Some Applications of the Weighted Combinatorial Laplacian

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Christian Szegedy

aus

Budapest

Bonn 2005

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Referent: Professor Bernhard Korte
2. Referent: Professor Jens Vygen

Tag der Promotion: 22.2.2005

Contents

Chapter 1

Introduction

The *Laplacian matrix* of a graph G (also called *combinatorial Laplacian*) is a basic and extensively studied object of graph theory. It can be regarded as a discrete version of the Laplacian operator and plays an important role in various contexts. For example, a simple argument using the Cauchy-Binet formula shows that its determinant is the number of spanning trees of G. Its eigenvalues are known to hold a lot of information about the structure of the graph[1] and can be used for estimating various graph invariants, most prominently the Cheeger constant[2]. This is the common base of most approaches to proving that certain families of graphs possess expander properties. The Laplacian matrix of the graph of an electric circuit holds important physical information[3].

The *generalized* or *weighted Laplacian* is a generalization of the ordinary Laplacian matrix and was first introduced in [Fiedler 1973]. It is equal to $\mathbf{DWD}^T$ where $\mathbf{D}$ is the arc-node incidence matrix[4] of the graph and $\mathbf{W}$ is a diagonal weight matrix (also referred to as *conductance matrix*). The weighted Laplacian is useful in a wide range of fields: it is an important object of study in the theory of graph partitioning[5], graph embeddings[6], graph visualization[7], web

[1] [Anderson and Morley 1971]

[2] [Lubotzky 1994] Chapter 4 and [Alon 1986]

[3] [Doyle and Snell 1984], [Lovász 1993]

[4] For non-oriented graphs, an arbitrary orientation can be chosen as $\mathbf{DWD}^T$ does not depend on the orientation.

[5] [Donath and Hoffman 1973],[Goemans and Rendl, 1999],[Pothen, Simon and Liou 1990] and [Hendrickson and Leland 1995]

[6] [Guattery and Miller 2000]

[7] [Brandes et al. 2000]

search[8], mesh optimization[9], random walks[10], image processing[11], Feynman diagrams[12], and nonlinear control[13].

In this work, we will look at three recent deep and interesting applications of the weighted Laplacian matrix. In the above-mentioned contexts the combinatorial Laplacian was traditionally considered over the field of real numbers with positive arc weights. However it turns out that the weighted Laplacian over multivariate polynomial fields of characteristic 2 provides a valuable tool for studying matching-theoretical properties of graphs. We will show in Theorem 2.6.6 that a graph is factor-critical if and only if a suitably weighted Laplacian has maximum rank. This particular matrix can also be regarded as the best possible approximation of the Tutte matrix by a weighted Laplacian. This theorem is reformulated in Corollary 2.10.2 which states that a graph is factor-critical if and only if it has a weighted Laplacian with corank 1 which is an alternating matrix. It also extends to graphs that are not factor-critical: the minimum corank of the alternating weighted combinatorial Laplacian is equal to $1 + \varphi(G)$ where $\varphi(G)$ is the graph invariant introduced by A. Frank[14] denoting the minimum number of even ears an ear-decomposition of a graph can have[15]. In Theorem 2.9.5, we generalize this surprising result to arbitrary matroids representable over fields of characteristic 2 which gives rise to a very simple randomized-polynomial time algorithm[16] for the computation of φ. The proofs for the matroid case are joint work with Balázs Szegedy. In Section 2.11, to demonstrate the usefulness of this theory, we will give randomized polynomial-time algorithms for some graph orientation problems whose complexity status is not yet fully explored. The results presented in this thesis strongly suggest that these problems are solvable in polynomial time. Moreover, associated spaces and projection matrices will be shown to hold interesting combinatorial information too (cf. Theorems 2.6.8 and 2.8.8).

Other applications are from the area of the design process of very large scale integrated circuits. As already mentioned, the Laplacian matrix of

[8] [Drineas et al. 1999]

[9] [Kaveh and Bondarabady 2000]

[10] [Doyle and Snell 1984] and [Lovász 1993]

[11] [Marr and Hildreth 1980] and [Grady 2004]

[12] [Nakanishi 1971]

[13] [Meshaby and Hadaegh 2001] and [Slotine and Wang 2003]

[14] [Frank 1993]

[15] In fact A. Frank has shown that φ is tightly connected to the covering radius ρ of the cycle code $\mathcal{C}(G)$ of G by the formula $2\rho(\mathcal{C}(G)) = \mathrm{rk}\,(G) + \varphi(G)$. This theorem is inherently graph-theoretical and does not generalize to binary matroids.

[16] A polynomial-time combinatorial algorithm was given in [Frank 1993].

some graphs associated with the circuitry holds important physical information. The combinatorial Laplacian also defines a quadratic form that can be used to optimize the placement of the small building blocks (gates) that make up the design. This old and well-known approach goes back to [Wipfler, Wiesel and Mlynski 1983] and is adopted by the most successful layout tools for very large scale designs[17]. To improve the timing behaviour of the layout, net weighting techniques were first proposed in [Burstein and Youssef 1985] and in connection with quadratic placement in [Tsay and Koehl 1991][18]. The quadratic form to be optimized in this case is given by a weighted Laplacian of the same graph.

The contribution to this topic presented in Section 4.3 is a combination of the net weighting approach with a subgradient-method-based framework which provides a provably convergent algorithm for minimizing the power consumption of a design under additional timing constraints disregarding the disjointness constraints. Our method resembles that of [Srinisavan, Chaudary and Kuh 1992], where for the first time a subgradient-method based scheme was proposed. However the algorithm proposed there has obvious flaws: the dual function typically attains infinite values and the authors seem to ignore this fact. Moreover, our method works provably optimal when the clock arrival times are left variable. The new method is more similar to the one proposed in [Chen, Chu and Wong 1999] for solving the gate- and interconnection-sizing problem. New approaches to solve linear, quadratic and mixed linear-quadratic-multifacility location problems are presented at the end of that section.

The Lagrangian framework requires a subroutine for solving minimum cost flow problems with a quadratic cost function. The fastest known algorithm[19] for this problem is based on a Newton-type algorithm, where the Hessian is replaced by a weighted combinatorial Laplacian with appropriately adjusted weight matrix.

However, several authors[20] noticed that the constrained subgradient method for gate-sizing using exact projection is not very efficient in practice since all known exact projection methods scale superlinearly in the number of nodes of

[17] [Kleinhans et al. 1991], [Sigl, Doll and Johannes 1991], [Tsay and Kuh 1991], [Alpert et al. 1997a] and [Vygen 1997]

[18] Their idea was also extended to include clock-skew optimization in [Hurst, Cong and Kuehlmann 2004].

[19] [Ibaraki, Fukushima and Ibaraki 1991]

[20] including [Chen, Chu and Wong 1999] and [Sechen and Tennakoon 2002]

the timing graph. The common solution[21] is to use projection heuristics which may work well in a lot of situations but without provable convergence. One of the major results of this thesis is a common generalization of the constrained subgradient method and the method of cyclic projections in Section 3.5. We will show that if the feasible region is the intersection of some closed convex sets (with a newly introduced property possessed by any family of polyhedral sets) then the projection in the constrained subgradient method[22] can be replaced by successive projections onto the intersecting sets and the subgradient method will converge. For the specific applications of this method to gate-sizing or timing driven placement, one has to project the Lagrangian multipliers to the flow space of the the graph defining the timing constraints (*timing graph*). This can be efficiently computed by minimizing the quadratic form defined by the Laplacian of the timing graph. In fact, the system of linear equations to be solved is very similar to the one solved by analytical placement tools. These results give rise to the first theoretically justified practical algorithms for large scale instances. The results presented in this section are joint work with Dieter Rautenbach.

In this thesis we will also revisit the gate-sizing algorithm of proposed in [Chen, Chu and Wong 1999]. One step in this algorithm is the so called local refinement method, which minimizes the weighted sum of delays and powers with respect to a given set of Lagrangian multipliers. The linear convergence of this method follows immediately from the results of in [Luo and Tseng 1992]. A special proof of linear convergence with new error bounds was presented in [Chu and Wong 1999] for timing graphs with tree-topology. We will give a much simpler proof a greatly generalized version of this problem in Section 4.4.8. The general case handled here contains the case of arbitrary timing graphs and we arrive at similar error-estimations as in the above mentioned special csae.

The presented timing driven placement approach can easily be combined with established partitioning heuristics to produce overlap-free placements. So our methods can be extended to a fully functional performance-driven placement algorithm simultaneously optimizing the gate-sizes and placement of designs with millions of movable objects in reasonable time on current workstations. The theoretical approach described in this thesis is fully implemented in C++ as part of a long-term cooperation project with IBM and tested on real-world chips. It is part of a tool collection called BonnTools developed at the Institute

[21] proposed in [Muuss 1999] and [Sechen and Tennakoon 2002]

[22] [Ermoliev 1966], [Polyak 1967] and [Polyak 1969]

for Discrete Mathematics, University of Bonn.

The implementation of this framework requires a lot of care and clever tricks in order to ensure good run-time and high quality solutions at the same time. These ideas are often of mathematical nature, but based on formally not justified intuition. Some of the most important details will be discussed in Section 4.5.

Experimental results will be presented in a conclusive section.

Acknowledgements

I am very grateful and indebted to the following people. Their contributions were essential to the quality of this thesis.

The encouragement, feedback and insights of Dieter Rautenbach were extremely valuable. The results in Sections 3.5 and 4.4.8 are joint work with him. He has also proof-read large parts of this thesis.

The results in Section 2.9 are joint work with Balázs Szegedy. He has also put a lot of effort into making Chapter 2 more readable including a lot of structural changes and simplified proofs.

I had a lot of very helpful discussions with Jens Vygen on the topic of timing driven placement and timing in general. He suggested studying the timing driven placement problem as described in 4.3 and hinted at the approach to slew-aware gate-sizing in Section 4.4.10. He has also proof-read the whole thesis and made a lot of essential structural suggestions.

Jürgen Werber has also proof-read large parts of this work and proposed several improvements. He is a great colleague and it was a lot of fun to work with him on the BonnTime project.

It was a great pleasure to work with Stephan Held on the BonnTime project. His work made significant contributions to the excellent performance of the timing engine of BonnTime. He is an outstandingly helpful colleague and made valuable suggestions to the section on clock-skew scheduling.

Christoph Bartoschek was also a great collaborator on the BonnTime project and made important experiments with the gate-sizing tool.

I had a lot of very helpful discussions with Markus Struzyna on timing driven placement. He brought Weiszfeld's original paper to my attention. He has also done excellent work on providing usable interfaces to the placement engine, which was essential for testing our ideas in practice.

I learned a lot from Zoltán Szigeti about matching theory and ear-decompositions. Chapter 2 would not exist without his influence.

I had some very stimulating discussions with András Frank on the topic of Chapter 2.

Jürgen Schietke has taught me a lot about timing and timing optimization in general.

Karsten Muuss was always ready to share his valuable insights about the gate-sizing and about a lot of other topics with me.

Katharina Langkau has developed an earlier version of the gate-sizing tool. I was happy to collaborate with her.

The experience of Matthias Ringe was very helpful for designing the timing optimization tool. He was always ready to answer questions about a broad range of topics in chip-design.

Last, but not least, I would like to express my gratitude to my supervisor Bernhard Korte. This thesis would not have been possible without his encouragement, guidance and help.

Chapter 2

Weighted Laplacian in Matching Theory

The aim of this chapter is to give new insights into the algebraic structures underlying matching theory, especially the structure of factor-critical graphs and the related ear-matroid introduced by A. Frank and Z. Szigeti[1]. We also show that some important results can be generalized from graphs to matroids representable over a field of characteristic 2.

In the last section of this chapter the algebraic theory is utilized to give randomized polynomial time algorithms to solve some problems whose exact complexity status is currently unknown. The results in Section 2.11 suggest that these problems are solvable in polynomial time.

Inherent connections between matching theory and algebra were already recognized by Tutte, who proved his famous characterization[2] of perfectly matchable graphs using the Tutte-matrix. More recent examples of similar techniques are L. Lovász's algebraic description of matroid-parity[3], W. Cunningham's and J. Geelen's work[4] on the path-matching problem. Most recently the very general problem of linear Δ-matroid matching has been solved[5] by J. Geelen and S. Iwata by using mixed skew-symmetric matrices over polynomial fields.

[1] [Frank 1993], [Szigeti 1994] and [Szigeti 1996]
[2] [Tutte 1947]
[3] [Lovász 1979]
[4] [Cunningham and Geelen 1997]
[5] [Geelen and Iwata 2003]

The first main result[6] of this chapter is Theorem 2.6.6 stating that a graph G is factor-critical if and only if there exists a weighted Laplacian for G of corank 1 which is an alternating matrix[7]. Of course this is only possible if the base field is of characteristic 2 since every weighted Laplacian is symmetric. However one can not restrict oneself to GF(2). The result can also be reformulated by looking at a single weighted Laplacian with suitable weights which are multivariate polynomials over GF(2). This matrix can also be regarded as the *most generic approximation of the Tutte matrix by a weighted Laplacian.*

F. Jaeger observed[8] that some fundamental properties of graphs and binary matroids, such as being bipartite, Euler, graphic or planar, can be characterized in terms of symmetric representations over GF(2). Theorem 2.9 gives a similar characterization of factor-criticality: a graph is factor-critical if and only if its cycle matroid can be represented by an alternating projection matrix. This is a more subtle result than the characterizations of F. Jaeger, since we cannot restrict ourselves to binary representations, but must consider arbitrary ground fields of characteristic 2. There are examples of ear-matroids of binary spaces (or even of graphs) that do not have binary representations.

An equivalent and more generic reformulation of the above criterion is the following statement. A bridgeless graph is factor-critical if and only if its cycle matroid can be represented by a space (over fields of characteristic 2) on which the induced scalar product is a nondegenerate symplectic form.

A novel algebraic method in our treatment is the symplectification of spaces. This enables us to represent a matroid by a new space whose scalar product is symplectic. We will show that the bicycle space of the symplectified representation holds important combinatorical information: the bases of its matroid are those minimal edge sets whose contraction results in a factor-critical graph (or matroid, in the general case). This also proves the result conjectured in [Frank 1993] and first proved in [Szigeti 1996]. The proof presented here is fundamentally different. It works for matroids representable over some field of characteristic 2 (instead of graphs) and also yields an explicit representation of the matroid in question.

This result will be further generalized in the following way: those edge sets whose contraction results in a factor-critical matroid form the feasible sets of a representable Δ-matroid. In fact, it turns out that a submatrix of the projection matrix on the symplectified representing space of some suitable

[6] [Szegedy 1999]

[7] skew-symmetric with zero diagonal entries

[8] [Jaeger 1983a] and [Jaeger 1983b]

subdivision represents this Δ-matroid. The results presented in the sections about the general case of matroids are joint work with Balázs Szegedy[9].

It is worth noting that in general the matroid of the bicycle space (even its rank) depends on the chosen representation. The above results show that the matroid of the bicycle space of a symplectified representation is uniquely determined by the matroid. One must also note that the dimension of the bicycle space of different symplectic representations may vary too – the symplectification of a symplectic representation may alter the matroid of the bicycle space.

The interest in factor-criticality and ear-decomposition is also justified by the fact that factor-critical graphs play the role of elementary building blocks in several central decomposition theorems in matching theory[10]. Ear-decompositions play an important role in the theory of machine learning[11] too. A. Frank proved[12] that function φ (the main object of study in this Chapter) of a graph G is tightly connected to the covering radius ρ of the cycle code by the formula $2\rho(\mathcal{C}(G)) = \mathrm{rk}\,(G) + \varphi(G)$. In fact this deep and surprising result was the starting point and main motivation for the research presented in this chapter.

2.1 Factor-Criticality and Ear-Decomposition

Let $G = (V, E)$ be a graph with vertex set V and edge set E. Throughout the chapter we allow loops and parallel edges in G. A *matching* of G is a subset of E consisting of non-loop edges such that no vertex of G is covered by more than one edge. A *perfect matching* or 1-*factor* of G is a matching that covers all vertices of G. A graph G is called *factor-critical* if the subgraph obtained by removing any vertex has a perfect matching. A simple parity argument shows that factor-critical graphs are 2-edge-connected.

An *ear-decomposition* D of a graph G is a sequence $(G_0, ..., G_k = G)$ of graphs such that G_0 is the one-vertex graph and G_{i+1} is constructed from G_i by adding a simple path (ear) between two vertices of G_i such that the other vertices of the path are not in the vertex set of G_i. We denote by $e(D)$ the number of ears with an even number of edges in an ear-decomposition D and by $\varphi(G)$ the

[9] [Szegedy and Szegedy 2004]
[10] [Lovász 1986]
[11] [Coullard and Hellerstein 1996]
[12] [Frank 1993]

minimum of $e(D)$ over all ear-decompositions D of G. Note that if G is 2-edge-connected then it has at least one ear-decomposition. An ear-decomposition D is called *optimal* if $e(D) = \varphi(G)$. One of the basic properties of φ is that inserting new edges parallel to an existing edge of G does not alter its value. It is obvious that each connected graph can be made 2-edge-connected by applying some such operations. So we can define φ for an arbitrary connected graph G by letting $\varphi(G) \stackrel{\text{def}}{=} \varphi(G')$ for some in this way extended graph G'. For an unconnected graph G we define $\varphi(G)$ to be the sum of $\varphi(C)$ over all components C of G.

The following theorem connects factor-criticality to ear-decompositions:

Theorem 2.1.1 ([Lovász 1972]) *A connected graph G is factor-critical if and only if $\varphi(G) = 0$.*

Corollary 2.1.2 *A connected graph G is factor-critical if and only if G can be obtained from the one-vertex graph K_1 by using the following operations:*

(1) Add a new edge between two existing (not necessarily different) vertices

(2) Replace an edge by a path of length 3

The second operation is called *double subdivision* of an edge.

The reader is assumed to be familiar with the basics of matroid theory and we will use the standard notation introduced in [Welsh 1976].

Let M be a matroid with edge set E. An ear-decomposition of M is a sequence of circuits $C_0, C_1, ..., C_k$ of M with the following properties:

(1) $C_i \setminus (\cup_{j=0}^{i-1} C_j)$ is not empty for all $1 \leq i \leq k$

(2) $C_i \cap (\cup_{j=0}^{i-1} C_j)$ is not empty for all $1 \leq i \leq k$

(3) $C_i \setminus (\cup_{j=0}^{i-1} C_j)$ is a circuit in $M/(\cup_{j=0}^{i-1} C_j)$ for all $1 \leq i \leq k$

(4) $\cup_{j=0}^{k} C_j = E$.

The sets $C_i \setminus (\cup_{j=0}^{i-1} C_j)$ and the set C_0 are called *ears*. An ear is said to be odd (resp. even) if it consists of odd (resp. even) number of edges. We say that D is an *odd ear-decomposition* if all ears occurring in D are odd. Let M be a connected, bridgeless matroid. Similarly to graphs, we denote by $\varphi(M)$ the minimal possible value of the number of even ears in an ear-decomposition of

M. If M is bridgeless but not connected, we define $\varphi(M)$ to be the sum of $\varphi(K)$ over all blocks K of M. In particular $\varphi(M) = 0$ if and only if every block of M has an odd ear-decomposition.

Lemma 2.1.3 *Let M be a bridgeless matroid. Then $\varphi(M) = 0$ if and only if the edge set of M can be partitioned into sets $E_0, E_1, ..., E_k$ with an odd number of edges in each of them such that E_0 is a circuit and E_i is a circuit in $M/\left(\cup_{j=0}^{i-1} E_j\right)$.* □

In other words, $\varphi(M) = 0$ if and only if the edge set of M can be eliminated by a process in which we contract an odd circuit in each step.

Definition 2.1.4 *The bridgeless matroid M is defined to be factor-critical if $\varphi(M) = 0$.*

2.2 (E, F)-Spaces

We will extensively use the notion of *(E, F)-spaces*. Let F be an arbitrary field and let E be a finite set. We call a subspace $U \subseteq F^E$ an (E, F)-space and we refer to the elements of E as *edges*. A *generating matrix* of an (E, F)-space U is a matrix over F such that its columns are indexed by the elements of E and its rows generate U. We call a generating matrix *minimal* if its rows are linearly independent. It is well-known that the column matroid of a generating matrix is uniquely determined by the space. So we can associate this matroid $\mathcal{M}(U)$ with the space U itself. Let $S \subseteq E$ be a subset of the edge set and $\mathbf{u} \in U$ be a vector. We denote by $\mathbf{u}_S \in F^E$ the vector that we obtain by setting to zero all components of u which correspond to edges in S. Let $\mathbf{u}|_S \in F^S$ denote the vector that comes from $\mathbf{u}$ by omitting the components of $\mathbf{u}$ that are not in S. We will use the induced scalar product $\langle \mathbf{x}, \mathbf{y} \rangle \stackrel{\text{def}}{=} \sum \mathbf{x}_e \mathbf{y}_e$ on F^E and we denote by $U^\perp$ the orthogonal subspace of U with respect to this scalar product in F^E. Space U is called symplectic if $\langle \mathbf{x}, \mathbf{x} \rangle = 0$ for all $\mathbf{x} \in U$. Obviously, a bilinear form f is symplectic, if and only if $f(\mathbf{x}, \mathbf{y}) = \mathbf{x}^t \mathbf{A} \mathbf{y}$ where $\mathbf{A}$ is an *alternating matrix*, i.e. a skew-symmatric matrix with zero diagonal entries. The condition on the diagonal entries is interesting for fields of characteristic 2. In this case, skew-symmetricity is equivalent to symmetricity.

