau of Standards, Series B*, 73(2):91–118.

Dantzig, G. B. (1963). Linear programming and its extensions.

Dantzig, G. B., Orden, A., Wolfe, P., et al. (1955). The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific Journal of Mathematics*, 5(2):183–195.

de Chazournes, L. B. (1998). Kyoto protocol to the united nations framework convention on climate change. *UN's Audiovisual Library of International Law (http://untreaty.un.org/cod/avl/ha/kpccc/kpccc.html)*.

Egging, R. (2013). Benders decomposition for multi-stage stochastic mixed complementarity problems – applied to a global natural gas market model. *European Journal of Operational Research*, 226(2):341 – 353.

Egging, R., Holz, F., and Gabriel, S. A. (2010). The world gas model: A multi-period mixed complementarity model for the global natural gas market. *Elsevier*, 25:4016–4029.

Egging, R. G. (2010). *Multi-Period Natural Gas Market Modeling Applications, Stochastic Extensions and Solution Approaches*. Phd dissertation, University of Maryland, College Park.

Egging, R. G. and Gabriel, S. A. (2006). Examining market power in the european natural gas market. *Energy Policy*, 34(17):2762 – 2778.

ENTSOG (2014). Proc. 11th Conference on Applied Infrastructure Research (INFRADAY, Berlin, Germany, October 2014).

European Commission (2014). European commission - competition - energy - overview. `http://ec.europa.eu/competition/sectors/energy/overview_en.html`. Accessed: 2014-12-11.

Ferris, M. C. and Munson, T. S. (1998). Complementarity problems in GAMS and the PATH solver. *Journal of Economic Dynamics and Control*, 24(2):165–188.

Floyd, R. W. (1962). Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345–.

Gabriel, S. A., Conejo, A. J., Fuller, D., Hobbs, B. F., and Ruiz, C. (2012). *Complementarity Modeling in Energy Markets*. Springer.

Gabriel, S. A., Zhuang, J., and Egging, R. (2009). Solving stochastic complementarity problems in energy market modeling using scenario reduction. *European Journal of Operational Research*, 197(3):1028 – 1040.

Hass, J., Weir, M., and Thomas, G. (2008). *University Calculus*. Pearson international edition. Pearson Addison-Wesley.

He, X., Guo, X., and Liu, H. X. (2010). A link-based day-to-day traffic assignment model. *Transportation Research Part B: Methodological*, 44(4):597 – 608.

Hillier, F. S. and Lieberman, G. J. (2010). *Introduction to Operations Research*. McGraw-Hill, 9th edition.

Holz, F., von Hirschhausen, C., and Kemfert, C. (2008). A strategic model of European gas supply (GASMOD). *Energy Economics*, 30:766–788.

Huppmann, D. and Holz, F. (2009). A model for the global crude oil market using a multi-pool mcp approach.

Karush, W. (1939). *Minima of functions of several variables with inequalities as side constraints*. PhD thesis, Master's thesis, Dept. of Mathematics, Univ. of Chicago.

Kuhn, H. W. and Tucker, A. W. (1951). Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492, Berkeley, Calif. University of California Press.

Leuthold, F. U., Weigt, H., and von Hirschhausen, C. (2012). A large-scale spatial optimization model of the european electricity market. *Networks and Spatial Economics*, 12(1):75–107.

Lundgren, J., Rönnqvist, M., and Värbrand, P. (2010). *Optimization*. Studentlitteratur, 1st edition.

Nash, J. (1951). Non-cooperative games. *Annals of Mathematics*, 54(2):pp. 286–295.

Ortega, J. and Rheinboldt, W. (2000). *Iterative Solution of Nonlinear Equations in Several Variables*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics.

Otero-Novas, I., Meseguer, C., Batlle, C., and Alba, J. (2000). A simulation model for a competitive generation market. *Power Systems, IEEE Transactions on*, 15(1):250–256.

Pindyck, R. S. and Rubinfeld, D. (2005). *Microeconomics*. Upper Saddle River, NJ: Pearson Prentice Hall, 6th edition.

Rosen, K. (2011). *Discrete Mathematics and Its Applications 7th edition:*. McGraw-Hill Education.

Rosenthal, R. E. (2015). *GAMS - A User's Guide*.

Rutherford, T. F. (1993). Miles: A mixed inequality and nonlinear equation solver. Technical report, Citeseer.

Smith, M. (1979). The existence, uniqueness and stability of traffic equilibria. *Transportation Research Part B: Methodological*, 13(4):295 – 304.

Ventosa, M., Baíllo, A., Ramos, A., and Rivier, M. (2005). Electricity market modeling trends. *Energy Policy*, 33(7):897 – 913.

Wardrop, J. G. (1952). Some theoretical aspects of road traffic research. *ICE Proceedings: Engineering Divisions*, 1(3).

Warshall, S. (1962). A theorem on boolean matrices. *J. ACM*, 9(1):11–12.

Watling, D. P. and Cantarella, G. E. (2013). Modelling sources of variation in transportation systems: theoretical foundations of day-to-day dynamic models. *Transportmetrica B: Transport Dynamics*, 1(1):3–32.