For a vector $\mathbf{u} \in U$ let $\operatorname{supp} \mathbf{u} \subseteq E$ denote the set of indices of non-zero components of $\mathbf{u}$ and let $\operatorname{supp} U \stackrel{\text{def}}{=} \bigcup_{\mathbf{u} \in U} \operatorname{supp} \mathbf{u}$. Note that $\operatorname{supp} U$ consists exactly of the non-loop elements of $\mathcal{M}(U)$. The characteristic vector of set

$S \subseteq E$ in F^E will be denoted by $\boldsymbol{\chi}_S$. We will make use of the following well known facts:

Lemma 2.2.1 *Let U be an (E, F)-space. Edge $e \in E$ is a bridge in $\mathcal{M}(U)$ if and only if $\boldsymbol{\chi}_e \in U$.* □

Let M be an arbitrary matroid with edge set E and let $C(e)$ denote the set of all circuits containing an edge $e \in E$. We say that e and e' are in the same *series class* if $C(e) = C(e')$.

Lemma 2.2.2 *Let U be an (E, F)-space. Then two edges e and e' are in the same series class if and only if $\boldsymbol{\chi}_e + a\boldsymbol{\chi}_{e'} \in U$ for some $0 \neq a \in F$.* □

We will also use the following notation:

$$U/S \stackrel{\text{def}}{=} \left\{\mathbf{u}|_{E\setminus S} \in F^{E\setminus S} \mid \mathbf{u} \in U \text{ and } \operatorname{supp}\mathbf{u} \subseteq E \setminus S\right\} \text{ and}$$

$$U \setminus S \stackrel{\text{def}}{=} \left\{\mathbf{u}|_{E\setminus S} \in F^{E\setminus S} \mid \mathbf{u} \in U\right\}.$$

Note that both U/S and $U \setminus S$ are $(E \setminus S, F)$ spaces.

The following well-known formulas will also prove useful.

Proposition 2.2.3

$$\mathcal{M}(U \setminus S) = \mathcal{M}(U) \setminus S \ , \ \mathcal{M}(U/S) = \mathcal{M}(U)/S \ , \ \mathcal{M}(U^\perp) = \mathcal{M}^*(U).$$

$$U/S \subseteq U \setminus S \ , \ (U \setminus S)^\perp = U^\perp/S \ , \ (U/S)^\perp = U^\perp \setminus S$$

$$(U \cap V)/S = (U/S) \cap (V/S) \ , \ (U \cap V) \setminus S \subseteq (U \setminus S) \cap (V \setminus S). \square$$

We will use the shorthand notation U/e and $U \setminus e$ for $U/\{e\}$ and $U \setminus \{e\}$, respectively.

We call a subset S of the edge set of a graph a *cut* or a *cutset* if there is a bipartition of the vertex set of G so that S is the set of edges having an incident vertex on both sides of the bipartiton. For example, $\emptyset$ is a cut of every graph. The set of characteristic vectors of all cuts of a graph $G = (V, E)$ over GF(2) is an (E, F)-space $S(G)$, called the *cutset space of G*. It is well-known that $\mathcal{M}(S(G))$ is the cycle-matroid of G.

2.3 The Bicycle Space of an (E, F)-Space

Let U be an (E, F)-space. We define the *bicycle space* $\mathcal{B}(U)$ of U by $\mathcal{B}(U) \stackrel{\text{def}}{=} U \cap U^{\perp}$ and $\beta(U)$ will denote its dimension. The following lemma describes how to determine $\beta(U)$ in terms of a generating matrix of U, the straightforward proof of which is left to the reader:

Lemma 2.3.1 *Let* $\mathbf{G}$ *be a generating matrix of an* (E, F)*-space* U*. Then* $\beta(U) = \dim(U) - \mathrm{rk}(\mathbf{G}\mathbf{G}^T)$*. If* $\mathbf{G}$ *is minimal (that is its rows are independent), then* U *is bicycle free if and only if* $\det(\mathbf{G}\mathbf{G}^T) \neq 0$*.* □

Note that $\mathcal{B}(U) = 0$ means that the scalar product induced on U is non-degenerate.

Lemma 2.3.2 *If* U *is an* (E, F)*-space and* $S \subseteq E$*, then:*

(1) $\mathcal{B}(U)/S \subseteq \mathcal{B}(U/S)$,

(2) $e \in \operatorname{supp} \mathcal{B}(U) \Longrightarrow \mathcal{B}(U/e) = \mathcal{B}(U)/e$.

Proof: The first statement is an immediate consequence of the formulas of Proposition 2.2.3:

$$\mathcal{B}(U)/S = (U \cap U^{\perp})/S = (U/S) \cap (U^{\perp}/S) \subseteq (U/S) \cap (U^{\perp} \setminus S) = \mathcal{B}(U/S).$$

The inclusion $\mathcal{B}(U)/e \subseteq \mathcal{B}(U/e)$ from the second statement follows from the first statement. To prove the other inclusion, take an arbitrary vector $\mathbf{v} \in \mathcal{B}(U/e)$. By definition, there is a vector $\mathbf{v}_1$ in U such that the e component of $\mathbf{v}_1$ is 0 and the restriction of $\mathbf{v}_1$ to $E \setminus \{e\}$ is $\mathbf{v}$. We have to prove that $\mathbf{v}_1 \in U^{\perp}$. Assume the contrary, $\mathbf{v}_1 \notin U^{\perp}$. Then, since $\mathbf{v} \in (U/e)^{\perp} = U^{\perp} \setminus e$, there is a nonzero $\alpha \in F$ such that $\mathbf{v}_1 + \alpha \boldsymbol{\chi}_e \in U^{\perp}$. So we obtain that $\boldsymbol{\chi}_e \in U + U^{\perp} = \mathcal{B}(U)^{\perp}$, contradicting $e \in \operatorname{supp} \mathcal{B}(U)$. □

Theorem 2.3.3 *Let* U *be an* (E, F)*-space and* $M \subseteq 2^E$ *the family of sets* $S \subseteq E$ *for which* $\dim \mathcal{B}(U/S) = \dim \mathcal{B}(U) - |S|$*. Then* M *is exactly the family of the independent sets of* $\mathcal{M}(\mathcal{B}(U))$ *and for any* $S \in M$*, we have* $\mathcal{B}(U/S) = \mathcal{B}(U)/S$*.*

Proof: First we show that if S is an independent set of $\mathcal{M}(\mathcal{B}(U))$ then $\dim \mathcal{B}(U/S) = \dim \mathcal{B}(U) - |S|$. For $|S| = 0$ the statement is trivial. We go by induction on $|S|$. Assume that the statement holds for every $|T| < |S|$ and (E', F)-space V, where E' is an arbitrary finite set. Let e be an element

of S. Since e is not a loop of $\mathcal{M}(\mathcal{B}(U))$ we know that $\mathcal{B}(U/e) = \mathcal{B}(U)/e$ by Lemma 2.3.2. It follows that $\dim \mathcal{B}(U/e) = \dim \mathcal{B}(U) - 1$. Now $S \setminus \{e\}$ is an independent set of $\mathcal{B}(U)/e$, and so we can use our induction hypothesis:

$$\dim \mathcal{B}(U)/S = \dim \mathcal{B}(U/e)/(S\setminus\{e\}) = \dim \mathcal{B}(U/e) - (|S|-1) = \dim \mathcal{B}(U) - |S|.$$

This also proves the last statement of the theorem.

For the other direction, assume that $\dim \mathcal{B}(U)/S = \dim \mathcal{B}(U) - |S|$ and S is dependent in $\mathcal{M}(\mathcal{B}(U))$. So there is an $e \in S$ such that e is a loop in $\mathcal{M}(\mathcal{B}(U/(S \setminus \{e\})))$. By Lemma 2.3.2, this implies

$$\mathcal{B}(U/S) \supseteq \mathcal{B}(U/(S \setminus \{e\}))/e \supseteq \mathcal{B}(U)/(S \setminus \{e\})$$

and so

$$\dim \mathcal{B}(U/S) \geq \dim \mathcal{B}(U)/(S \setminus \{e\}) \geq \dim \mathcal{B}(U) - |S| + 1$$

which is a contradiction. □

2.4 Tools from Commutative Algebra

Let F be an arbitrary field and U be an (E, F)-space. Associate algebraically independent indeterminates $X \stackrel{\text{def}}{=} \{x_e \mid e \in E\}$ with the elements of E. Let $I(U)$ be the ideal generated by the set of linear polynomials $\{\sum_{e\in E} \mathbf{v}_e x_e \mid \mathbf{v} \in U\}$. Then the ring $R_U \stackrel{\text{def}}{=} F[X]/I(U)$ is again a polynomial ring in $|E| - \operatorname{rk}(\mathcal{M}(U))$ variables. This can be seen in the following way: Let $B \subseteq E$ be a basis of $\mathcal{M}(U)$. We can represent U by a block matrix $\mathbf{M} = [\mathbf{I}\ \mathbf{A}]$, where $\mathbf{I}$ is an identity matrix, the columns of $\mathbf{M}$ are indexed by the elements of E (the first $|B|$ columns are indexed by elements of B) and the rows are also indexed by the elements of B.

Proposition 2.4.1 *Let $Y \subseteq X$ be the set of algebraically independent indeterminates associated with the elements of $E \setminus B$. Then the kernel of the homomorphism of F-algebras $\varphi_B : F[X] \longrightarrow F[Y]$ defined by*

$$\varphi_B(x_e) \stackrel{\text{def}}{=} \begin{cases} x_e & \text{for } e \in E \setminus B, \\ -\sum\limits_{f\in E\setminus B} a_{e,f} x_f & \text{for } e \in B \end{cases}$$

is $I(U)$ and so it gives an isomorphism between R_U and $F[Y]$. □

Proof: Let S be the set of linear polynomials $p_e = x_e + \sum_{f \in E \setminus B} a_{e,f} x_f$ where $e \in B$ and let I be the ideal generated by S. Since the vectors formed by the coefficients of the polynomials p_e generate the space U we have that $I = I(U)$. From $\varphi_B(p_e) = 0$ we obtain that I is contained in the kernel of φ_B. To see the other inclusion let p be an arbitrary polynomial from the kernel of φ_B. The definition of φ_B shows that $x_e - \varphi_B(x_e) \in I$ for all $e \in E$ or in other words $x_e \equiv \varphi_B(x_e)$ modulo I. This means that $g \equiv \varphi_B(g)$ modulo I for all $g \in F[X]$. From $\varphi_B(p) = 0$ we obtain that $p \in I$. □

The fact that R_U is a polynomial ring itself will be frequently used throughout the chapter. One very important application of it is that R_U is an integral domain so we can do linear algebra over its quotient field. If U and W are both (E, F)-spaces with $U \subseteq W$ then $I(U) \subseteq I(W)$. The homomorphism $F[X]/I(U) \to F[X]/I(W)$ results in a natural ring homomorphism from R_U to R_W. We will need the following Lemma.

Lemma 2.4.2 *Let U and W be two (E, F)-spaces and let φ be the natural homomorphism from R_U to R_W. Assume that $\mathbf{A} = (a_{i,j})$ is an n by m matrix with entries from R_U and let $\varphi(\mathbf{A}) = (\varphi(a_{i,j}))$ be its image under the map φ. Then* $\operatorname{rk}(\varphi(\mathbf{A})) \leq \operatorname{rk}(\mathbf{A})$.

Proof: The rank of a matrix is the size of the largest $r \times r$ sub-matrix with nonzero determinant. If a sub-matrix of A has determinant d then the corresponding sub-matrix in $\varphi(\mathbf{A})$ has determinant $\varphi(d)$. This implies that singular sub-matrices of $\mathbf{A}$ are singular in $\varphi(\mathbf{A})$. □

Abusing the notation, we identify the variables x_e with their images under various algebra homomorphisms. To avoid confusion, we will always indicate in which algebra we are working. For example $x_e \in R_U$ denotes the image of x_e under the map $F[X] \to R_U$. The proofs of the following two lemmas are left to the reader.

Lemma 2.4.3 *Let U be an (E, F) space and let $S \subseteq E$ be an independent set in $\mathcal{M}(U)$. Then there is a unique isomorphism $\varphi : R_{U/S} \to R_U$ with $\varphi(x_e) = x_e$ for all $e \in E \setminus S$.* □

Lemma 2.4.4 *Let U be an (E, F) space and let S be an arbitrary subset of E. Moreover, let V_S denote the space formed by all $\mathbf{v} \in F^E$ with* $\operatorname{supp} \mathbf{v} \subseteq S$ *and let W denote the space spanned by U and V_S. Then there is a unique isomorphism $\varphi : R_{U \setminus S} \to R_W$ with $\varphi(x_e) = x_e$ for all $e \in E \setminus S$.* □

We will also make use of the following well-known theorem (see e.g [Lang 1971] Chapter V):

Theorem 2.4.5 *If F is a field then the polynomial ring $F[x_1, \ldots, x_k]$ is a unique factorization domain, i.e. every polynomial can be written as a product of irreducible polynomials, and such factorizations are unique up to multiplication of the factors with some nonzero scalars.* □

In order to estimate the efficiency of our methods, we will use the following easy lemma:

Lemma 2.4.6 ([Zippel 1979],[Schwartz 1980]) *For a nonzero polynomial $p \in K[x_1, \ldots, x_n]$ of degree d and $S \subseteq K$, the probability that p evaluates to 0 on a random element of S^n is at most $d/|S|$.* □

2.5 A Field Associated with Graphs

Let $G = (V, E)$ be a graph. We denote by GF(2) the field with two elements. Associate algebraically independent indeterminates $X = \{x_e\}_{e \in E}$ over GF(2) with the edges of G. Let I be the ideal in GF(2)$[X]$ generated by the sums $\sum_{e \in S} x_e$ for all cutsets S of G. In the language used in Section 5. we have that $I = I(U)$ where U is the cutset subspace of G. We define

$$F(G) \stackrel{\text{def}}{=} Q(\text{GF}(2)[X]/I),$$

where $Q(R)$ denotes the quotient field of ring R. The validity of this definition follows from Proposition 2.4.1 which shows that $R_U = \text{GF}(2)[X]/I$ is an integral domain. Another consequence of Proposition 2.4.1 is the following.

Proposition 2.5.1 *Let $T \subseteq E$ be a spanning forest of G and Y the set of the indeterminates associated with the edges not in T. We denote by $S(T, e)$, for $e \in T$, the cutset induced by the components of $T \setminus \{e\}$ in $G \setminus \{e\}$ (so we have $e \notin S(T, e)$). Then kernel of the* GF(2)*-algebra homomorphism*

$$f : \text{GF}(2)[X] \longrightarrow \text{GF}(2)[Y]$$

defined by

$$f(x_e) = \begin{cases} x_e \text{ for } e \notin T \\ \sum\limits_{d \in S(T,e)} x_d \text{ for } e \in T \end{cases}$$

is I and so it gives an algebra isomorphism between GF(2)$[X]/I$ *and $K[Y]$* □

The previous statement also implies that the field $F(G)$ is always isomorphic to a function field over GF(2) with corank (G) algebraically independent indeterminates. We could formulate our subsequent results using this explicite function f. This is practical for computing examples or constructing algorithms, but from a theoretic point of view the original definition has the advantage of not depending on the choice of a special tree. Another advantage is that proving theorems will be technically simpler if the indeterminates associated with the edges are treated homogeneously. The next lemma is an immediate consequence of lemma 2.4.3.

Lemma 2.5.2 *Let $G = (V, E)$ be a graph and $T \subseteq E$ be the edge set of a forest of G. Then the map $f : F(G/T) \longrightarrow F(G)$ defined by $f(x_e) = x_e$ is an isomorphism of fields.*

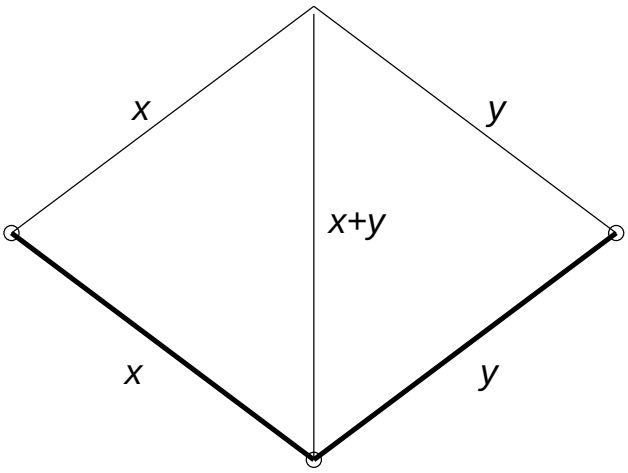
2.6 Representation of the Ear-Matroid

On the analogy of the incidence matrix of G, let $D(G) = (d_{ev})_{e\in E, v\in V}$ be an $E \times V$ matrix over $F(G)$ defined by

$$d_{ev} \stackrel{\text{def}}{=} \begin{cases} x_e \text{ for an incident pair } e, v \\ 0 \text{ otherwise} \end{cases}$$

We define the $V \times V$ matrix $T(G) = (t_{uv})$ over $F(G)$ by $t_{uv} \stackrel{\text{def}}{=} \sum x_e$ over all edges that connect u and v.

To illuminate the real nature of these matrices, we give a simple example using the above isomorphism given in Proposition 2.5.1. The graph G on the figure is K_4 with one edge deleted. The subgraph consisting of the two bold arcs is cotree $T' = E \setminus T$. We associate two algebraically independent indeterminates x and y with the bold edges. With each arc e of T we associate the sum of indeterminates associated with the arcs in the in the interesection of T' and the fundamental cut of e with respect to T.



Now we get the matrices $D(G)$ and $T(G)$:

$$D(G) = \begin{pmatrix} x & y & x+y & 0 & 0 \\ x & 0 & 0 & x & 0 \\ 0 & y & 0 & 0 & y \\ 0 & 0 & x+y & x & y \end{pmatrix}, T(G) = \begin{pmatrix} 0 & x & y & x+y \\ x & 0 & 0 & x \\ y & 0 & 0 & y \\ x+y & x & y & 0 \end{pmatrix}$$

Proposition 2.6.1 *For any graph G, let $T'(G) = (t'_{uv}) \stackrel{\text{def}}{=} D(G)D(G)^T$. Then $t'_{uv} = (t_{uv})^2$, where t_{uv} denotes the corresponding entry in $T(G)$.*

Proof: Using the identity $(a+b)^2 = a^2+b^2$, the equalities for the nondiagonal elements are clear. The diagonal elements of the product are zero because they are equal to $(\sum_{e \in S} x_e)^2 = 0$ for the star S of the corresponding vertex. □

Here is an easy consequence of this:

Proposition 2.6.2 *For each graph G holds:* $\operatorname{rk}(D(G)D(G)^T) = \operatorname{rk}(T(G))$

Proof: This follows from the last proposition, because in the expansion of a subdeterminant each term on the left-hand side is the square of a term in the expansion of the corresponding subdeterminant of the right side, and these terms are in one-to-one correspondence. So using the identity $(a+b)^2 = a^2+b^2$ we can see that each subdeterminant on the left-hand side equals the square of the corresponding subdeterminant on the right-hand side. □

It should be mentioned that $x_e \neq 0$ if and only if e is no bridge, so we can easily see that $\operatorname{rk}(D(G)) \leq \operatorname{rk}(G)$, with equality if G is 2-edge-connected. Denote by $T(G)[U]$ the symmetric submatrix of $T(G)$ induced by the rows and columns associated with the vertices of $U \subseteq V$.

Proposition 2.6.3 $\mathrm{rk}\,(T(G)) = \mathrm{rk}\,(T(G)[V \setminus \{v\}])$.

Proof: This can be seen using the fact that the sum of the rows and the sum of the columns of $T(G)$ are both 0. □

The aim of this section is to show that the graph invariant

$$\psi(G) \stackrel{\text{def}}{=} \mathrm{rk}\,(G) - \mathrm{rk}\,(T(G))$$

is nothing else than $\varphi(G)$ from section 2.1. Before we can prove it we need some preparation.

Denote by $\mathcal{R}(G)$ the row space of $D(G)$. Obviously, by 2.3.1:

Proposition 2.6.4 $\dim \mathcal{B}(\mathcal{R}(G)) = \psi(G)$. □

Lemma 2.6.5 *If G is a connected graph then*

$$\psi(G) = 0 \Leftrightarrow \varphi(G) = 0$$

Proof: $\Rightarrow$: Let v be an arbitrary vertex of G. By Proposition 2.6.3 and $\mathrm{rk}\,(G) = |V \setminus \{v\}|$, we have that $\psi(G) = 0$ implies $T(G)[V \setminus \{v\}]$ having a non-zero determinant. The expansion of this determinant is a sum of some products over all 1-factors of $G \setminus \{v\}$. Hence, for all $v \in V$, $G \setminus \{v\}$ has a 1-factor i.e. G is factor-critical. Using Theorem 2.1.1 we obtain that $\varphi(G) = 0$.

$\Leftarrow$: The statement is true for the one-vertex graph K_1. Using Theorem 2.1.1 and Corollary 2.1.2 it is enough to show that if G is connected, $\psi(G) = 0$ and G' is obtained from G by one of the operations described in Corollary 2.1.2 then $\psi(G') = 0$.

Case 1: Assume that G' is constructed by adding a new edge e between two vertices of G. Since G is a connected graph $\mathrm{rk}\,(G') = \mathrm{rk}\,(G)$. On the other hand it is clear, that $F(G') = F(G)(x_e)$, where x_e is an indeterminate over $F(G)$. Thus substituting $x_e = 0$ in $T(G')$ yields $T(G)$. This implies by Lemma 2.4.2 that $\mathrm{rk}\,(T(G')) \geq \mathrm{rk}\,(T(G)) = \mathrm{rk}\,(G)$. Using that $\mathrm{rk}\,(G) \geq \mathrm{rk}\,T(G')$, we obtain that $\psi(G') = \psi(G) = 0$.

Case 2: Assume that G' is constructed from G by double subdivision of some edge e. It is obvious that $\mathrm{rk}\,(G') = \mathrm{rk}\,(G) + 2$. We will show $\mathrm{rk}\,(T(G')) = \mathrm{rk}\,(T(G)) + 2$. It is clear, by the construction of our field $F(G)$, that the indeterminates associated with the edges involved in the subdivision are all

equal. We denote them by x_e. By Lemma 2.5.2, $F(G) = F(G')$, in the natural way. $T(G')$ can be written as follows:

$$T(G') = \begin{pmatrix} & & & & 0 & 0 \\ & T(G \setminus \{e\}) & & & \vdots & \vdots \\ & & & & 0 & x_e \\ & & & & x_e & 0 \\ 0 & \cdots & 0 & x_e & 0 & x_e \\ 0 & \cdots & x_e & 0 & x_e & 0 \end{pmatrix}$$

The four x_e entries not in the lower-right corner can be eliminated so that one obtains a block matrix with blocks $T(G)$ and $\left(\begin{smallmatrix} 0 & x_e \\ x_e & 0 \end{smallmatrix}\right)$, which implies $\operatorname{rk}(T(G')) = \operatorname{rk}(T(G)) + 2$. □

Theorem 2.6.6 *For any graph G*

$$\psi(G) = \varphi(G).$$

Proof: Since both φ and ψ are additive under taking disjoint union of graphs, we can assume that G is connected. We prove by induction on a that $\psi(G) = a \Leftrightarrow \varphi(G) = a$ holds for any connected graph G. According to Lemma 2.6.5, the statement is true for $a = 0$. Assume that it is true for all $a \leq k-1$ where $k > 0$. We show both implications for k:

$\Rightarrow$: Assume that $\psi(G) = k$. Using our induction hypothesis, we have that $\varphi(G) \geq k$. By Lemma 2.5.2 there is an edge $e \in \operatorname{supp} \mathcal{B}(\mathcal{R}(G))$. Using Lemma 2.3.2 we obtain that $\psi(G/e) = \psi(G) - 1 = k-1$. From the induction hypothesis it follows that $\varphi(G/e) = k-1$. Using that $\varphi(G/e) \geq \varphi(G) - 1$ we get that $\varphi(G) \leq k$.

$\Leftarrow$: Assume that $\varphi(G) = k$. Using our induction hypothesis we have that $\psi(G) \geq k$. Let e be an edge of an even ear of an optimal ear-decomposition D. It is clear that $\psi(G/e) = \psi(G) - 1 = k-1$. From the induction hypothesis it follows that $\psi(G/e) = k-1$. Lemma 2.3.2 implies that $\psi(G/e) \geq \psi(G) - 1$. Now $\psi(G) \leq k$ completes the proof. □

A consequence of this characterization of φ is a simple proof of

Corollary 2.6.7 ([Szigeti 1994]) *Let G be a graph such that $G \setminus v$ has a unique perfect matching containing no bridge of G, then G is factor-critical.*

Proof: The unique matching produces a unique nonzero product in the expansion of $\det T(G)[V \setminus \{v\}]$. A different combinatorial proof can be found in [Szigeti 1994]. □

It is more interesting that our results yield a representation of the ear-matroid $\mathcal{E}(G)$ of G. We emphasize that the verification of the matroid axioms for $\mathcal{E}(G)$ by combinatorial means is not trivial at all (see [Szigeti 1996]). Now this follows from our foregoing results and moreover we obtain a representation of $\mathcal{E}(G)$.

Theorem 2.6.8 *Let $G = (V, E)$ be a graph and $M \subseteq 2^E$ the family of the subsets $S \subseteq E$ for which $\varphi(G/S) = \varphi(G) - |S|$. This family is identical to the family of the independent sets of $\mathcal{M}(\mathcal{B}(\mathcal{R}(G)))$. Moreover for each set $S \in M$ $\mathcal{E}(G/M) = \mathcal{E}(G)/M$.*

The proof is an immediate consequence of Theorem 2.3.3, Lemma 2.5.2 and 2.6.4 and Theorem 2.6.6. □

Now given our algebraic machinery we can prove the following corollaries:

Corollary 2.6.9 *Let G be a graph. A dependent set in the cycle or in the cocycle matroid of G is also dependent in $\mathcal{E}(G)$.* □

Corollary 2.6.10 *Let G be a 2-edge-connected graph. G is factor-critical if and only if, for each basis F of the cutset space, there is a partition of F into pairs so that, for each pair $\{S_1, S_2\}$ of the partiton, $S_1 \cap S_2$ is not a cutset of G.*

Proof: For the *if* part of the statement one can choose any basis of the cutset space consisting of $|V| - 1$ stars and obtains the factor-criticality immediately. For the other direction, let $\mathbf{D}'$ be the generating matrix of $\mathcal{R}(G)$ constructed on the analogy of $\mathbf{D}$, but using the basis F for the rows. Then

$$0 = \varphi(G) = \dim \mathcal{B}(\mathcal{R}(G)) = \dim \mathcal{B}(\mathcal{R}(\mathbf{D}')) = \mathrm{rk}\,(G) - \mathrm{rk}\,(\mathbf{D}'\mathbf{D}'^T).$$

So $\det \mathbf{D}'\mathbf{D}'^T \neq 0$ and one can argue as in the proof of Theorem 2.6.6. □

Corollary 2.6.11 *Let $G = (V, E)$ be a factor-critical graph and $T \subseteq E$ a tree of G. Construct a graph G' on the vertex set T by connecting e_1 and $e_2 \in T$ by an edge iff their fundamental cuts intersect. Then G' has a perfect matching.*

Proof: This is an obvious special case of the last corollary. Clearly, if the intersection of two fundamental cuts is not a cut, then it is nonempty by definition. □

The notion of φ-coveredness was introduced in [Szigeti 2001]. A graph G is called φ-covered iff $\mathcal{E}(G)$ has no loop. Z. Szigeti demonstrated, by generalizing

several theorems on matching-covered graphs to φ-covered graphs, that this concept is a natural generalization of matching-coveredness. This motivates

Corollary 2.6.12 *Each bipartite graph is φ-covered.*

Proof: Clearly the vector $(x_e)_{e\in E}$ is in $\mathcal{B}(\mathcal{R}(G))$, hence $\operatorname{supp}\mathcal{B}(\mathcal{R}(G)) = E$. □

Corollary 2.6.13 *Let $G = (V, E)$ be a factor-critical graph. Define $G' = (V, E')$ by*

$$E' \stackrel{\text{def}}{=} \left\{ \{u, v\} \in \binom{V}{2} \mid \exists z \in V \setminus \{u, v\} : \{u, z\} \in E \text{ and } \{v, z\} \in E \right\}.$$

Then G' is factor-critical.

Proof: It is straightforward that all $(V \setminus v) \times (V \setminus v)$ subdeterminants of $T(G)$ are equal. $T(G)T(G)$ is a alternating matrix. If an entry of $T(G)T(G)$ is nonzero, then the corresponding vertices must have a common neighbour in G. From the Cauchy-Binet formula and since $|V|$ is odd, the determinant of $T(G)T(G)[V \setminus v]$ is $\det T(G)[V \setminus v]^2$, which is nonzero. Therefore, every induced subgraph $G'[V \setminus v]$ has a perfect matching. □

2.7 Symplectification of Spaces

Now we will generalize the result of the last section to arbitrary matroids representable over some field of characteristic 2. For this reason, we will start with extending the notion introduced in Section 2.5.

Let F be a field of characteristic 2 and U be an (E, F)-space. Let R_U be the ring described in Section 2.5 and let F_U be its quotient field. This field is isomorphic to a rational function field over F in $\operatorname{corank}(U)$ indeterminates. As in Section 2.5, we identify the variables x_e with their images under the map $F[X] \to F[X]/I(U) = R_U \subseteq F_U$. Let $\mathbf{A}$ be a generating matrix of U, and let $\mathbf{A}_e$ denote its column corresponding to e.

Let $\mathbf{D}$ be the $E \times E$ diagonal matrix with $x_e \in F_U$ at the row indexed by e. We define the *symplectification* $\mathrm{S}(U)$ of U to be the row space of the matrix $\mathbf{AD}$. Clearly, $\mathrm{S}(U)$ is the row space of the matrix $\mathbf{G}$ which is obtained from $\mathbf{A}$ by multiplying each column $\mathbf{A}_e$ by the element $x_e \in F_U$. Note that the space $\mathrm{S}(U)$ is an (E, F_U)-space and that it does not depend on the choice of the generating matrix $\mathbf{A}$ (it only depends on the space U). As the name suggests, the symplectification of the space U has the following property:

Lemma 2.7.1 *The induced scalar product on the space* $\mathrm{S}(U)$ *is symplectic, i.e.* $\langle \mathbf{v}, \mathbf{v} \rangle = 0$ *for all* $\mathbf{v} \in \mathrm{S}(U)$. □

The good thing about symplectification is that it does not really alter the matroid structure on E. To be more precise, if $\mathcal{M}(U)$ has no bridge, then $\mathcal{M}(\mathrm{S}(U)) = \mathcal{M}(U)$ since we just multiplied the columns of G by some nonzero scalars. In fact, the only difference between $\mathcal{M}(U)$ and $\mathcal{M}(\mathrm{S}(U))$ is that the bridges of $\mathcal{M}(U)$ are replaced by loops. It will prove crucial that $\mathrm{S}(U/e)$ and $\mathrm{S}(U)/e$ are basically indentical up to the natural isomorphism (see Lemma 2.4.3).

Lemma 2.7.2 *The symplectification* $\mathrm{S}(U)$ *of a bicycle free symplectic* (E, F)*-space* U *is a bicycle free symplectic space.*

Proof: The symplectification of any space is symplectic, so we have to show that $\mathrm{S}(U)$ is bicycle free. Let $\boldsymbol{\chi}_E$ be the everywhere 1 vector in F^E. Since U is symplectic, it is contained in the space $W = \boldsymbol{\chi}_E^{\perp}$. It is easy to see that R_W is a polynomial ring in one variable x and that the natural homomorphism $\varphi : R_U \to R_W$ maps x_e to x for all $e \in E$. We obtain that $\varphi(\det(\mathbf{A}\mathbf{D}\mathbf{D}^T\mathbf{A}^T)) = x^2 \det(\mathbf{A}\mathbf{A}^T)$. This means that $\det(\mathbf{A}\mathbf{A}^T) \neq 0$ implies $\det(\mathbf{A}\mathbf{D}\mathbf{D}^T\mathbf{A}^T) \neq 0$. Lemma 2.3.1 completes the proof. □

2.8 A △-Matroid Defined by Ear-Decompositions

A Δ-*matroid* is a nonempty set-system $\mathcal{F} \subseteq 2^E$ satisfying the symmetric exchange axiom: For $F_1, F_2 \in \mathcal{F}$ and $e \in F_1 \triangle F_2$, there exists $f \in F_1 \triangle F_2$ such that $F_1 \triangle \{e, f\} \in \mathcal{F}$. The members of set-system $\mathcal{F}$ are called *feasible* sets of the Δ-matroid. A Δ-matroid is *even* if all feasible sets are of the same parity. If F is a subset of E then $F \triangle \mathcal{F} \stackrel{\text{def}}{=} \{F \triangle F' \mid F' \in \mathcal{F}\}$ is called the *twist* of $\mathcal{F}$ by F, and it also satisfies the symmetric exchange axiom. While matrices give rise to matroids, representable Δ-matroids arise from symmetric or skew-symmetric matrices:

Theorem 2.8.1 ([Bouchet 1988]) *If* $\mathbf{A}$ *is a symmetric or skew-symmetric* $E \times E$ *matrix, then* $\mathcal{F}(\mathbf{A}) \stackrel{\text{def}}{=} \{F \subseteq E \mid \mathbf{A}[F, F] \text{ is regular}\}$ *form the family of feasible sets of a* Δ*-matroid and* $\mathbf{A}$ *is called a (skew-)symmetric representation of* $\mathcal{F}(\mathbf{A})$. □

A Δ-matroid $\mathcal{F}$ is called *representable* if some twist of it arises from a symmetric or skew-symmetric matrix in the above way.

We will also need the concept of subdivision of edges of an (E, F)-space U. Let $e \in E$ be an arbitrary element. We introduce a new edge e' and we denote by E' the set $E \cup \{e'\}$. Space U is naturally embedded into $F^{E'}$ by extending every vector $\mathbf{u} \in U$ with a 0 coordinate corresponding to e'. Let U' denote the (E', F) space which is obtained from U by switching the coordinates corresponding to e and e'. We define the (E', F)-space $U \div e$ resulting from the *subdivision* of edge e as the space spanned by U and U'. One can easily see (for example, by looking at the generating matrices) that the subdivision of an edge increases the dimension of the space by exactly one.

Since the subdivisions of distinct edges commute, $U \div S$ can be defined to be the subsequent subdivision of all edges in $S \subseteq E$. It can easily be checked that $\dim(U \div S) = \dim(U) + |S|$. Another simple but useful fact is that if S and T are disjoint subsets of E, then $(U \div S)/T = (U/T) \div S$.

Later on, we will need the following simple facts:

Lemma 2.8.2 *Let U be a symplectic (E, F)-space and assume that* $\operatorname{supp}(\mathbf{v}) \subseteq \{e, f\}$ *with $\mathbf{v} \in U$ and $e, f \in E$. Then $\mathbf{v} = c(\boldsymbol{\chi}_e + \boldsymbol{\chi}_f)$ for some scalar $c \in F$. In particular e and f are in the same series class of $\mathcal{M}(U)$ if and only if $\boldsymbol{\chi}_e + \boldsymbol{\chi}_f \in U$.*

Proof: Since $\operatorname{supp}(\mathbf{v}) \subseteq \{e, f\}$ we have that $\mathbf{v} = a\boldsymbol{\chi}_e + b\boldsymbol{\chi}_f$ for some scalars $a, b \in F$. Using that the space is symplectic we obtain that $0 = \langle \mathbf{v}, \mathbf{v} \rangle = a^2 + b^2 = (a + b)^2$. It follows that $a + b = 0$ and so $a = b$. The second statement follows from Lemma 2.2.2. □

Proposition 2.8.3 *, Let U be a symplectic space and e, f two series elements of $\mathcal{M}(U)$, then $(U/e) \div f = U$ if we identify the newly created edge f' and e.*

Proof: Since e and f are series, $\boldsymbol{\chi}_e + \boldsymbol{\chi}_f \in U$. The dimension of U and $U/e \div f$ coincide, so we must only show that $U/e \div f$ is contained in U. Take an arbitrary vector of $\mathbf{v} \in U/e$. Since switching f and $e = f'$ is equivalent to adding $\boldsymbol{\chi}_e + \boldsymbol{\chi}_f$ to $\mathbf{v}$, therefore the inclusion is clear. □

The straightforward proof of the following Lemma is left to the reader.

Lemma 2.8.4 *Let S be an independent set of the matroid of (E, F)-space U. Then U has a minimal generating matrix of the form* $\begin{array}{c} \\ S \\ E \setminus S \end{array} \begin{array}{c} \begin{array}{cc} S & E \setminus S \end{array} \\ \begin{pmatrix} \mathbf{I} & \mathbf{A} \\ 0 & \mathbf{B} \end{pmatrix} \end{array}$, *where $\mathbf{B}$ is a minimal generating matrix of U/S. In this case,*

$$\begin{array}{c} \\ S' \\ S \\ E \setminus S \end{array} \begin{array}{c} \begin{array}{ccc} S' & S & E \setminus S \end{array} \\ \begin{pmatrix} \mathbf{I} & -\mathbf{I} & 0 \\ 0 & \mathbf{I} & \mathbf{A} \\ 0 & 0 & \mathbf{B} \end{pmatrix} \end{array}$$ *is a minimal generating matrix of* $U \div S$. *(*$\mathbf{I}$ *always denotes an identity matrix of suitable size.)*

Using this representation we can show

Lemma 2.8.5 *Let* F *be a field of characteristic 2 and let* S *be an independent set of the matroid of* (E, F)*-space* U. *Then* $\beta(U \div S) = \beta(U/S)$. *(*$\beta(V)$ *denotes the dimension of the bicycle-space of* V *as defined in Section 2.1)*

Proof: Take a minimal representation $\mathbf{M}$ of U according to Lemma 2.8.4. By Lemma 2.3.1 we have

$$\beta(U \div S) = \dim(U \div S) - \mathrm{rk}(\mathbf{M}\mathbf{M}^T) =$$

$$= 2|S| + \dim(U/S) - \mathrm{rk} \begin{pmatrix} 0 & \mathbf{I} & 0 \\ \mathbf{I} & \mathbf{I} + \mathbf{A}\mathbf{A}^T & \mathbf{A}\mathbf{B}^T \\ 0 & \mathbf{B}\mathbf{A}^T & \mathbf{B}\mathbf{B}^T \end{pmatrix},$$

where $\mathbf{I}$ is always an identity matrix of suitable size. The rightmost matrix can be transformed to the form $\begin{pmatrix} 0 & \mathbf{I} & 0 \\ \mathbf{I} & 0 & 0 \\ 0 & 0 & \mathbf{B}\mathbf{B}^T \end{pmatrix}$ by using rank-preserving column- and row-operations.

Since B is a minimal generating matrix of U/S we obtain $\beta(U \div S) = 2|S| - 2|S| + \dim(U/S) - \mathrm{rk}(BB^T) = \beta(U/S)$. □

Lemma 2.8.6 *Let* F *be a field of characteristic 2 and let* S *and* T *be two independent sets of the matroid* $\mathcal{M}(U)$ *of* (E, F)*-space* U. *Then* $\beta(U/T) = \beta((U \div S)/(T \triangle S))$

Proof: Since $T \cap S$ is independent in $\mathcal{M}(U/(T \setminus S))$, Lemma 2.8.5 implies that

$$\beta(U/T) = \beta(U/(T \setminus S)/(T \cap S)) =$$

$$\beta((U/(T \setminus S)) \div (T \cap S)) = \beta((U \div S)/(S \triangle T)).\square$$

Let U be a bicycle free (E, F)-space. Then F^E is the direct sum of U and $U^\perp$, so every vector $\mathbf{v} \in F^E$ can be uniquely written as $\mathbf{v} = \mathbf{v}_1 + \mathbf{v}_2$, where $\mathbf{v}_1 \in U$ and $\mathbf{v}_2 \in U^\perp$. This gives rise to the unique orthogonal projections $P_U : F^E \longrightarrow U$ and $P_{U^\perp} : F^E \longrightarrow U^\perp$ which are represented by the symmetric idempotent matrices $\mathbf{P}_U$ and $\mathbf{P}_{U^\perp}$.

Proposition 2.8.7 *Let U be an (E,F)-space. Let $\mathbf{P}_U$ be the matrix of the orthogonal projection onto U. The following two statements for an independent subset $S \subseteq E$ are equivalent:*

(1) U/S is bicycle free.

(2) The symmetric submatrix $\mathbf{P}_U[S,S]$ is nonsingular.

Proof: We use that U/S is naturally embedded into K^E and that this embedding is scalar product preserving. Let V_S be the subspace of all vectors from K^E whose components outside S are zero. The matrix $\mathbf{P}_U[S,S]$ has full rank if and only if $\mathbf{P}_U(V_S) \setminus (E \setminus S)$ is full K^S. It means that $\mathbf{P}_U[S,S]$ has full rank if and only if $\mathbf{P}_U(V_S) + U/S = U$. First assume that $W \stackrel{\text{def}}{=} \mathbf{P}_U(V_S) + U/S$ is strictly smaller than U. In this case, there is a nonzero vector $\mathbf{v}$ in $W^\perp \cap U$. By $\mathbf{v} \perp \mathbf{P}_U(V_S)$, we obtain that $\mathbf{v} \perp V_S$ and so $\mathbf{v} \in U/S$. From $\mathbf{v} \perp U/S$ we get that U/S is not bicycle-free. Now, assume that $W = U$. To prove that U/S is bicycle-free, let $\mathbf{v} \in (U/E) \cap (U/E)^\perp$. Since $\mathbf{v} \in U/E$, we have that $\mathbf{v} \perp \mathbf{P}_U(V_S)$. From the assumption $W = U$ follows that $\mathbf{v} \in U^\perp$ and so $\mathbf{v} = 0$. □

Theorem 2.8.8 *Let F be a field of characteristic 2 and U an (E,F)-space. Let $\mathcal{F}$ be the family of those subsets of E which are independent in $\mathcal{M}(U)$ and whose contraction results in a bicycle-free space. Then $\mathcal{F}$ is the family of feasible sets of a representable Δ-matroid.*

Proof: We use Theorem 2.8.1 and construct a symmetric representation of $\mathcal{F}$. Let B be a basis of $\mathcal{M}(\mathcal{B}(U))$. Then U/B and so $U \div B$ are bicycle-free by Theorem 2.3.3 and Lemma 2.8.5. Let $\mathbf{P}$ be the orthogonal projection matrix to the $(E \cup B', F)$-space $U \div B$. Let $T \subseteq E$ be independent in $\mathcal{M}(U)$. Lemma 2.8.6 yields $\beta(U/T) = \beta((U \div B)/(S \triangle T))$, so by Proposition 2.8.7 we get that $\mathcal{F} \triangle B$ is represented by $\mathbf{P}[E,E]$. □

2.9 The Characterization Theorem

In this section, we assume that F is a field of characteristic 2. Here, we will prove the central result of this chapter: the characterization of factor-critical matroids representable over some field of characteristic 2. Before stating the main theorems we will prove some lemmas.

Lemma 2.9.1 *Let $\mathcal{F}$ be a family of matroids satisfying the following properties:*

(1) Every matroid in $\mathcal{F}$ is bridgeless.

(2) If $M \in \mathcal{F}$ and $\mathrm{rk}\,(M) > 0$ then either there is an $e \in E(M)$ such that $M \setminus e \in \mathcal{F}$ or there is a series class of M consisting of at least three elements e, f, g such that $M/\{e, f\} \in \mathcal{F}$.

Then every element of $\mathcal{F}$ is factor-critical

Proof: We go by induction on $|E(M)| + \mathrm{rk}\,(M)$. Let M be a matroid from $\mathcal{F}$ and assume that the statement is true for all $M' \in \mathcal{F}$ with $|E(M')| + \mathrm{rk}\,(M') < |E(M)| + \mathrm{rk}\,(M)$. There are two possible cases

1.: Let $e \in E(M)$ such that $M \setminus e$ is factor-critical, and let $\mathcal{P}$ be a partition of $E(M \setminus e)$ described in Lemma 2.1.3. Since e is contained in some circuit C we obtain that e is a loop (circuit with one element) in $M/(M \setminus e)$. By extending $\mathcal{P}$ with $\{e\}$ as a last element we obtain a partition of $E(M)$ which proves that M is factor-critical.

2.: If there is no such edge, then there is a series class of M with at least three edges $e, f, g \in E(M)$, such that $M/\{e, f\}$ is factor-critical. Let $(C_1, \ldots, C_k)$ be an odd ear-decomposition of a block of $M/\{e, f\}$ which contains a circuit C_i containing g. It is easy to check that $(C_1, \ldots, C_i \cup \{e, f\}, \ldots, C_k)$ is an odd ear-decomposition of the block of M containing e,f and g. The existence of an odd ear-decomposition of the other blocks of M follows immediately from the induction hypothesis. □

Lemma 2.9.2 *Let U be an (E, F)-space. The induced scalar product on U is a nondegenerate symplectic form if and only if there is a generating matrix of U, which is an alternating projection.*

Proof: If the induced scalar product is a nondegenerate symplectic form on U, then U is bicycle free and so the matrix $\mathbf{P}_U$ of the orthogonal projection to U is a alternating projection matrix representing U. If U is represented by a alternating projection matrix P, then the induced scalar product on U is clearly symplectic. If we assume that U is not bicycle-free, then there is a vector $\mathbf{v} \in F^E \setminus \mathcal{B}(U)^\perp = F^E \setminus (U + U^\perp)$. For an arbitrary vector $\mathbf{w} \in U$,

$$\langle \mathbf{v} + P\mathbf{v}, \mathbf{w} \rangle = \langle \mathbf{v}, \mathbf{w} \rangle + \langle P\mathbf{v}, \mathbf{w} \rangle = \langle \mathbf{v}, \mathbf{w} \rangle + \langle \mathbf{v}, P\mathbf{w} \rangle = \langle \mathbf{v}, \mathbf{w} \rangle + \langle \mathbf{v}, \mathbf{w} \rangle = 0$$

and therefore $\mathbf{v} = P\mathbf{v} + (\mathbf{v} + P\mathbf{v}) \in U + U^\perp$, contradicting the choice of $\mathbf{v}$. □

Lemma 2.9.3 *Let U be a symplectic space, e, f series elements of U, then:*

$$\beta(U) = \beta(U/\{e, f\}).$$

In particular, if U is a symplectic bicycle-free space, then the so is $U/\{e, f\}$.

Proof:

$$\beta(U) = \beta(U/e \div f) = \beta(U/\{e, f\}).$$

Where the first equality follows from Proposition 2.8.3, the second one follows from Lemma 2.8.6. □

Lemma 2.9.4 *Let $\mathrm{S}(U)$ be the symplectification of a bridgeless (E, F)-space U, B a basis of U, $\mathbf{A}$ a minimal generating matrix of U and $F(\{y_e\}_{e\in E\setminus B})$ the ground field of $\mathrm{S}(U)$ according to the isomorphism of Proposition 2.4.1. Let $\mathbf{D}$ be the diagonal matrix such that $\mathbf{AD}$ is a minimal generating matrix of $\mathrm{S}(U)$. Assume that squaring is an isomorphism on F (i.e. each element $c \in F$ has a square root $\sqrt{c}$). Then $\det(\mathbf{ADD}^T\mathbf{A}^T) = p^4$, where $p \in F[\{y_e\}_{e\in E\setminus B}]$ is of total degree at most $\dim(U)/2$.*

Proof: For an alternating $V \times V$ matrix $\mathbf{X} = (x_{uv})_{u,v\in V}$ over a field of charcteristic 2, we denote by $\mathrm{Pf}(\mathbf{X}) \stackrel{\text{def}}{=} \sum_M \prod_{\{v,w\}\in M} x_{vw}$ the Pfaffian of $\mathbf{X}$, where the summation is over all perfect matchings M of the complete graph on V. It is well-known (see e.g [Godsil 1993]) that $\mathrm{Pf}(\mathbf{X})^2 = \det(\mathbf{X})$. Since $\mathbf{ADD}^T\mathbf{A}$ is a alternating matrix, $\det(\mathbf{ADD}^T\mathbf{A}^T)$ can be expressed as $\mathrm{Pf}(\mathbf{ADD}^T\mathbf{A}^T)^2$, where $\mathrm{Pf}(\mathbf{ADD}^T\mathbf{A}^T)$ is a polynomial in the entries of $\mathbf{ADD}^T\mathbf{A}^T$ of total degree $\dim(U)/2$. Since the entries of $\mathbf{ADD}^T\mathbf{A}^T$ are linear polynomials in the squares of the indeterminates and squaring is an endomorphism of the ring in question, we obtain the statement by setting $p \stackrel{\text{def}}{=} \sqrt{\mathrm{Pf}(\mathbf{ADD}^T\mathbf{A}^T)}$. □

Theorem 2.9.5 *Let M be a matroid representable over a field of characteristic 2. Then the following statements are equivalent:*

(1) M is factor-critical.

(2) M is bridgeless and the symplectification of each representation of M by an (E, F)-space U over a field F of characteristic 2 is bicycle free.

(3) M is bridgeless and the symplectification of some representation of M by an (E, F)-space U over a field F of characteristic 2 is bicycle free.

(4) M is representable by a space on which the induced scalar product is a nondegenerate symplectic form.

(5) M is representable by an alternating projection matrix.

Proof:
(1)$\Longrightarrow$(2): If a matroid has an ear-decomposition then every edge is contained in some circuit and so it is bridgeless. We fix an odd ear-decomposition of M and a representation U of M. Clearly, the symplectification of any space is symplectic, so our objective is to prove that $\mathrm{S}(U)$ is bicycle free. Let $\{x_e\}_{e\in E}$ be the indeterminates defined by the symplectification. We proceed by induction on $|E|$. We can assume that the implication holds for any matroid M' with $|E(M')| < |E|$ and have two cases:

Case 1: *The lobe of the last ear is a single edge e.*

In this case, since e is no bridge, $\mathrm{rk}\,(U \setminus e) = \mathrm{rk}\,(U)$. Let $\mathbf{A}$ be a minimal generating matrix of U and $\mathbf{A}'$ the matrix obtained from $\mathbf{A}$ by deleting the e-th column. Clearly $\mathbf{A}'$ is a minimal generating matrix of $U \setminus e$. Let $\{y_f\}_{f\in E\setminus e}$ be the indeterminates occuring in the definition of $S(U \setminus e)$. One can check that the substitution $x_f = \begin{cases} y_f \text{ if } f\neq e, \\ 0 \text{ if } f=e \end{cases}$ preserves the dependence between the indeterminates $\{x_f\}_{f\in E}$. This means that $\det(\mathbf{A}'(\mathbf{A}')^T) \neq 0$ implies $\det(\mathbf{A}\mathbf{A}^T) \neq 0$. Since the first inequality holds by the induction assumption and Lemma 2.3.1, so does the latter one, i.e. $\mathrm{S}(U)$ is bicycle free.

Case 2: *The lobe of the last ear contains at least two independent edges e and f.*

Contracting the last ear by $\{e, f\}$ shows that $M/\{e, f\}$ admits an odd ear decomposition. By Lemma 2.9.3 and since the symplectification commutes with contraction, we get:

$$\beta(\mathrm{S}(U)) = \beta(\mathrm{S}(U)/\{e, f\}) = \beta(\mathrm{S}(U/\{e, f\})) = 0,$$

where the last equality follows from the induction assumption. Thus, we obtain by Lemma 2.3.1 that $\mathrm{S}(U)$ is bicycle free.

(2)$\Longrightarrow$(3): Clear.

(3)$\Longrightarrow$(4): Let U be an (E, F)-space which is a non degenerate (bicycle free) symplectic representation of M. The property that M is bridgeless guarantees that the symplectification of U is a representation of M. Using Lemma 2.3.1 and Lemma 2.7.2 we obtain that statement (4) holds for $\mathrm{S}(U)$.

(4)$\Longleftrightarrow$(5): Lemma 2.9.2.

(4)$\Longrightarrow$(1): Let $\mathcal{F}$ be the family of matroids satisfying (4). We show the implication by checking the conditions of Lemma 2.9.1 for $\mathcal{F}$. Let M be a matroid in $\mathcal{F}$. Fix an (E,F)-space U which is a representation of M satisfying (4) such that squaring on the ground field F is an isomorphism. (This can always be achieved by tensoring U with the algebraic closure of F, which does not change the involved matrices at all) The first condition of Lemma 2.9.1 follows from Lemma 2.2.1 because a symplectic (E,F)-space can't contain the characteristic vector of en edge e. We have two cases:

Case 1: *There is an edge e such that $\mathrm{S}(U \setminus e)$ is bicycle free.*

If $M \setminus e$ is bridgeless then $\mathrm{S}(U \setminus e)$ represents $M \setminus e$ and so $M \setminus e \in \mathcal{F}$. If there are two bridges f, g in $M \setminus e$ then e,f and g are in the same series class of M and by Lemma 2.9.3 we have that $M/\{e,f\} \in \mathcal{F}$. Finally we exclude the case that there is exactly one bridge f in $M \setminus e$. To see this, recall that a symplectic, bicycle free space has to be even dimensional. In particular U is even dimensional. Assume that $\mathrm{supp}\,(\mathbf{v}) \subseteq \{e,f\}$ for some vector $\mathbf{v} \in U$. Lemma 2.8.2 shows that $\mathbf{v} = c(\boldsymbol{\chi}_e + \boldsymbol{\chi}_f)$ for some scalar $c \in F$. Since e and f are in the same series class, we have that $\boldsymbol{\chi}_e + \boldsymbol{\chi}_f \in U$. These two facts together imply that $\dim(U \setminus \{e,f\}) = \dim(U) - 1$. Thus $U \setminus \{e,f\}$ is an odd dimensional bridgeless space. Now it is easy to see that $\dim(\mathrm{S}(U \setminus e)) = \dim(U \setminus \{e,f\})$ which contradicts the fact that $\mathrm{S}(U \setminus e)$ is bicycle free.

Case 2: *The bicycle space of $\mathrm{S}(U \setminus e)$ is non-zero for all $e \in E$.*

Let $B \subseteq E$ be a basis of M and $\{x_e\}_{e \in E}$ be the elements in F_U occurring in the definition of $\mathrm{S}(U)$ (The diagonal entries of $\mathbf{D}$). By Lemma 2.7.2 $\mathrm{S}(U)$ is also bicycle free. We have a unique generating matrix $\mathbf{A}$ of U of the form

$$\begin{array}{c} \quad B \quad E \setminus B \\ B \begin{pmatrix} \mathbf{I} & \mathbf{C} \end{pmatrix}. \end{array}$$

Set

$$\gamma(B,e) \stackrel{\text{def}}{=} \begin{cases} \sum\limits_{f \in E \setminus B} \mathbf{A}[e,f] x_f & \text{for } e \in B, \\ x_e & \text{otherwise.} \end{cases}$$

By Proposition 2.4.1, the map $\varphi : x_e \to \gamma(B,e)$ induces an isomorphism of rings between F_U and $F[\{x_e\}_{e \in E \setminus B}]$, where the indeterminates x_e for $e \in E \setminus B$ are algebraically independent. Let $\mathbf{A}_{E \setminus e}$ be the matrix obtained from

$\mathbf{A}$ by deleting the e-th column. Then $\mathbf{A}' \stackrel{\text{def}}{=} \mathbf{A}_{E\setminus e}\mathbf{D}[E \setminus e, E \setminus e]$ is a minimal generating matrix of $S(U \setminus e)$ with entries from the polynomial ring

$$F[\{x_f\}_{f\in E\setminus B}]/(\gamma(B,e)),$$

where $(\gamma(B,e))$ denotes the ideal generated by the $\gamma(B,E)$ in $F[\{x_f\}_{f\in E\setminus B}]$. If we compute $\mathbf{A}'(\mathbf{A}')^T$, we formally obtain $\mathbf{ADD}^T\mathbf{A}^T$; we have only changed our ground field. To be more specific, we see, that $\mathbf{A}'(\mathbf{A}')^T$ is singular if and only if $\det(\mathbf{ADD}^T\mathbf{A}^T)$ is in the ideal generated by $\gamma(B,e)$ i.e. $\gamma(B,e)$ divides $\det(\mathbf{ADD}^T\mathbf{A}^T)$. The degree of the polynomial $\det(\mathbf{ADD}^T\mathbf{A}^T)$ is $2|B|$ and according to Lemma 2.9.4 it is the 4-th power of some polynomial of degree $|B|/2$. Since the polynomials $\gamma(B,e)$ are linear (and thus irreducible) we obtain that, up to multiplication with scalars, there are at most $|B|/2$ different values of $\gamma(B,e)$. If for $e,f \in B$ holds $\gamma(B,e) = c\gamma(B,f)$, then $\boldsymbol{\chi}_e + c\boldsymbol{\chi}_f \in U$, i.e. e and f are in the same series class of M. Lemma 2.8.2 shows that in this case $\boldsymbol{\chi}_e + \boldsymbol{\chi}_e \in U$. Using the pigeonhole principle, we can deduce that either there is a series class with at least three elements (M satisfies the conditions of Lemma 2.9.1) or B can be partitioned into series classes of size two. Since every edge is contained in some basis B of M, we can assume that the whole edge set can be partitioned into series classes each of them having size at least 2. If there is a series class with at least three element we are done. It remains to rule out the case when E is partitioned into series classes with two elements. Because of the symplecticity of U, the characteristic vectors of these classes are all in U, and so the same is true for their sum $\boldsymbol{\chi}_E$. Using again that U is a symplectic space and that F has characteristic 2 one can see easily that $\boldsymbol{\chi}_E$ is also in $U^\perp$, contradicting the nondegeneracy of the scalar product on U. □

Theorem 2.9.6 *Let U be a bridgeless (E,F)-space. Then $\varphi(\mathcal{M}(U))$ is the dimension of the bicycle space of* $\mathrm{S}(U)$.

Proof: Let us introduce the invariants $\psi(W) \stackrel{\text{def}}{=} \beta(\mathrm{S}(W))$ and $\varphi(W) \stackrel{\text{def}}{=} \varphi(\mathcal{M}(W))$ for any (E,F)-space W. Let α be any of the two invariants φ and ψ. One can check easily that α has the following two properties:

(1) $\alpha(W/e) \geq \alpha(W) - 1$

(2) If $\alpha(W) \neq 0$ then there exists $e \in E$ with $\alpha(W/e) = \alpha(W) - 1$.

By Theorem 2.9.5 we know that $\varphi(U) = 0$ if and only if $\psi(U) = 0$. Now assume that $\varphi(U) > \psi(U)$. According to the above properties, there is an

edge set $S \subseteq E$ such that $|S| = \psi(U)$ and $\psi(U/S) = 0$. Using again the above properties we obtain that $\varphi(U/S) \geq \varphi(U) - |S| = \varphi(U) - \psi(U)$ which is a contradiction. The case $\psi(U) > \varphi(U)$ can be excluded exactly in the same way. □

Theorem 2.9.7 *Let U be a bridgeless (E, F)-space where F is either a finite field of characteristic 2 or the field of fractions of some polynomial ring over a finite field of characteristic 2. Let $\mathbf{M}$ be a generating matrix of U. Then the value of $\varphi(\mathcal{M}(U))$ can be computed in randomized polynomial time in terms of the input $\mathbf{M}$.*

Proof: Using Theorem 2.9.6, Lemma 2.3.1 we obtain that the computation of φ can be reduced to the computation of the rank of the matrix $\mathbf{MDD}^T\mathbf{M}^T$ where D is the diagonal matrix occurring in the definition of the symplectification of U. Applying Proposition 2.4.1, the matrix $\mathbf{L} \stackrel{\text{def}}{=} \mathbf{MDD}^T\mathbf{M}^T$ is represented as a matrix where the entries are second degree polynomials in the indeterminates $\{y_e\}_{e \in E \setminus B}$ for some basis B of $\mathcal{M}(U)$ such that these polynomials are short (polynomially long in terms of the input data) expressions. The complexity of the computation of the rank of such a matrix is known to be randomized polynomial. Namely, the evaluation of a random substitution $\mathbf{L}'$ in a finite extension $\tilde{F}$ of the finite field F can be evaluated in polynomial time by Gaussian elimination. By Lemma 2.4.6, the probability that $\operatorname{rk}(\mathbf{L}) \neq \operatorname{rk}(\mathbf{L}')$ can be made arbitrarily small by choosing $\tilde{F}$ large enough. This method also works if F is a field of rational functions over a finite field F', we just have to substitute randomly chosen elements of a suitably large extension of F'. □

Theorem 2.9.8 *Let U be an (E, F)-space and $M \subseteq 2^E$ the family of sets $S \subseteq E$ for which $\varphi(U/S) = \varphi(U) - |S|$. Then M is the family of the independent sets in $\mathcal{M}(\mathcal{B}(\mathrm{S}(U)))$.*

Proof: The proof is an immediate consequence of Theorem 2.9.6, Lemma 2.3.3 and Lemma 2.4.3. □

2.10 Factor-Criticality Revisited

In this section we discuss implications of the Characterization Theorem to graphs. We have seen in Section 2.6 that a graph is factor-critical if and only if the symplectification of its cutset space is bicycle free. Now we can easily deduce the following characterization of factor-criticality from Theorem 2.9.5:

Corollary 2.10.1 *A graph G is factor-critical if and only if its cycle matroid can be represented by an alternating projection matrix over some field of characteristic* 2. □

Note the simlarity of this result to the ones in [Jaeger 1983a] and [Jaeger 1983b]. Our factor-criticality condition are more sophisticated than the results there, since we had to drop the binarity condition and go over to arbitrary fields of characteristic 2. In fact there are examples of factor-critical graphs whose cycle matroid cannot be represented by a binary alternating projection matrix.

Another reformulation can be given in terms of weighted Laplacians:

Corollary 2.10.2 *A graph G with incidency matrix $\mathbf{D}$ is factor-critical if and only if there is a diagonal weight matrix $\mathbf{W}$ such that $\mathbf{DWD}^T$ is an alternating matrix with corank 1.*

Proof: We can restrict our attention to connected graphs since neither statement holds if G is not connected. Let $\mathbf{D'}$ be the matrix resulting from the deletion of an arbitrary row of $\mathbf{D}$. It is a minimimal generating matrix of the cutset space of G. By Theorem 2.9, G is factor-critical iff the symplectification of the representation given by $\mathbf{D'}$ is bicycle free. Let $\mathbf{W}$ be the diagonal weight matrix of the symplectification coefficients. Let U be the row space of $\mathbf{D'}$ (which is the same as the row space of $\mathbf{D}$). By Lemma 2.3.1,

$$\beta(U) = \operatorname{corank} \mathbf{D'WD'}.$$

Since $\operatorname{corank} \mathbf{DWD}^T \geq 1$, it is clear that

$$\operatorname{corank} \mathbf{D'WD'}^T = 0 \iff \operatorname{corank} \mathbf{DWD}^T = 1.$$

If there is diagonal weight matrix with $\operatorname{corank} \mathbf{DWD}^T = 1$ such that $\mathbf{DWD}^T$ is alternating, then it implies that one can omit a row of $\mathbf{D}$ resulting in $\mathbf{D'}$ with $\operatorname{corank} \mathbf{D'WD'}^T = 0$. Then again, by Lemma 2.3.1, it means that $\mathbf{D'W}$ is a bicycle-free symplectic representation of the cycle matroid of G. Hence, G is factor-critical by Theorem 2.9. □

Theorem 2.8.8 specializes to the following statement in the case of graphs:

Corollary 2.10.3 *In a graph G, the family of independent edge sets (with respect to the cycle matroid of G) contracting to a factor-critical graph forms the family of feasible sets of an even representable Δ-matroid.* □

This result generalizes the main result of [Szigeti 1996]. Note that the proof of Theorem 2.8.8 gives an explicit representation of this Δ-matroid. In fact the representation can be contructed by the following steps:

- Let $G' = G \div S$ be some factor-critical subdivision of G. E.g. for any spanning tree T, $G \div T$ is factor-critical.
- Let $\mathbf{P}$ be the projection matrix to the symplectified cutset space of G'.
- Let M be the Δ-matroid defined by $\mathbf{P}$.

$M\Delta S$ is the matroid in question.

2.11 Acyclic Orientations with Parity Conditions

Orientation problems with parity conditions were studied in [Frank 1999] and [Frank and Király 1999]. They dealt with parity constrained k-connected orientations. Acyclicity can also be viewed as a connectedness criterion, although it is an upper rather than a lower bound on the connectedness of the (oriented) graph. We will show that our main result gives an algebraic randomized polynomial time algorithm to find T-odd acyclic orientations of a graph. It is not even known whether this problem is in co-NP. The results of this section strongly suggest that it is polynomial.

Let $G = (V, E)$ be a connected graph and T be a subset of the nodes. An orientation of G is said to *T-odd* (or *T-even*) if the set of nodes with odd (or even) outdegree is exactly T.

A cycle of the cographic matroid of a connected graph G is a cut $E(X, V \backslash X)$ for which both $G[X]$ and $G[V \setminus X]$ are connected. Hence, an ear-decomposition of the cocycle matroid is equivalent to a procedure in which we repeatedly delete a cut increasing the number of connected components by exactly one as long as we have some edge. To describe this process formally we will need some basic terminology: A *laminar* set system on V is a family of subsets of V satisfying that each pair (S_1, S_2) of sets of the family holds $S_1 \cap S_2 \in \{\emptyset, S_1, S_2\}$.

Definition 2.11.1 (Cut-Decoposition) *A cut decomposition of G is a laminar set system $\mathcal{D}$ on G satisfying the following conditions:*

(1) $\emptyset \notin \mathcal{D}$.

(2) $V \in \mathcal{D}$.

(3) $\{v\} \in \mathcal{D}$ *for each* $v \in V$.

(4) For each nonsingleton $S \in \mathcal{D}$, there are $S_1, S_2 \in \mathcal{D}$ partitioning S.

We call two disjoint sets of a cut-decomposition *neighbouring* if their union is also in the decomposion. A *cut* of decomposition $\mathcal{D}$ is an edge set $E(S_1, S_2)$ (consisting of all edges having one end in S_1 and the other one in S_2), where S_1 and S_2 are neighbouring sets of the decomposition. We call a decomposition *odd* if each of its cuts is odd. A decomposition is *connected* if each of its sets induces a connected subgraph of G. Is is easy to see that each odd ear-decomposition of the cocycle matroid of G comes from a connected odd cut-decomposition of G. On the other hand, a connected odd cut-decomposition gives rise to a odd ear-decomposition of the cocycle matroid of G. Since each odd cut-decomposition is necessarily connected, we can drop the connectedness condition in the above correspondence. Now we show that we can restrict ourselves to a very special class of cut-decompositions, namely to the ones induced by some total ordering of the nodes: Let $v_1 < v_2 < \cdots < v_n$ be a total ordering of the node-set of G. This *induces* the cut-decomposition $\mathcal{D}(<)$ in the following way

$$\mathcal{D}(<) \stackrel{\text{def}}{=} \{\{v_1\}, \{v_2\}, \ldots, \{v_n\}, \{v_1, v_2\}, \{v_1, v_2, v_3\}, \ldots, \{v_1, \ldots, v_n\}\}\,.$$

Now we state:

Proposition 2.11.2 *If there is an odd cut-decomposition of $G = (V, E)$, then there is one induced by some total ordering of V.*

Proof: We define the *depth* of a set S of a cut-decomposition to be the length of a maximum chain of subsets of S in the decomposition. The *depth of a cut-decomposition* is the depth of V in that decomposition. It is straightforward that the depth of some cut-decomposition of G is $|V|$ if and only if it is induced by some total ordering of the nodes. We prove that G has an odd cut-decomposition of depth $n \stackrel{\text{def}}{=} |V|$ by induction on n. The statement clearly holds if $n \leq 3$. We assume that $n \geq 4$ and that the statement holds for each graph having less than n nodes. Consider an odd cut-decomposition $\mathcal{D}$ of maximal depth. Let $S \cup T = V$ the unique partition of V in $\mathcal{D}$. There are two cases:

(1) One of them (let us say S) is singleton: $\mathcal{D}$ induces an odd cut-decomposition $2^T \cap \mathcal{D}$ on T. By the induction assumption, there is an odd cut decompositon $\mathcal{D}'$ on T of length $n-1$. Hence, $\mathcal{D}' \cup \{S\} \cup \{V\}$ is an odd cut-decomposition of G of length $|V|$.

(2) S and T are not singletons: We have $S = S_1 \cup S_2$ and $T = T_1 \cup T_2$ where $S_1, S_2, T_1, T_2 \in \mathcal{D}$. Let S be of higher (or equal) depth than T. Since $E(S,T)$ is odd, we can assume w.l.o.g. that $E(S,T_1)$ is odd, $E(S,T_2)$ is even and therefore, $(\mathcal{D} \setminus \{T\}) \cup \{S \cup T_1\}$ is an odd cut-decomposition of higher depth than $\mathcal{D}$, contradicting the maximality of the depth of $\mathcal{D}$. □

We say that an acyclic orientation is *odd* if each node but the unique sink has odd outdegree. The last proposition can be easily reformulated:

Proposition 2.11.3 *The cocycle matroid of a connected graph has an odd ear-decomposition if and only if the graph has an odd acyclic orientation.*

Namely, we can take the acyclic orientation induced by the ordering, or an arbitrary ordering compatible with the given orientation. □

We call an odd acyclic orientation *v-rooted* if its unique sink is v.

Proposition 2.11.4 *If graph G allows an odd acyclic orientation, then for every node v of G, there is a v-rooted odd acyclic orientation.*

Proof: The statement is trivial for one node graphs. Let G be an counterexample with minimal number of nodes. Let u be the root of the known odd acyclic orientation. Let u' be an element next to u in some ordering compatible to the orientation. That is, u' has exactly one outgoing edge e and it points to u. If $u' = v$, then reversing e does not induce any cycle and it results in a v-rooted acyclic orientation of G. Otherwise the contraction of e does not induce any new cycle, so it gives rise to an odd acyclic orientation on G/e. By minimility of G, there is a v-rooted acyclic orientation of G/e. It is easy to see that (by choosing an appropriate orintation for e), it induces a v-rooted odd-acyclic orientation on G. □

Proposition 2.11.5 *Let $G = (V,E)$ be a graph and $T \subseteq V$ an arbitrary subset of nodes. Let $G' = (V \cup v, E \cup \{(v,u) | u \in T\})$ a new graph which arises from G by adding the new node v and $|T|$ edges connecting v with each node of T. G has a T-even acyclic orientation if and only if G' has an odd acyclic orientation.*

Proof: If G has a T-even acyclic orientation, then orienting each edge adjacent to v in the direction of v induces an odd acyclic orientation of G. On the other hand, if G has an odd acyclic orientation, there is a v-rooted one. Its restriction to G is a T-even orientation.

Proposition 2.11.3 together with Theorem 2.9.5 gives:

Corollary 2.11.6 *A graph has an odd acyclic orientation if and only if its cocycle matroid can be represented by an alternating projection matrix over some field of characteristic* 2. □

Corollary 2.11.7 *A graph has an odd acyclic orientation if and only if the symplectification of its binary cocycle space is bicycle free.* □

The matrix reformulation (cf. Lemma 2.3.1 and [Lovász 1979])) of this result implies the following corollary:

Corollary 2.11.8 *One can decide in randomized polynomial time whether a graph has an odd acyclic orientation.*

Proof: The dimension of the symplectification of the bicycle space is

$$\operatorname{corank}(G) - \operatorname{rk}(\mathbf{XWX}^T)$$

where $\mathbf{X}$ is a minimal generating matrix of the cycle space over GF(2) and $\mathbf{W}$ is a diagonal matrix whose entries are suitably chosen linear polynomials (see Section 2.7) in $\operatorname{rk}(G)$ many algebraically independent indeterminates. The regularity of $\mathbf{XWX}^T$ is to be decided. This can be done by substituting the indeterminates by random values from a sufficiently large algebraic extension of GF(2) (cf. Lemma 2.4.6 and Theorem 2.9) and computing the rank of this matrix. □

It is not known whether there is a polynomial algorithm for solving this problem.

Corollary 2.11.9 *For a graph $G = (V, E)$ and $T \subseteq V$, it can be decided in randomized polynomial time whether G has a T-even acyclic orientation.*

Proof: Immediate by the last corollary and 2.11.5.

Corollary 2.11.10 *For a connected graph G, the system of those edge sets whose deletion gives an odd acyclic oriantable graph is a (representable) even Δ-matroid.*

This is an easy consequence of Theorem 2.8.8. □

Note that one can clearly drop the independentness criterion of Theorem 2.8.8, since deleting such edge sets gives rise to unconnected graphs which cannot have any odd acyclic orientation.

Corollary 2.11.11 *For a connected graph $G = (V, E)$, the system of those substets $T \subseteq E$ admitting a T-odd acyclic orientation form the feasible sets of a representable even Δ-matroid.*

Proof: We show the result for the T-even case. The Δ-matroid for the T-odd sets are obtained by twisting (taking symmetric difference with V). Construct the auxilary graph $G' = (V \cup v, E \cup \{(u,v)|u \in V\})$. The Δ-matroid of Corollary 2.11.10 restricted to the newly inserted edges induces the Δ-matroid in question by Proposition 2.11.5. □

Corollaries 2.11.10 and 2.11.11 imply that even the following weighted generalizations are solvable in randomized polynomial time:

(1) Given a weighting on the edges and a subset T of nodes, determine a minimum weight subset S of edges whose deletion admits a T-odd acyclic orientation of $G \setminus S$.

(2) Given a weighting on the nodes and a subset T of nodes, determine a minimum weight subset S of nodes admitting a $(T\Delta S)$-odd acyclic orientation of G.

However, it is an open question whether these problems can be solved in deterministic polynomial time. These results strongly suggest a positive answer.

Chapter 3

Tools from nonlinear optimization

In this section we will not only restate some well-known facts from nonlinear convex optimization, but also develop a new generalization of the contrained subgradient method. This is used to minimize a convex function over a convex region, by projecting the iterated values to the region in each step of the method Practical tests show that then the projection step may dominate the run time of the constrained subgradient algorithm even if the region is moderately complicated.

Our newly developed method combining the subgradient method and the idea of cyclic projections can give practically superior results in term of run-time if the feasible region is the intersection of convex sets onto which the projection is easily computed. On the way to the proof of convergence of this generalized method, we will have to introduce the concept of *nicely intersecting constellations*. The subgradient method with cyclic projections can be proved to converge for nicely intersecting set systems, however it will be shown that any constellation consisting of closed polyhedral sets with nonempty intersection is nicely intersecting, furthermore any set systems consisting of closed convex sets with bounded intersection is nicely intersecting as well.

The new algorithm contains the original constrained subgradient method and the method of cyclic projections as subcases. It provides an extremely general new method to compute the projection of an arbitrary point of the space onto a polyhedral set with respect to an arbitrary norm.

3.1 General Convergence

While studying the convergence of optimization algorithms, the following simple, but often useful theorem will be needed. We do not state it in its full generality, but in a simplified form suitable for our means (cf. [Minoux 1986] Chapter 2):

Theorem 3.1.1 (Zangwill 1969) *Let X be a topological vector space, $Y \subseteq X$ a subset of it, $f : X \longrightarrow X$ a continuous map and $(\mathbf{x}^n)_{n=1}^{\infty}$ the sequence defined by*

$$\mathbf{x}^{n+1} = f(\mathbf{x}^n).$$

Assume that the following conditions hold:

(1) There is a compact subset of X containing all $\mathbf{x}^n$.

(2) There exists a **function of descent** *$z : X \mapsto \mathbb{R}$ satistfying*

(a) $\mathbf{x} \notin Y \Longrightarrow z(f(\mathbf{x})) < z(\mathbf{x})$,

(b) $\mathbf{x} \in Y \Longrightarrow z(f(\mathbf{x})) \leq z(\mathbf{x})$;

then the limit of any convergent subsequence of $(\mathbf{x}^n)_{n=1}^{\infty}$ is in Y.

3.2 Lagrangian Duality

Normally, constrained problems are harder to solve than unconstrained ones. An established approach for solving costrained problems is to introduce dual variables for a subset of constraints and optimize the dual problem.

For proofs of the theorems in this section, the interested reader is referred to [Bazaraa, Sherall and Shetty 1993].

Suppose that we are given a domain $X \subseteq \mathbb{R}^k$ and functions $f : X \longrightarrow \mathbb{R}$, $g_1, \ldots g_l : X \longrightarrow \mathbb{R}$ and $h_1, \ldots h_m : X \longrightarrow \mathbb{R}$. Let the primal problem be given by:

$$\begin{array}{lll} \text{minimize} & f(\mathbf{x}) & \\ \text{s.t} & g_i(\mathbf{x}) \leq 0 & \forall i \in \{1, \ldots, l\} \\ & h_i(\mathbf{x}) = 0 & \forall i \in \{1, \ldots, m\} \end{array}$$

Then the **Lagrange function** is defined by

$$Ł(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \stackrel{\text{def}}{=} f(\mathbf{x}) + \sum_{i=1}^{l} \boldsymbol{\lambda}_i g_i(\mathbf{x}) + \sum_{i=1}^{m} \boldsymbol{\mu}_i h_i(\mathbf{x})$$

The **dual objective function** is given by

$$f^*(\boldsymbol{\lambda}, \boldsymbol{\mu}) \stackrel{\text{def}}{=} \inf\{Ł(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \mid \mathbf{x} \in X\}.$$

It is a simple fact that f^* is concave (without making any assumptions about f). The **dual problem** is

$$\begin{array}{lll} \text{maximize} & f^*(\boldsymbol{\lambda}, \boldsymbol{\mu}) & \\ \text{s.t} & \boldsymbol{\lambda}_i \geq 0 & \forall i \in \{1, \ldots, l\} \end{array}$$

The proof of the following well-known fact is very simple:

Theorem 3.2.1 (Weak duality) *Let* $\mathbf{x}$ *be a feasible solution of the primal problem,* $\boldsymbol{\lambda}, \boldsymbol{\mu}$ *a feasible solution of the dual problem, then the following inequality holds:*

$$f(\mathbf{x}) \geq f^*(\boldsymbol{\lambda}, \boldsymbol{\mu}).$$

Of course, in general, the optimal values of the primal and dual problems are not equal, even in the case of convex objective and contraint functions.

Theorem 3.2.2 (Slater condition) *Let the domain* X *be a nonempty closed convex subset of* $\mathbb{R}^n$*. Assume that all involved functions* $f, g_1, \ldots, g_l$ *are convex, moreover* $h = (h_1, \ldots, h_m)$ *is an affine linear map, that is given by*

$$h(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$$

where $\mathbf{A}$ *is matrix of suitable size. Additionally assume that there is vector* $\mathbf{x} \in X$ *with* $g(\mathbf{x}) < 0$ *and that the zero vector is in the interior of* $h(X) \stackrel{\text{def}}{=} \{h(\mathbf{x}) \mid \mathbf{x} \in X\}$*. Then the following statements are true:*

(1) The optimal values of the primal and dual problems are equal.

(2) If the primal optimum is finite then the optimum of the dual is attained at $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ *with* $\boldsymbol{\lambda} \geq 0$*. If the optimum of the primal problem is attained at* $\mathbf{x}$*, then* $\boldsymbol{\lambda}^T g(\mathbf{x}) = 0$ *holds.*

The concept of saddle point is also useful for studying the relationship between the primal and dual problem.

Definition 3.2.3 (Saddle point) $(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \in X \times \mathbb{R}^l_{\geq 0} \times \mathbb{R}^m$ *is called* **saddle point** *of the Lagrange function Ł, if*

$$Ł(\mathbf{x}, \boldsymbol{\lambda}', \boldsymbol{\mu}') \leq Ł(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \leq Ł(\mathbf{x}', \boldsymbol{\lambda}, \boldsymbol{\mu})$$

holds for all $(\mathbf{x}', \boldsymbol{\lambda}', \boldsymbol{\mu}') \in X \times \mathbb{R}^l_{\geq 0} \times \mathbb{R}^m$.

Theorem 3.2.4 (Saddle point optimality) *For* $(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \in X \times \mathbb{R}^l_{\geq 0} \times \mathbb{R}^m$ *the following statements are equivalent:*

(1) $(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ *is saddle point of Ł.*

(2) $\mathbf{x}$ *and* $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ *are optimal solutions of the primal and the dual problem with equal value (That is* $f(\mathbf{x}) = f^*(\boldsymbol{\lambda}, \boldsymbol{\mu})$*).*

(3) *(a)* $Ł(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \min\{Ł(\mathbf{x}', \boldsymbol{\lambda}, \boldsymbol{\mu}) \mid \mathbf{x}' \in X\}$,
(b) $g(\mathbf{x}) \leq 0, h(\mathbf{x}) = 0$ *and*
(c) $\boldsymbol{\lambda}^T g(\mathbf{x}) = 0$.

Definition 3.2.5 (Karush-Kuhn-Tucker-Point) *Assume that* f,g *and* h *are differentiable functions. Let* $\mathbf{x}$ *be a feasible solution of the primal problem. If there are* $\boldsymbol{\lambda} \in \mathbb{R}^l_{\geq 0}$ *and* $\boldsymbol{\mu} \in \mathbb{R}^m$ *with*

$$\nabla f(\mathbf{x}) = \sum_{i=1}^{l} \boldsymbol{\lambda}_i \nabla g_i(\mathbf{x}) + \sum_{i=1}^{m} \boldsymbol{\mu}_i \nabla h_i(\mathbf{x}) = 0$$

and

$$\boldsymbol{\lambda}^T g(\mathbf{x}) = 0,$$

then $\mathbf{x}$ *is called* **Karush-Kuhn-Tucker-Point**.

The following theorem gives necessary and sufficient condition for a point being saddle point:

Theorem 3.2.6 (Karush, Kuhn and Tucker) *Assume that* X *is nonempty and open,* f *and* g *are convex and differentiable and* h *is affine.*

(1) If $\mathbf{x}$ *is a Karush-Kuhn-Tucker-point then it is the first component of a saddle point.*

(2) If $\mathbf{x}$ *is and interior point of* X *and it is the first component of a saddle point, then it is a Karush-Kuhn-Tucker-point.*

3.3 Subgradient Method

Definition 3.3.1 (Subgradient) *Let $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ be a convex function. A vector $\mathbf{v} \in \mathbb{R}^n$ is called subgradient of f at $\mathbf{x}^0$ if for all $\mathbf{x} \in \mathbb{R}^n$ holds*

$$f(\mathbf{x}) \geq f(\mathbf{x}^0) + \mathbf{v}^T(\mathbf{x} - \mathbf{x}^0).$$

For a concave function f, $\mathbf{v}$ is called subgradient of f at $\mathbf{x}^0$ if for all $\mathbf{x} \in \mathbb{R}^n$ holds:

$$f(\mathbf{x}) \leq f(\mathbf{x}^0) + \mathbf{v}^T(\mathbf{x} - \mathbf{x}^0).$$

The notion of subgradient generalizes that of the ordinary gradient: if f is differentiable in an open neighborhood of $\mathbf{x}^0$, then the gradient is the only subgradient.

Theorem 3.3.2 *Let $X \subseteq \mathbb{R}^n$ nonempty and compact, $f : \mathbb{R}^n \longrightarrow \mathbb{R}$, $h : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ continous functions,*

$$\Theta(\boldsymbol{\lambda}) \stackrel{\text{def}}{=} \inf_{\mathbf{x} \in X} \{f(\mathbf{x}) + \boldsymbol{\lambda}^T h(\mathbf{x})\}$$

and

$$M(\boldsymbol{\lambda}) \stackrel{\text{def}}{=} \left\{\mathbf{x} \in X \mid \mathbf{x} \text{ minimizes } f(\mathbf{x}) + \boldsymbol{\lambda}^T h(\mathbf{x})\right\}.$$

If $\mathbf{x} \in M(\boldsymbol{\lambda})$ for some $\boldsymbol{\lambda} \in \mathbb{R}_{\geq 0}$, then $h(\mathbf{x})$ is a subgradient of $\Theta(\boldsymbol{\lambda})$.

The interested reader is referred to [Bazaraa, Sherall and Shetty 1993] Theorem 6.3.4 for a proof. □

Subgradient methods can be used to optimize non-differentiable objective functions that have a subgradient at each point of the domain of f. The constrained subgradient method can be used for optimizing f over a closed convex set $X \subseteq \mathbb{R}^n$. The subgradient algorithm requires a positive zero-series (ρ_k) with divergent sum (For example $\rho_n = \frac{1}{n}$ is a suitable series) for provable convergence.

CONSTRAINED SUBGRADIENT METHOD	
Input:	A convex function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ and a convex set $X \subseteq \mathbb{R}^n$.
Output:	Sequence $(\mathbf{x}^n) \in (\mathbb{R}^n)^{\mathbb{N}}$ containing a subsequence converging to a minimum of f in X.

Let π_X denote the orthogonal projection onto X.

① Set $k = 0$, choose an arbitrary starting point $\mathbf{x}^0$

② Compute a subgradient $\mathbf{g}_k$ of f at $\mathbf{x}^k$, **stop** if $\mathbf{g}_k = 0$.

③ Let

$$\mathbf{x}^{k+1} = \pi_X \left(\mathbf{x}^k - \rho_k \frac{\mathbf{g}_k}{||\mathbf{g}_k||} \right)$$

③ Increase k and **go to** ②

Theorem 3.3.3 ([Ermoliev 1966],[Polyak 1967]) *Let $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ be a convex function, $X \subseteq \mathbb{R}^n$ a closed convex set so that either X is bounded or $f(\mathbf{x}) \to \infty$ for $||\mathbf{x}|| \to \infty$. Let $(\rho_k) \in \mathbb{R}^{\mathbb{N}}_{>0}$ be a positive zero sequence the sum of which tending to ∞. Then the constrained subgradient algorithm generates a sequence that has a subsequence tending to a minimum of f.* □

For a proof, see also [Minoux 1986] Theorem 4.9. Note that since the subgradient of the dual function with respect to some constraints can be easily computed by Theorem 3.3.2, the subgradient method can be used to optimize the dual function of a constrained optimization problem. To maximize concave functions, only step ③ is to be modified: the substraction of the renormalized subgradient must be replaced by addition.

3.4 The Cyclic Coordinates Method

The method of *cyclic coordinates* (also called the *method of cyclic relaxation* in [Minoux 1986] Section 4.4.1) is a simple method for minimizing convex functions over $\mathbb{R}^n$ or over a box. This method consists of successively minimizing one component of the vector while leaving all other components fixed. It is not very efficient in general, but performs well if the effect of the variables on the objective function interact weakly among themselves. The linear convergence of this method was established in [Luo and Tseng 1992] for a large class of objective functions. Here we describe this result.

Let $f : \mathbb{R}^n \longrightarrow \mathbb{R} \cup \{\pm\infty\}$ and $\underline{\mathbf{b}} < \overline{\mathbf{b}} \in \mathbb{R}^n$. The method of cyclic coordinates is an iterative method consisting of steps of the following form:

$$\mathbf{x}^{k+1} \leftarrow \arg \min_{\underline{\mathbf{b}}_i \leq x \leq \overline{\mathbf{b}}_i} f(\mathbf{x}_1^k, \ldots, \mathbf{x}_{i-1}^k, x, \mathbf{x}_{i+1}^k, \ldots, \mathbf{x}_n^k)$$

The only requirement made on the choice of the coordinates to be optimized is that there has to be an integer B such that every coordinate is optimized at

least once in every B successive iterations. This is the so called *almost cyclic rule*.

Theorem 3.4.1 ([Luo and Tseng 1992]) *Let $f : \mathbb{R}^n \longrightarrow \mathbb{R} \cup \{\pm\infty\}$ be a proper closed convex function*[1] *of the form*

$$f(\mathbf{x}) = g(\mathbf{E}\mathbf{x}) + \langle \mathbf{c}, \mathbf{x} \rangle$$

where $g : \mathbb{R}^m \longrightarrow \mathbb{R} \cup \{\pm\infty\}$ is a proper closed convex function and $\mathbf{E}$ is an $m \times n$ matrix having no zero column. Let $\underline{\mathbf{b}} < \overline{\mathbf{b}} \in \mathbb{R}^n$ and $X \stackrel{\text{def}}{=} \{\mathbf{x} \mid \underline{\mathbf{b}} \leq \mathbf{x} \leq \overline{\mathbf{b}}\}$. Let X^* denote the set of vectors minimizing f over X. Furthermore, assume that:

(1) $X^* \neq \emptyset$.

(2) The effective domain $D = \{\mathbf{x} \in \mathbb{R}^m \mid g(\mathbf{x}) < \infty\}$ of g is open and g is strictly convex and twice continuosly differentiable over D.

(3) $\nabla^2 g(\mathbf{E}\mathbf{x}^*)$ is positive definite for each $\mathbf{x}^* \in X^*$.

Then the method of cyclic coordinates (using any almost cyclic rule) generates a sequence which converges at least linearly to an element of X^*. □

The following easy corollary can be formulated:

Corollary 3.4.2 *Let $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ be a strictly convex and twice continously differentiable function. Let $\underline{\mathbf{b}} < \overline{\mathbf{b}} \in \mathbb{R}^n$ and $X \stackrel{\text{def}}{=} \{\mathbf{x} \mid \underline{\mathbf{b}} \leq \mathbf{x} \leq \overline{\mathbf{b}}\}$. The method of cyclic coordinates (using any almost cyclic rule) generates a sequence which converges at least linearly to* $\arg\min_{\mathbf{x} \in X} f(\mathbf{x})$. □

3.5 The Method of Cyclic Projections

Let $\mathcal{C} = \{C_1, \ldots, C_m\}$ be a system of closed convex sets in $\mathbb{R}^n$ with nonempty intersection I and orthogonal projections π_i to C_i (we assume that π_i are relatively efficiently evaluated for each $\mathbf{x} \in \mathbb{R}^n$). To find an element of I, the following simple method was proposed in [Cheney and Goldstein 1959]:

[1] See [Rockafellar 1997]. In particular, every real valued convex function $\mathbb{R}^n \longrightarrow \mathbb{R}$ is proper closed convex.

METHOD OF CYCLIC PROJECTIONS

Input: A system $\mathcal{C} = \{C_1, \ldots, C_m\}$ of closed convex sets in $\mathbb{R}^n$ and $\mathbf{x}^0 \in \mathbb{R}^n$

Output: Sequence $(\mathbf{x}^k)$ with $\lim_{k\to\infty} d(\mathbf{x}^k, I) = 0$.

① $k \leftarrow 0$

② $\mathbf{x}^{k+1} \leftarrow \pi_{C_1} \circ \cdots \circ \pi_{C_m}(\mathbf{x}^k)$

③ increase k and **go to** ②

The sequence becomes stationary if an element of the intersection is reached, which is not necessarily the case, even if the sets are compact polyhedrons. The convergence of the method may be excessively slow, but it may be practically useful under certain circumstances.

Theorem 3.5.1 ([Cheney and Goldstein 1959]) *For a family of closed convex sets with nonempty intersection I and initial vector $\mathbf{x}^0 \in \mathbb{R}^n$, the method of cyclic projections generates a sequence $(\mathbf{x}^k)_{k=1}^{\infty}$ converging to a point in I.*

The algorithm of cyclic projections has been modified in [Dykstra 1983], [Han 1988] and [Boyle and Dykstra 1986] to generate the projection of $\mathbf{x}^1$ onto I. The methods presented there are not particularly efficient but may be useful if there is no special purpose algorithm because the structure of the intersection is somewhat arbitrary.

For our purposes – to use them in the projection step of the constrained subgradient method – the above projections are not really useful: they are quite slow and it turns out that the computation of an exact projection is often an overkill. Instead we will develop a common generalization of the method of cyclic projections and the constrained subgradient method fitting our practical needs. The generalized method contains the constrained subgradient method and the method of cyclic projections as subcases and yields a general method for computing the projection onto the intersection with respect to any norm.

3.6 Nicely Intersecting Constellations

Although the method of cyclic projection does not need special assumptions about the sets it is applied to, our generalized method is more sensible to pathological cases and needs that the system of sets satisfies a certain property

("being nicely intersecting") in order to converge. This property will turn out to be not too restrictive: for example any family of closed convex sets with bounded intersection or any family of polyhedrons possesses it.

Definition 3.6.1 *Let $\mathcal{C} = \{C_1, \ldots, C_m\}$ be a system of subsets of in $\mathbb{R}^n$ with intersection $I \stackrel{\text{def}}{=} \bigcap_{i=1}^{m} C_i$. For $\epsilon > 0$, let $\mathfrak{d}(\epsilon, \mathcal{C})$ be the supremum of all $\delta \geq 0$ for that the following implication holds for all $\mathbf{x} \in \mathbb{R}^n$*

$$d(\mathbf{x}, C_i) \leq \delta \quad \forall i \in \{1, \ldots, m\} \Longrightarrow d(\mathbf{x}, I) \leq \epsilon. \tag{3.1}$$

We say that $\mathcal{C}$ is a **nicely intersecting constellation** *if $\mathfrak{d}(\epsilon, \mathcal{C}) > 0$ for all $\epsilon > 0$.*

Proposition 3.6.2 *Let C_1 be a compact set and $C_2, \ldots, C_m$ closed sets of $\mathbb{R}^n$, then $\{C_1, \ldots, C_m\}$ is a nicely intersecting constellation.*

Proof: Assuming the converse, there is a positive ϵ and sequence $(\mathbf{x}^k)$ with

$$d(\mathbf{x}^k, C_i) \to 0 \qquad \forall i \in \{1, \ldots, m\},$$

and

$$d(\mathbf{x}^k, I) > \epsilon. \tag{3.2}$$

We can deduce by the compactness of C_1 that $\mathbf{x}^k$ is bounded. So it has a subsequence whose limit is contained in I by its closedness, contradicting (3.2). □

However, the compactness of at least one set is crucial. There are examples of pairs of two closed convex sets in 3-dimensional space that are not nicely intersecting. For example, let C_1 be the halfspace $\{(x, y, z) \mid x \leq 0, y, z \in \mathbb{R}\}$, and C_2 the closure of the convex hull of $\{(0, 0, z) \mid z > 0\} \cup \{x, 1, 1/x \mid x \in (0, 1]\}$.

Lemma 3.6.3 *Let $\mathcal{C} = \{C_1, \ldots, C_m\}$ be a family of closed convex sets with nonempty intersection I that is not nicely intersecting. Then for each vector $\mathbf{c} \in I$, there is a ray $\mathbf{c} + \mathbb{R}_{\geq 0}\mathbf{d}$ contained in I. Furthermore, if every subconstellation of $\mathcal{C}$ is nicely intersecting then for each $i \in \{C_1, \ldots, C_m\}$, there is a sequence $(\mathbf{x}^k)$ disjoint to C_i such that*

$$\frac{\mathbf{x}^k - \mathbf{c}}{\|\mathbf{x}^k - \mathbf{c}\|} \to \frac{\mathbf{d}}{\|\mathbf{d}\|}.$$

Proof: We can assume without loss of generality that $\mathbf{c} = 0 \in I$. Since the constellation is not nicely intersecting, there is $\epsilon > 0$ and a sequence $(\mathbf{y}^k)$ with

$$d(\mathbf{y}^k, C) \to 0 \qquad \forall C \in \mathcal{C}$$

and

$$d(\mathbf{y}^k, I) > \epsilon. \tag{3.3}$$

We can assume that $(\mathbf{y}^k)$ is not bounded, otherwise we can finish the proof like in Proposition 3.6.2. Consider the set $\left\{ \frac{\mathbf{y}^k}{\|\mathbf{y}^k\|} \mid k \in \mathbb{N} \right\}$. Since it is bounded, it has an accumulation point $\mathbf{d}$, so we can assume without loss of generality that $\frac{\mathbf{y}^k}{\|\mathbf{y}^k\|} \to \mathbf{d}$. Now we can easily deduce by the first assumption and using the convexity of the sets that the ray $\mathbb{R}_{\geq 0}\mathbf{d}$ is contained in any $C \in \mathcal{C}$. Let us assume that every subconstellation of $\mathcal{C}$ is nicely intersecting and i is an arbitrary index in $\{1, \ldots, m\}$. Then there is an index $K \in \mathbb{N}$ such that $d(\mathbf{y}^k, \bigcap_{j \neq i} C_j) < \epsilon$. Let π denote the projection onto $\bigcap_{j \neq i} C_j$. So we have $d(\pi(\mathbf{y}^k), \mathbf{y}^k) < \epsilon$ and therefore $\pi(\mathbf{y}^k)$ is not in C_i otherwise it would be in I contradicting (3.3). Hence, $(\pi(\mathbf{y}^k))_{i=1}^{\infty}$ is a sequence fitting the second statement. □

An immediate consequence of this is the following corollary:

Corollary 3.6.4 *A family of closed convex sets with bounded intersection is a nicely intersecting constellation.* □

The obvious proof of the following proposition is left to the reader:

Proposition 3.6.5 *Let $\mathcal{C} = \{C_1, \ldots, C_m\}$ be a nicely intersecting constellation of sets, and $X_1, \ldots, X_k \subseteq \mathcal{C}$ with $\bigcup_{i=1}^{m} X_i = \mathcal{C}$, then $\{ \bigcap_{C \in X_1} C, \ldots, \bigcap_{C \in X_k} C\}$ is a nicely intersecting constellation.* □

Proposition 3.6.6 *Let $\mathcal{C} = \{C_1, \ldots, C_m\}$ be a family of sets in $\mathbb{R}^n$ then $\{C_1 \times \mathbb{R}, \ldots, C_m \times \mathbb{R}\}$ is a nicely intersecting constellation of sets in $\mathbb{R}^{n+1}$ if and only if $\mathcal{C}$ is a nicely intersecting constellation.*

Proof: One can easily see that for a vector $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_{n+1}) \in \mathbb{R}^{n+1}$ and any set $C \subseteq \mathbb{R}^n$ holds: $d(\mathbf{x}, C \times \mathbb{R}) = d((\mathbf{x}_1, \ldots, \mathbf{x}_n), C)$ and

$$\left(\bigcap_{i=1}^{m} C_i \right) \times \mathbb{R} = \bigcap_{i=1}^{m} (C_i \times \mathbb{R})$$

These two observations easily imply the desired result. □

Lemma 3.6.7 *Let $\mathcal{C} = \{C_1, \ldots, C_m\}$ be a family in $\mathbb{R}^n$ consisting of closed half-spaces. Then $\mathcal{C}$ is a nicely intersecting constellation.*

Proof: We proceed by induction on $n+m$. The statement is trivial for $n \leq 1$ or for $m = 1$. So, we can assume $\min\{n, m\} \geq 2$. Then, by Lemma 3.6.3, there is a ray $R = \mathbf{c} + \mathbb{R}_{\geq 0}\mathbf{d}$ contained in $\bigcap_{i=1}^{m} C_i$. Moreover, for each i, there are points $(\mathbf{x}^k) \in \mathbb{R}^n \setminus C_i$ with

$$\lim_{k\to\infty} \mathbf{x}^k = \mathbf{d}$$

whose direction relative to $\mathbf{c}$ tends to that of R, but are not in C_i. This is only possible if the ray is orthogonal to the normal-vector of each C_i. Therefore, the line of R is fully contained in all C_i. So we can rotate the whole constallation so that the line of R becomes the last coordinate achse and $\mathcal{C}$ can be written as

$$\mathcal{C} = \{C_1' \times \mathbb{R}, \ldots, C_m' \times \mathbb{R}\}.$$

which implies the that $\mathcal{C}$ is nicely intersecting by the induction assumption and Lemma 3.6.6. □

This lemma and Proposition 3.6.5, trivially implies the following important corollary:

Corollary 3.6.8 *Any family of closed convex polyhedral sets is a nicely intersecting constellation.* □

3.7 Subgradient Method with Cyclic Projections

In order to prove the convergence of the new version of the subgradient method, we will need a generalized version of the ordinary constrained subgradient method:

GENERALIZED SUBGRADIENT METHOD

Input: A convex function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ and a contraction $\pi : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ with fixpoint $\mathbf{x}$.

Output: Sequence $(\mathbf{x}^k)$ with $\liminf_{k\to\infty} f(\mathbf{x}^k) \leq f(\mathbf{x})$.

Let (ρ_k) be a sequence of positive numbers converging to zero with divergent sum. We define

$$n(\mathbf{x}) \stackrel{\text{def}}{=} \begin{cases} \frac{\mathbf{x}}{\|\mathbf{x}\|} & \text{if } \mathbf{x} \neq 0 \\ 0 & \text{otherwise} \end{cases}.$$

① Set $k = 0$ and choose an arbitrary starting point $\mathbf{x}^0$.

② Compute a subgradient $\mathbf{g}^k$ of f at $\mathbf{x}^k$.

③ Let $\mathbf{x}^{k+1} = \pi(\mathbf{x}^k - \rho_k n(\mathbf{g}^k))$.

④ Increase k and **go to** ②

The proof of the following theorem is a straightforward generalization of that of Theorem 3.3.3, but it is included here for the sake of completeness.

Theorem 3.7.1 *Let $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ be a convex function, a contraction $\pi : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ with fixpoint $\mathbf{x}$. Then the generalized subgradient method generates a sequence with $\liminf_{k\to\infty} f(\mathbf{x}^k) \leq f(\mathbf{x})$.*

Proof: If $\mathbf{g}^k = 0$ for infinitely many indices, then each corresponding $\mathbf{x}^k$ is a minimum of f over $\mathbb{R}^n$ and therefore they form a subsequence with $f(\mathbf{x}^k) \leq f(\mathbf{x})$ and the statement of the theorem is clear.

So, we can assume without loss of generality that $\mathbf{g}^k \neq 0$ for all $k \in \mathbb{N}$.

We have to show that for each positive ϵ and $N \in \mathbb{N}$, there is a $k > N$ such that $f(\mathbf{x}^k) \leq f(\mathbf{x}) + \epsilon$. Assume the converse. That is for each $k > N$ holds

$$f(\mathbf{x}^k) > f(\mathbf{x}) + \epsilon. \tag{3.4}$$

Because of the continuity of f, there is $\delta > 0$ such that

$$\|\mathbf{y} - \mathbf{x}\| \leq \delta \Longrightarrow f(\mathbf{y}) \leq f(\mathbf{x}) + \epsilon.$$

Let us define

$$\tilde{\mathbf{x}}^k \stackrel{\text{def}}{=} \delta \frac{\mathbf{g}^k}{\|\mathbf{g}^k\|} + \mathbf{x}.$$

Obviously, $\|\tilde{\mathbf{x}}^k - \mathbf{x}\| = \delta$ and hence $f(\tilde{\mathbf{x}}^k) \leq f(\mathbf{x}) + \epsilon$, so by the definition of $\mathbf{g}_k$ holds:

$$\langle \mathbf{g}^k, \tilde{\mathbf{x}}^k - \mathbf{x}^k \rangle \leq f(\tilde{\mathbf{x}}^k) - f(\mathbf{x}) < 0. \tag{3.5}$$

Now, using that π is a contraction and $\mathbf{x}$ is a fixpoint of π we see:

$$\|\mathbf{x}^{k+1} - \mathbf{x}\| = \|\pi(\mathbf{x}^k - \rho_k \frac{\mathbf{g}^k}{\|\mathbf{g}^k\|}) - \pi(\mathbf{x})\| \leq \|\mathbf{x}^k - \rho_k \frac{\mathbf{g}^k}{\|\mathbf{g}^k\|} - \mathbf{x}\|.$$

So we get:

$$\|\mathbf{x}^{k+1} - \mathbf{x}\|^2 \leq \|\mathbf{x}^k - \mathbf{x}\|^2 + \rho_k^2 - 2\rho_k \langle \frac{\mathbf{g}^k}{\|\mathbf{g}^k\|}, \mathbf{x}^k - \mathbf{x} \rangle.$$

Using

$$\langle \mathbf{g}^k, \mathbf{x}^k - \mathbf{x} \rangle = \langle \mathbf{g}^k, \mathbf{x}^k - \tilde{\mathbf{x}}^k + \delta \frac{\mathbf{g}^k}{\|\mathbf{g}^k\|} \rangle = \langle \mathbf{g}^k, \mathbf{x}^k - \tilde{\mathbf{x}}^k \rangle + \delta \|\mathbf{g}^k\|$$

and (3.5) we obtain:

$$\|\mathbf{x}^{k+1} - \mathbf{x}\|^2 \leq \|\mathbf{x}^k - \mathbf{x}\|^2 + \rho_k^2 + 2\rho_k \langle \frac{\mathbf{g}^k}{\|\mathbf{g}^k\|}, \tilde{\mathbf{x}}^k - \mathbf{x}^k \rangle - 2\delta\rho_k \leq \|\mathbf{x}^k - \mathbf{x}\|^2 + \rho_k(\rho_k - 2\delta).$$

Since $\rho_k \to 0$, there is a $N < K \in \mathbb{N}$ such that $K \leq k \Longrightarrow \rho_k \leq \delta$ and therefore for each $k \geq K$ holds:

$$\|\mathbf{x}^{k+1} - \mathbf{x}\|^2 \leq \|\mathbf{x}^k - \mathbf{x}\|^2 - \delta\rho_k.$$

Summing these inequalities starting with K yields for any $j \in \mathbb{N}$:

$$\delta \sum_{k=K}^{k+j} \rho_k \leq \|\mathbf{x}^k - \mathbf{x}\|^2 - \|\mathbf{x}^{K+j+1} - \mathbf{x}\|^2 \leq \|\mathbf{x}^k - \mathbf{x}\|^2$$

contradicting $\sum \rho_k \to \infty$. □

Lemma 3.7.2 *Let $C' \subseteq C$ be a closed convex sets in $\mathbb{R}^n$, π the L_2-projection onto C and $\mathbf{x} \in \mathbb{R}^n$. Then the following inequality holds:*

$$d(\pi(\mathbf{x}), C')^2 \leq d(\mathbf{x}, C')^2 - d(\pi(\mathbf{x}), \mathbf{x})^2 \tag{3.6}$$

Let $\mathbf{y}$ be the L_2-projection of $\mathbf{x}$ onto C'. Since C is convex and π is the L_2-projection onto C, we have:

$$\langle \pi(\mathbf{x}) - \mathbf{y}, \pi(\mathbf{x}) - \mathbf{x} \rangle \leq 0,$$

hence:

$$\begin{aligned}\|\mathbf{x}-\mathbf{y}\|^2 = \|\mathbf{x}-\pi(\mathbf{x})+\pi(\mathbf{x})-\mathbf{y}\|^2 = \\ \|\mathbf{x}-\pi(\mathbf{x})\|^2+\|\pi(\mathbf{x})-\mathbf{y}\|^2-2\langle\pi(\mathbf{x})-\mathbf{y},\pi(\mathbf{x})-\mathbf{x}\rangle \geq \\ \|\mathbf{x}-\pi(\mathbf{x})\|^2+\|\pi(\mathbf{x})-\mathbf{y}\|^2\end{aligned}$$

Therefore,

$$d(\pi(\mathbf{x}),C')^2 \leq d(\pi(\mathbf{x}),\mathbf{y})^2 \leq d(\mathbf{x},\mathbf{y})^2 - d(\pi(\mathbf{x}),\mathbf{x})^2 = d(\mathbf{x},C')^2 - d(\pi(\mathbf{x}),\mathbf{x})^2.\square$$

Proposition 3.7.3 *Let $\mathcal{C}=\{C_1,C_2,\ldots,C_m\}$ be a nicely intersecting constellation of convex sets in $\mathbb{R}^n$ with*

$$I \stackrel{\text{def}}{=} \bigcap_{i=1}^{m} C_i \neq \emptyset,$$

denote by π_i the L_2-projection onto C_i and let us define

$$\tilde{\pi}_i \stackrel{\text{def}}{=} \pi_i \circ \pi_{i-1} \circ \cdots \circ \pi_1,$$

where $\tilde{\pi}_0$ is the identity function of $\mathbb{R}^n$. Let $\mathbf{p}^k$ be a sequence of vectors in $\mathbb{R}^n$ such that $\|\mathbf{p}^k\|$ tends to 0. Let $\mathbf{x}^1$ be an arbitrary point of $\mathbb{R}^n$ and the sequence $(\mathbf{x}^k)$ defined inductively by

$$\mathbf{x}^{k+1} \stackrel{\text{def}}{=} \tilde{\pi}_m(\mathbf{x}^k)+\mathbf{p}^k.$$

For all $A>0$, there is a $\beta(A,\mathcal{C})$ such that for each zero sequence $(\mathbf{p}^k)$ satisfying

$$\|\mathbf{p}^k\| \leq \beta(A,\mathcal{C}) \qquad \forall k \in \mathbb{N}$$

holds that if $d(\mathbf{x}^1,I) \leq A$, then

$$\lim_{k\to\infty} d(I,\mathbf{x}^k) = 0. \tag{3.7}$$

Proof: We define

$$\delta(A,\epsilon,\mathcal{C}) \stackrel{\text{def}}{=} \frac{1}{2}\min\left(1,\epsilon,\frac{\left(\mathfrak{d}(\frac{\epsilon}{2},\mathcal{C})\right)^2}{m^2(2A+1)}\right)$$

and

$$\beta(X,\mathcal{C}) \stackrel{\text{def}}{=} \delta(A,A,\mathcal{C}).$$

Let $M(\epsilon)$ be defined by

$$\|\mathbf{p}^k\| < \delta(A, \epsilon, \mathcal{C}) \qquad \forall k > M(\epsilon).$$

The following simple corollary of Lemma 3.7.2 will be handy:

Claim A: *For each $k \in \mathbb{N}$ and $i < j \in \{0, \ldots, m\}$ holds:*

$$d(\tilde{\pi}_i(\mathbf{x}^k), I) \geq d(\tilde{\pi}_j(\mathbf{x}^k), I).$$

Proof of Claim A: It follows by a straightforward induction on $j - i$ using Lemma 3.7.2 applied to $I \subseteq C_{j-1}$ and $\mathbf{x}^k$. □

Let $\epsilon > 0$ be arbitrary. We want to show that there is an index K so that $d(\mathbf{x}^k, I) \leq \epsilon$ for all $k > K$. For this reason, the following claim should be proved:

Claim B: *For each k with $k > M(\epsilon)$ and $\frac{\epsilon}{2} < d(\mathbf{x}^k, I) \leq A$ holds:*

$$d(\mathbf{x}^{k+1}, I)^2 < d(\mathbf{x}^k, I)^2 - \frac{1}{2}\left(\frac{\mathfrak{d}(\frac{\epsilon}{2}, \mathcal{C})}{m}\right)^2.$$

Proof of Claim B: There is an i such that $d(\tilde{\pi}_i(\mathbf{x}^k), \tilde{\pi}_{i+1}(\mathbf{x}^k)) > \mathfrak{d}(\frac{\epsilon}{2}, \mathcal{C})/m$, otherwise for all $i \in \{1, \ldots, m\}$ holds:

$$d(\mathbf{x}^k, C_i) \leq d(\mathbf{x}^k, \tilde{\pi}_i(\mathbf{x}^k)) \leq \sum_{j=1}^{i} d(\tilde{\pi}_{j-1}(\mathbf{x}^k), \tilde{\pi}_j(\mathbf{x}^k)) \leq \frac{j\mathfrak{d}(\frac{\epsilon}{2}, \mathcal{C})}{m} \leq \mathfrak{d}(\epsilon/2, \mathcal{C})$$

contradiciting the definition of $\mathfrak{d}(\frac{\epsilon}{2}, \mathcal{C})$.

According to Lemma 3.7.2, we can write:

$$d(\tilde{\pi}_m(\mathbf{x}^k), I)^2 \leq d(\mathbf{x}^k, I)^2 - \sum_{i=1}^{m} d(\tilde{\pi}_{i-1}(\mathbf{x}^k), \tilde{\pi}_i(\mathbf{x}^k))^2 \leq d(\mathbf{x}^k, I)^2 - \left(\frac{\mathfrak{d}(\frac{\epsilon}{2}, \mathcal{C})}{m}\right)^2$$

By the triangle inequality, we can write:

$$\begin{aligned} d(\mathbf{x}^{k+1}, I)^2 \leq \big(d(\tilde{\pi}_m(\mathbf{x}^k), I) + \|\mathbf{p}^k\|\big)^2 = \\ d(\tilde{\pi}_m(\mathbf{x}^k), I)^2 + 2\|\mathbf{p}^k\| d(\tilde{\pi}_m(\mathbf{x}^k), I) + \|\mathbf{p}^k\|^2 \leq \\ d(\mathbf{x}^k, I)^2 + \|\mathbf{p}^k\| \left(d(\tilde{\pi}(\mathbf{x}^k), I) + \|\mathbf{p}^k\|\right) - \left(\frac{\mathfrak{d}(\frac{\epsilon}{2}, \mathcal{C})}{m}\right)^2 \leq \\ d(\mathbf{x}^k, I)^2 + \frac{\left(\mathfrak{d}(\frac{\epsilon}{2}, \mathcal{C})\right)^2}{2m^2(2A+1)}(2A+1) - \left(\frac{\mathfrak{d}(\frac{\epsilon}{2}, \mathcal{C})}{m}\right)^2 \leq \\ d(\mathbf{x}^k, I)^2 - \frac{1}{2}\left(\frac{\mathfrak{d}(\frac{\epsilon}{2}, \mathcal{C})}{m}\right)^2. \end{aligned}$$

So we have finished the proof of Claim B. □

Claim C: *Assume that $A \geq \epsilon$, $d(\mathbf{x}^K, I) \leq A$, and $\rho_k < \delta(A, \epsilon, \mathcal{C})$ then for any $k \geq K$ one of the following two cases is possible:*

$$d(\mathbf{x}^k, I) \geq \frac{\epsilon}{2} \text{ and } d(\mathbf{x}^{k+1}, I) < d(\mathbf{x}^k) - \frac{1}{2}\left(\frac{\mathfrak{d}(\frac{\epsilon}{2}, \mathcal{C})}{m}\right)^2 \tag{3.8}$$

$$d(\mathbf{x}^k, I) < \frac{\epsilon}{2} \text{ and } d(\mathbf{x}^{k+1}, I) < \epsilon. \tag{3.9}$$

Proof of claim C: Let us proceed by induction on k. If we have shown the statement for k, then it implies $d(\mathbf{x}^{k+1}) \leq A$ by $A \geq \epsilon$. So we can generally assume $d(\mathbf{x}^k) \leq A$ for $k > K$.

- If $d(\mathbf{x}^k, I) \geq \frac{\epsilon}{2}$, then $d(\mathbf{x}^{k+1}, I) < d(\mathbf{x}^k)$ follows from Claim B.
- If $d(\mathbf{x}^k, I) < \frac{\epsilon}{2}$, then by Claim A and the triangle inequality, we have:

$$d(\mathbf{x}^{k+1}, I) \leq d(\tilde{\pi}_m(\mathbf{x}^k), I) + \|p_k\| \leq d(\mathbf{x}^k, I) + \frac{\epsilon}{2} \leq \epsilon.$$

Proving the claim. □

Now we come to the proof of the proposition. Since $\rho_k < \beta(A, \mathcal{C}) = \delta(A, A, \mathcal{C})$ and $\mathbf{x}^1 \leq A$, we see by Claim C, that $d(\mathbf{x}^k, I) \leq A$ for each $k \in \mathbb{N}$. Let $M(\epsilon)$ be an index, such that $\rho_k < \delta(A, \epsilon, \mathcal{C})$ for each $k > M(\epsilon)$. Therefore, again by Claim C, we see that for every positive $\epsilon < A$ if there is an index $K > M(\epsilon)$ with $d(\mathbf{x}^K, I) \leq \epsilon$ then the $d(\mathbf{x}^k, I) \leq \epsilon$ holds for each $k > K$. We only have to prove that for each positive ϵ there exists such a K. Assume the contrary. We can apply Claim C to the sequence beginning at $\mathbf{x}^{M(\epsilon)}$ again and again i times, so we have for all $i \in \mathbb{N}$:

$$d(\mathbf{x}^{M(\epsilon)+i}, I)^2 \leq d(\mathbf{x}^{M(\epsilon)}, I)^2 - \frac{i}{2}\left(\frac{\mathfrak{d}(\frac{\epsilon}{2}), \mathcal{C}}{m}\right)^2.$$

which would mean $d(\mathbf{x}^{M(\epsilon)+i}, I)^2 < 0$ after $2\left(\frac{md(\mathbf{x}^{M(\epsilon)}, I)}{\mathfrak{d}(\frac{\epsilon}{2})}\right)^2 + 1$ steps. □

Subgradient method with cylic projections	
Input:	A convex function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ and $I \subseteq \mathbb{R}^n$ which is the intersection of a nicely intersecting collection $\{C_1, \ldots, C_m\}$ of closed convex sets.
Output:	Sequence $(\mathbf{x}^n)$ in $\mathbb{R}^n$.

Let $\pi \stackrel{\text{def}}{=} \pi_1 \circ \cdots \circ \pi_m$, where π_i denotes the L_2-projection onto C_i. We define

$$n(\mathbf{x}) \stackrel{\text{def}}{=} \begin{cases} \frac{\mathbf{x}}{\|\mathbf{x}\|} & \text{if } \mathbf{x} \neq 0 \\ 0 & \text{otherwise} \end{cases}.$$

① Set $k = 0$, choose an arbitrary starting point $\mathbf{x}^0$
② Compute a subgradient $\mathbf{g}^k$ of f at $\mathbf{x}^k$.
③ $\mathbf{x}^{k+1} \leftarrow \pi(\mathbf{x}^k - \rho_k n(\mathbf{g}^k))$
④ Increase k and **go to** ②

Now we come to the main result of this section:

Theorem 3.7.4 *Let $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ be a convex function. $\mathcal{C} = \{C_1, \ldots, C_m\}$ a nicely intersecting constellation of closed convex sets in $\mathbb{R}^n$, with nonempty intersection I. Let (ρ_k) be a zero sequence of positive numbers with divergent sum. Assume that either $\lim_{\mathbf{x}\to\infty} f(\mathbf{x}) = \infty$ or C_m is bounded, then the subgradient method with cyclic projections generates a sequence $(\mathbf{x}^k)$ that has a subsequence $(\mathbf{x}'_k)$ tending to an optimum of f in X.*

Proof: Let $\pi \stackrel{\text{def}}{=} \pi_1 \circ \cdots \circ \pi_m$, where π_i denotes the L_2-projection onto C_i (which is a contraction by [Minoux 1986]). π is a contraction since it is the composition of contractions. Therefore the subgradient method with cyclic projections is a special case of the generalized subgradient method introduced at the beginning of this section. On the other hand side, note that the situation assumed here fits the conditions of Proposition 3.7.3 where the scaled subgradients $\rho_k \mathbf{g}^k$ are the perturbation vectors $\mathbf{p}^k$ and our sequence $\mathbf{x}^k$.

Claim A: $\lim_{k\to\infty} d(\mathbf{x}^k, I) = 0$.

Proof of claim A: If C_m is bounded, then the sequence $(\mathbf{x}^k)$ is bounded since $d(\mathbf{x}^k, C_m) \leq \rho_k \to 0$ and therefore we can choose K such that

$$\rho_k \leq \beta(\sup_{i\in\mathbb{N}} d(\mathbf{x}^k, I), \mathcal{C}) \qquad \forall k > K$$

and Proposition 3.7.3 implies that $d(\mathbf{x}^k, I) \to 0$.

So we can assume that $\lim_{\mathbf{x}\to\infty} f(\mathbf{x}) = \infty$. Let $\mathbf{x}$ be an arbitrary vector of I, so it is a fixpoint of π and therefore $\liminf_{k\to\infty} f(\mathbf{x}^k) \leq f(\mathbf{x})$ implying that infinitely many vectors of $(\mathbf{x}^k)$ are in

$$S \stackrel{\text{def}}{=} \{\mathbf{y} \in \mathbb{R}^n \mid f(\mathbf{y}) \leq f(\mathbf{x}) + 1\}$$

which is a bounded set and therefore

$$A \stackrel{\text{def}}{=} \sup_{\mathbf{y} \in S} d(\mathbf{y}, I)$$

is a finite value. Now Proposition 3.7.3 implies that if we choose $K \in \mathbb{N}$ such that $\rho_k \leq \beta(A, \mathcal{C})$ holds for every $k > K$ and $\mathbf{x}^K \in S$ then

$$d(\mathbf{x}^k, I) \to 0$$

proving claim A. □

Theorem 3.7.1 implies that the generated sequence $(\mathbf{x}^k)$ contains a subsequence $(\mathbf{x}'_k)$ with

$$\liminf_{k \to \infty} f(\mathbf{x}'_k) = \inf_{\mathbf{x} \in I} f(\mathbf{x}).$$

Now we show that $(\mathbf{x}'_k)$ is bounded. If C_m is compact, then $d(\mathbf{x}'_k, I) \leq d(\mathbf{x}'_k, I) < \rho_k \to 0$ implies that $(\mathbf{x}'_k)$ is bounded. So we can assume that $\lim_{\mathbf{x} \to \infty} f(\mathbf{x}) = \infty$. This means that

$$S \stackrel{\text{def}}{=} \left\{ \mathbf{y} \in \mathbb{R}^n \mid f(\mathbf{y}) \leq 1 + \inf_{\mathbf{y} \in I}(f(\mathbf{y})) \right\}$$

is bounded and missed by only finitely many elements of $(\mathbf{x}'_k)$ showing that $(\mathbf{x}'_k)$ is bounded.

So there is a convergent subsequence $(\mathbf{x}''_k)$ of $(\mathbf{x}'_k)$ with limes $\mathbf{x}''_k \in I$ and $f(\mathbf{x}) \leq \liminf_{k \to \infty} f(\mathbf{x}''_k) = \inf_{\mathbf{x} \in I} f(\mathbf{x})$, hence $\lim_{k \to \infty} \mathbf{x}''_k$ is a minimum of f over I. □

Note that the subgradient method with cyclic projections generalizes the ordinary constrained subgradient method: the case of $|\mathcal{C}| = 1$. It can also be regarded as a generalization of the method of cyclic projections: the case of constant f.

Moreover it gives rise to a projection algorithm to determine the projection of vector $\mathbf{x}$ onto the intersection of a nicely intersecting constellation of closed convex sets assuming that we can easily project to any of the sets by applying the method to the objective function

$$f(\mathbf{y}) \stackrel{\text{def}}{=} d(\mathbf{x}, \mathbf{y}),$$

This method is more general than that of in [Dykstra 1983] and [Han 1988], since it can be used with the distance function of an arbitrary norm not only with that of $\|.\|_2$. The result in [Dykstra 1983] and [Han 1988] has been

genaralized to Hilbert spaces in [Boyle and Dykstra 1986], but it still assumes that the norm to be minimized is induced by a scalar product. Our algorithm does not need this assumption, since the objective function can be an arbitrary convex function, which is true in the case of arbitrary norms.

Chapter 4

Weighted Laplacian in Chip Design

4.1 Introduction

Several interesting large-scale combinatorial and geometric optimization problems arise from the *physical design of very large scale integrated cicrcuits*. Here we will focus on one of the most central problem: the combined performance constrained placement and timing optimization problem, the goal of which is to determine positions and sizes for the gates of a chip so that the wire-length is minimized subject to additional constraints. In this exposition we will only focus on the constraints coming from timing requirements.

There are two main sections: methods for optimizing the placement subject to timing restriction, the other is the gate- and wire-sizing. Eventually, we will discuss the problems arising from their unification.

For the performance driven placement, we will present an approach based on the subgradient method in which the weighted Laplacian is the central object at two completely different places. The quadratic form describing the intermediate objective function is given by a weighted Laplacian matrix of the *gate graph*. On the other hand, the weighted Laplacian of the same graph is a central object of the algorithm for projecting the dual variables. Although the graphs are the same, the weights differ and the weighted Laplacians occur in completely different contexts, but play central roles in the two most essential parts of the algorithm.

The nodes of the gate graph are the small elementary building blocks the circuit is built from. Two nodes are incident if the gates are connected by a wire. The weighted Laplacian of this graph is the matrix of the quadratic form of the overall weighted quadratic arc-length of the chip. (This model subsums the well-known clique modell). The minimization of this function is equivalent to solving a linear system of inequalitities involving the weighted Laplacian. Using appropriate weights to improve the speed of the chips is an old an established approach ([Burnstein and Youssef 1985] and [Tsay and Koehl 1991]) However the choice of the weights remained mostly heuristical, and the question was handled purely experimentally. The present thesis puts this problem into a mathematically well-founded framework which allows for generating weights in a manner which results in a method with provable convergence to an optimal solutions: the netlength is minimized while meeting all timing constraints while the disjointness of the cells is ignored. The main tools in our treatment are Lagrangian duality and the subgradient method. Similar methods were already applied to the problem of gate- and wire-sizing in [Chen, Chu and Wong 1999].

We will discuss the Lagrangian relaxation and constrained subgradient method based approach for the gate- and wire-sizing as well. The computation of the subgradient consists of the solution of another constrained minimization problem for a weighted sum of area and delay. Several authors (e.g. [Chu and Wong 1999]) have proposed a simple local optimization method, the so called *local refinement* (or *cyclic coordinates method*) to solve this and similar problems. In fact the linear convergence of this method follows from the general result of [Luo and Tseng 1992]. However, [Chu and Wong 1999] have given explicit error estimations and a new proof for this the special case of timing graphs with tree topologies. Their result has been further extended in [Langkau 2000]. In section 4.4.8, we will present a simpler proof and similar error estimations for a much more general case.

An important constituent of the Lagrangian-based subgradient method is a projection step to the space of nonnegative flows on the timing graph. The most successful exact algorithm for solving this problem is a generalized newton method applied to the dual problem. The complication is that the dual function is not always twice differentiable. The fastest known method for this problems in [Ibaraki, Fukushima and Ibaraki 1991] approximates the Hessian by suitably weighted Laplacians.

However, we have seen in Section 3.5 that an exact projection is not needed. We can successively project to the flow space and then to the quadrant of

nonnegative vectors which can be performed much more efficiently while still getting the same guarantees for convergence as before.

First, we will consider the problem of placing the gates while neglecting the disjointness constraints. Fortunately, the method can be easily integrated with tools that ensure disjointess of the placement. Of course, in this case optimality can not be guaranteed anymore.

Another important extension is the integration with gate- and wire-sizing. Since both methods use the same framework, they can be integrated without problems. Altough both of them are convex, the combined problem is nonconvex and it is unclear whether it can be transformed to a convex one. It is also an open question whether the combined method converges to some optimum.

4.2 Basics and Notation

In order to specify the arising optimization problems and their solutions exactly, we will need formal notation of the instances occuring in the physical design process of chips.

4.2.1 The Netlist

A chip is typically composed of a large number of small building blocks: so called **gates**. They are to be connected by wires. At the start of the physical design the physical layout of wires and gates is not fixed, just the abstract topology of the network. Each gate has a prescribed number of input and output **pins**. They are the connection points for the wires. At the end of the design process, the wires are realized by metal shapes depending on the physical layout of the gates chosen in the layout phase. However, the number and logical context of the pins is given at the start of the physical design. The pins are partitioned into classes called **nets**. Two pins being in the same net means that they must be connected by a wire in the resulting design, whereas pins in different nets are not allowed to be connected electrically, otherwise logical failures or short circuits may arise. Normally each net has a **driving** (**source**) pin where the electric signal starts and several **driven** (**sink**) pins where the signals enter a new gate or leave the chip. The driving (driven) pins of a gate are called **output** (and **input**) pins of the gate. The **fanout** of the source pin of a gate is the set of the driven pins in the same net. Sometimes we also use the same word to denote the size of this set.

Therefore, the netlist can be modelled by two hypergraphs on the set of pins: the set system of the gates, and that of the nets. One must note that current physical design systems change the netlist of the circuits while maintainig the invariant of logical equivalence to the reference netlist.

4.2.2 An Overview of Objectives

The objectives of the physical design process are manifold: first of all: the gates must be placed disjointly so that they fit into the prescribed area. The wires must be realized correctly, without short circuits, too. Moreover, the electric signals must obey a large number of timing specifications: the chip must run at specified frequencies while communicating with its neighbourhood within specified response times and signal shapes. Besides that, the yield (the percentage of functioning chips on a wafer) is to be maximized and power consumption is to be minimized.

This large number of sometimes conflicting objectives and constraints is impossible to be optimized and met in one single optimization pass. Therefore the physical design is typically divided into parts, each of them optimizing a different aspect of the chip. However, simultaneous optimization of different goals has the advantage of producing higher quality solutions in general.

Perhaps the most serious bottleneck in the VLSI physical design process is timing closure, that is producing a layout meeting the timing specifications. On easy chips this is a simple routine, but on more complicated designs it can involve a lot of manual work, resynthesis of whole parts of the logic or even reconsidering the specifications.

In our work, we will concentrate on methods improving the timing behaviour of the chip that are based on solely manipulating the physical layout of the given netlist without changing the graph significantly. Most notably:

- placement of gates,
- changing the size of gates
- changing the size, type and physical layout of wires
- insertion of repeater trees.

Up to the last operation, the insertion of buffering trees, the above operations do not change the netlist. However as feature sizes continue to shrink, the insertion of repeaters gains on importance. Nearly optimal buffering can be viewed as an indispensible and necessary constituent of the layout process. Moreover, buffering today tends to be viewed as part of the routing problem.

In the routing phase, nets are mainly realized in a 3-dimensional grid. Since current chips contain only a small number of routing layers, plane L_1-Steiner trees approximate their length and timing behaviour acceptably. However even neglecting disjointness, the optimization of the Steiner netlength is a hard task because the length of a Steiner tree is a non-convex function of the terminals. Therefore the common idea is to use less accurate but convex netlength estimations.

4.2.3 Netlength Estimations

For worst case ratio comparisons between different netlength estimations the reader is referred to [Brenner and Vygen 2002]. Here we only give a very short overview relevant to our exposition.

The following net-models are typically used:

(1) shortest L_1-Steiner tree (nonconvex)

(2) shortest spanning tree (nonconvex)

(3) bounding box (convex)

(4) symmetric star model (convex)

(5) asymmetric star model (convex)

(6) clique model (convex)

The most accurate one is the *L_1-Steiner-tree model*. This means that the length of the nets estimated by the length of a shortest L_1-Steiner tree on its terminals. Its computation is *NP*-hard in general [Garey and Johnson 1977], but there are very efficient exact [Warme, Winter and Zachariassen 2000] and approximation algorithms [Arora 1996] that are fast enough for all sizes relevant in chip design (that is up to 10000 terminals). Moreover, very well working almost linear heuristics based on the sweepline method are known for computing Steiner trees on larger instances too.

The computation of the *shortest (L_1 or L_2) spanning tree* on the terminals is possible in almost linear time in their number. Since its length is also nonconvex in the coordinates of the terminals and the computation of the Steiner is similarly fast, this estimation in not used frequently in practice.

The *bounding box estimation* is the perimeter of the L_1-bounding box of the terminals.

The *symmetric star model* is the length of the segments of the star on the terminals and centered in the midpoint of them. The quadratic star netlength is the sum of the squares of the length of the involved segments.

The pins of nets play different electrical roles: there is a driving (source) pin, where the electrical signal comes from and one or more driven (sink) pins where the signals enter the gates. Therefore the length of the star on the sinks, centered at the source is also a natural choice for netlength estimations. This is the *asymmetric star model*.

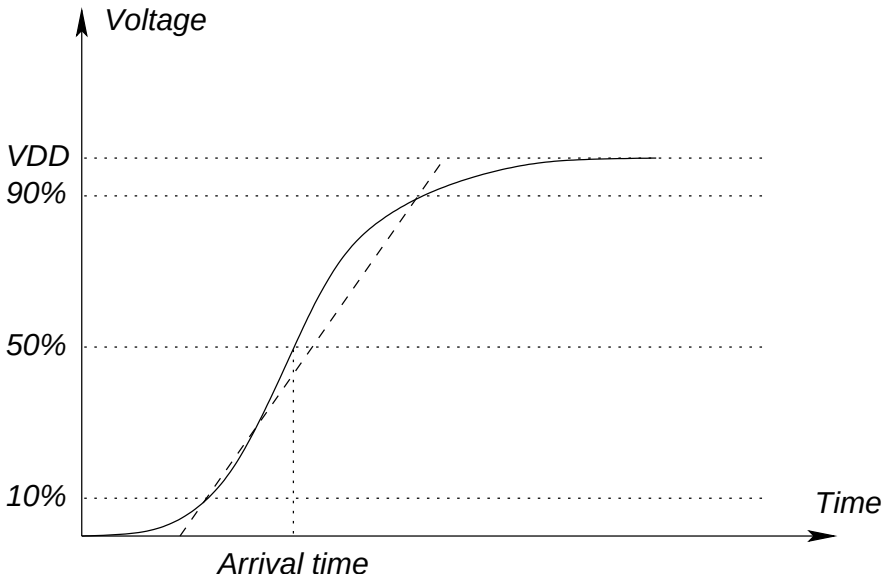
The *clique model* is simply the sum of the squares of L_2-length of all line segments connecting any two pins of the net. This netlength estimation is $(n-2)s(n)$ where n is the number of the terminals and $s(n)$ is the symmetric quadratic L_2-star netlength. It means especially that it can be computed in linear time in the number of terminals. It is widely used in quadratic placement algorithms. This fact, the above identity and its easy computability may explain its popularity.

4.2.4 Signals and their Shapes

An *electrical signal* is simply a function of voltage in time. The timing analyis of a design consists of the estimation of the arrival time and shape of the electric signals at different points of a design.

The exact simulation [Vladimirescu 1994] of the electric signals even on small networks is a difficult and extremely time consuming task. Although accurate methods are used for verifying the expected timing behaviour of critical parts of a design, for example the clock tree, it would not make much sense to use such exect methods at the beginning of the physical design process. Besides that, an exact simulation of *combinatorial circuitry* works only for a selected set of patterns of input signals. The number of such patterns grows exponentially with the number of possible input bits. This makes an exhaustive analysis of the timing behaviour of a chip practically infeasible.

A possible way to perform a crude analyis and optimization is to neglect signal shapes completely and work with arrival times only. This approach is usable and useful in the synthesis phase, but it tends to be too inaccurate even at the beginning of the physical layout.



A well established middle ground is to take the shapes of signals into account but approximate them with linear functions. Altough the form of real signals may be quite diverse, there are two basic shape types: the rising and the falling ones. A signal is called rising (or falling) if its voltage/time function is roughly monotone rising (or falling), respectively. However it also matters a lot how fast the signals rise and fall. This can be quantified by the slope of the approximating linear function (the dashed line in the picture). This quantity is commonly called **slew**. The arrival time of a signal is defined to be the time point where the voltage reaches half of the Vdd (the supplied voltage), this is only a question of convention and equally good analysis and optimization could be done by adopting a different one. The main point is that the space of possibly occuring signals is reduced from an infinite dimensional to a two dimensional one.

4.2.5 The Basics of Timing Analysis

The gates of chips can be partitioned into three different classes according to their function:

- Combinatorial logic
- Storage elements
- Clocking gates

Although, most computation on a chip (like adding two numbers or decoding an instruction) is performed by the combinatorial logic, large parts of a typical design are spent on **storage elements** or **memory cells**. They are capable of holding information for a limited amount of time, then this information (typically a single bit) is released in form of an electrical signal and fed back to the combinatorial logic or leaves the chip. After releasing it, the storage element is put in a state that allows for receiving a new bit of information. Clearly, this receive/release cycle must happen in a coordinated manner, otherwise predictible complex operations on more than one bit would be very hard to design.

The synchronisation of the memory cells is the task of the clock trees. A clock tree is a large network consisting of wires and repaters distributing the same periodical signal to all memory cells belonging to the same *clock domain*. Of course, a chip may have different clock trees with different frequencies, even completely asynchronous ones. Signals released from a memory cell may be fed into cells clocked at different (but possibly synchronized) frequency.

Fact of life is that it takes some time until a signal released at pin p arrives at pin p'. Even the shape of the signal changes during the process. Signal propagation through gates takes time too. The task of timing analysis is to tell whether all signals arrive in time. Typical constraints are:

- Each signal arrives at the respective memory cell early enough before it ceases to receive information. (**setup test**)
- Each signal remains stable at the end of the receiving stage for a given amount of time before the latch ceases to accept information. (**hold test**)
- Some signals arrive approximately synchronously at a specified place.
- Signals communicating with the outside world do it respecting some imposed constraints. (**I/O timing**)

For the arrival times and shapes of signals arriving at the inputs of the chip lower and upper bounds are assumed.

Note that a setup test imposes an upper bound on the arrival time of the signals while hold tests specify lower bounds. Since for checking the upper bound only the latest possible relevant signal is interesting it is also called a **late mode test** or **late mode constraint**. Similarly, the notion **early mode test** or **early mode constraint** indicates that a lower bound is imposed on the arrival time of some signal. Synchronicity constraints can be viewed as simultaneously imposed late- and early-mode constraints.

4.2.6 Delay Models in Timing Analysis

In order to perform timing analysis, the transformation of the signal shapes must be computed for each time a signal passes a gate or a piece of interconnection.

The simplest timing model which disregards the shapes of the signals and assumes uniform delay for each stage of logic is not used at all for its unpreciseness. Even in the synthesis, fanout based rules are used, that is they take the fact into account that an increase in the number of driven pins or gates makes the driving stage slower.

The gain based method still ignores the shapes, but takes into account that the gates may have different sizes. Making a gate larger makes it faster but slows down the the stages driving it. This approach assumes a linear behaviour in any case. It is fast to compute and sufficiently exact in a limited range, especially for synthesis.

The cell library based approach uses a discrete set of carefully designed realization of the gates. The **cells** of the library are partinoned into logically equivalent classes. The cells of the same class represent logically equivalent but physically different building blocks. The timing behaviour of the cells is represented by a directed graph There are (typically piecewise polynomial) transformation functions attached to each arc describing the characteristic the output signal in dependence of the input signals at the tail.

The signals also change when running through the nets. There are estimation of different accuracy to compute this transformation. In the synthesis the net delays are typically neglected or assumed to be constant. In the initial late mode optimization typically the moderately accurate, but very simple Elmore delay model [Elmore 1948] is used. For fixed wire width and plane,

this is a quadratic function for a two-point connection. For a nets with more than 2 pins, it can be very efficiently computed in linear time of the number of pins. In later design phases, delay models based on poles and residues [Ratzlaff and Pillage 1994] are used. Post routing checking may use even more exact simulation for the most critical nets.

4.2.7 Static Timing Analysis

Still, we can see an important property of the verificaton: *The timing behaviour of all possible signal paths must be checked in order to verify the correct functioning of a design.* The problem is that the number of such paths is exponentional in the number of gates and nets. A complete enumeration of possible paths is computationally infeasible even for small graphs.

Fortunately, there is a commonly applied solution to this problem. It is based on the following idea: to check the late mode (upper bound) constraints at a pin, only the latest arriving signals are relevant. If we assume that the shape of the signals does not influence the arrival time dramatically, or that the shapes of the signals arriving at a given point are similar, then we can assume that a signal arriving earlier will never become the latest one anymore, since they must run through the same subsequent stages as the later ones.

This means that for checking the late mode constraints, the earlier arriving signals are irrelevant and can be eliminated. The same idea applies to the early mode constraints. Therefore, in order to check both type of constraints for a specific design, we must propagate only the earliest and latest arriving signal for each pin to verify its functioning.

This idea immediately gives rise to an algorithm which verifies the timing feasibility of a physical layout of the chip in linear time in the number of gates and nets.

First of all, note that nets and gates implicitly specify a signal propagation direction. In the gates, the signals propagate from the inputs to the outputs, in the nets they propagate from the source to the sinks.[1]

So a directed graph (**timing graph**) can be extracted from the netlist which describes the propagation of the possible electric signals on a chip.

[1]Under exceptional circumstances multi-directional and multi-source nets are possible. They are in minority and do not pose any theoretical difficulty and would unnecessarily complicate this simple introduction

Definition 4.2.1 (Timing Graph) *The timing graph of a chip is a directed graph on the pins of a design. It arises from the netlist using the following two substitutions:*

- *For each net with driver pins D and sink pins S, add all arcs in $D \times S$ to the timing graph.*
- *For each gate with input pins I and output pins O add all arcs in $I \times O$ to the timing graph.*

To get the main idea of the static timing analysis, let us consider an acyclic subgraph G' of the timing graph without isolated nodes. The nodes of this graph are pins which can be partitioned into three disjoint classes:

- **Left boundary**: the nodes without incoming arcs.
- **Right boundary**: the nodes without outgoing arcs.
- **Internal nodes**: the nodes with both incoming and outgoing arcs.

We can assume that we have bounds for the arrival time and shape of the electric signals arriving at the boundary nodes. The static timing analysis processes the nodes from the left to the right, that is the nodes are sorted topologically and when node v is starting to be processed, all its predecessors are already processed. To process v, its incoming arcs are considered and propagate, which means that lower and upper bound for the arrival times and shapes of the electric signals are computed at the head. These bounds depend on the bounds already computed at the tail and the timing model describing the signal transformation over the arc. After all signals are propagated to the head, the latest and earliest ones are kept, all others are thrown away as being irrelevant.[2] At the right boundary, there are external constraints. A design is feasible if these constraints are weaker then the propagated ones, otherwise a **timing violation** is detected and must be eliminated via design changes.

Of course, the timing graph is not acyclic in general. A typical source of cycle is the presence of clock-gating, that is if the combinatorial logic changes the clocking behaviour of the memory cells. However, cycles normally can be treated by a two level approach: a (typically small) subset of arc is snipped (deleted

[2]If signal shapes play an important role, there may be several possible candidates of worst signals. To solve this situation without propagating too many candidates, different pruning techniques were proposed in [Lee et al. 2001] and [Vygen 2001]

temporarily) from the timing graph so that it becomes acyclic. Heuristically computed signal bounds are assumed at the incident nodes of the snipped arc and the design is propagated and checked via the above algorithm. Then, for each snipped arc the constraints are checked again. Of course, its success depends on how optimal the initial bounds on the arc ends were chosen at the beginning. If the method exits without reporting violation then there are no problems, but if the initial bounds at the signal end were chosen unluckily, the algorithm might cause false alarms. We note that the algorithm may be refined so that it correctly checks the design, however it may require succesive repropagation of larger parts of the design and introduce some runtime overhead.

4.2.8 Overview of the Design Flow

Until now, we treated both type of bounds symetrically. However, lower bounds on signal arrival times are normally much easier to be optimized than upper bounds (that is: it is easier to make signals slower than faster). Therefore, the typical procedure in timing optimization is to meet the late mode constraints first. The early mode violations are eliminated at the end in a different phase. Experience shows that they can be eliminated without introducing any new late mode violations (an industrially applied solution is described in [Schietke 1999]).

In physical design the following approach proved to be succsessful: In the first phase, the clock trees are not built at all. Either prescribed clock arrival times at the memory cells are specified, or the clock arrival times are left completely variable (the latter is a more modern approach enabled by a new generation of clock tree design tools that allow for realizing almost all reasonable target arrival time vectors at the clock inputs of the memory cells (cf. [Maßberg 2002] and [Held et al. 2003])). The chip is placed and the late mode constraints are optimized with less accurate timing estimations. Then the clock arrival times at the latches are optimized ([Albrecht 2001],[Albrecht et al. 2002],[Held 2001] and [Held et al. 2003]) and the clock trees are built. This phase already solves most early mode problems. Then the chip is timed using more accurate timing models and sophisticated clock analysis tools. Remaining late and early mode problems are solved by slower but more aggressive post-optimization tools. If the routability is bad or there are many timing violations, then the design cycle is repeated with different parameters. It is possible that parts of the netlist have to be resynthesized completely, or deep logic changes have to be

performed manually. If the timing is clean, then the chip is routed which can introduce new timing violations. These are typically solved manually or using special purpose post-optimization tools.

4.3 The Timing Driven Placement Problem

A crucial phase of the design cycle is the initial late mode optimization. The quality of the solution produced in this step is essential for the success of the whole layout process.

First, we will look at the problem of computing a power optimal placement meeting all late mode timing constraints while all other parameters (gate sizes, wire sizes, etc.) are fixed. We assume that only the placement of the gates is variable (up to some prescribed ones), all other parameters are fixed. The objective is to generate an overlap-free, routable (i.e. where each net can be disjointly realized by wires) placement of the gates so that all late mode timing constraints are met.

The basic idea [Wipfler, Wiesel and Mlynski 1983] of producing such placements is to compute a solution with minimal netlength neglecting the disjointness constraint. Then, in the so called partitioning step, the gates are moved away from the overpopulated regions and assigned to new regions that are placed recursively in the subsequent steps. This is a standard approach adopted by many authors ([Kleinhans et al. 1991],[Vygen 1996] and [Alpert et al. 1997a]) and used in industrial tools.

Of course, this way neither routability nor the timing constraints are taken into account. In this work, we will consider only the late mode timing constraints and leave routability untreated. Although the minimization of the netlength guarantees a reasonably well routable placement, harder designs require special care for congested areas. Since the approach presented in [Brenner and Rohe 2002] affects solely the partitioning step, it can easily be combined with the methods discussed here.

4.3.1 Problem Formulation

Let P denote the set of finite tuples of points in the plain $\mathbb{P} \stackrel{\text{def}}{=} \mathbb{R}^2$. A function $f : P \longrightarrow \mathbb{R}_{\geq 0}$ is called **convex** if for any $n \in \mathbb{N}$, f restricted to $\{v \in P \mid |v| = n\}$ is a convex function of the vector of coordinates of the points in the tuple.

Let d be some prescribed distance function on the points in the plane and $\mathbf{t}, \mathbf{t}' \in \mathbb{P}^n$. We say that $\mathbf{t}'$ **dominates** $\mathbf{t}$, if for each pair of indices $i < j \leq n$ holds: $d(\mathbf{t}_i, \mathbf{t}_j) \leq d(\mathbf{t}'_i, \mathbf{t}'_j)$.

Definition 4.3.1 (Distance monotony) *A function $f : P \longrightarrow \mathbb{R}_{\geq 0}$ is called* **distance monotone** *if and only if for all pair $(\mathbf{t}, \mathbf{t}')$ of tuples of the compatible size, if $\mathbf{t}'$ dominates $\mathbf{t}$, then $f(\mathbf{t}) \leq f(\mathbf{t}')$.*

Now we describe the input of the timing driven placement problem: We are given an acyclic timing graph. The nodes represent the outputs (sources) of the gates. Each arc represents an immediate data path (which is disjoint to all other gates) from one source of a gate to the source of another gate.

We are also given a set of gates C. Each node v of graph G is assigned a gate $\gamma(v)$ in order to allow multisource gates. This leads us to the notion of gate graph:

Definition 4.3.2 (Gate graph) *A gate graph $G = (V, E, C, \gamma)$ consists of directed graph (V, E), set of gates C and a surjective gate assignment $\gamma : V \longrightarrow C$.*[3]

To simplify our notion we have avoided standalone primary inputs/outputs which can easily be modelled by dummy gates.[4]

From now on, we will assume that we are given fixed convex, distance-monotone netlength and wiring delay estimation $l, l' : P \longrightarrow \mathbb{R}_{\geq 0}$. Note that each convex netlength estimations introduced in Section 4.2.3 is distance-monotone.

Some of the gates $C' \subseteq C$ are preplaced: we are given $\mathbf{p} \in \mathbb{P}^{C'}$. The task is to extend this vector to $\mathbf{p} \in \mathbb{P}^C$ minimizing the objective function (the sum of (quadratic) netlength). The complement $C \setminus C'$ is denoted by C_m which is called the *set of movable gates.*

For the arcs in E, we are given intrinsic delays $\mathbf{i} \in \mathbb{P}^E$ and positive gate and net resistances $\mathbf{r}^g, \mathbf{r}^n : \mathbb{R}^E_{\geq 0}$. Note that intrinsic delays are allowed to be negative which allows for modelling non-transparent arcs, those that are not propagated, but contribute to the objective function (sum of netlength).

[3]Note that in the case of single source gates, γ is superflous and the gate graph is simply a graph on the set of gates as nodes.

[4]The careful reader may notice that we do not allow timing pins "inside" gates. At the first sight this may look like a restriction, but our approach and results can easily be extended to that situation, but "internal timing pins" induce superflous complication of formalism. In fact, the current BonnTime implementation deals with gates containing internal pins without problems.

Furthermore, there is a vector of relative offsets $\mathbf{o} : \mathbb{P}^E$. In order to interpret the gate graph model, one has to understand thet there is a one-to-one correspondece between the arcs and the sink pins of the design. The offset of an arc is the difference of the offset of the tail node relative to its gate and the offset of the sink pin relative to the its gate (i.e. that of the head). More formally, for a given placement $\mathbf{p}$, the geometric configuration of the pins of the net rooted at source u can be translated to the *wiring configuration* w:

$$w(u, \mathbf{p}) \stackrel{\text{def}}{=} (\mathbf{p}_{\gamma(u)}, \mathbf{p}_{\gamma(e_1^+)} + \mathbf{o}_{e_1}, \ldots, \mathbf{p}_{\gamma(e_k^+)} + \mathbf{o}_{e_k}),$$

where $\{e_1, \ldots, e_k\}$ is the set of arcs with tail u.

Delay $d(e, \mathbf{p})$ over arc e with repect to placement $\mathbf{p}$ is defined as follows:

$$d(e, \mathbf{p}) \stackrel{\text{def}}{=} \mathbf{i}_e + \mathbf{r}_e^g l(w(e^-, \mathbf{p})) + \mathbf{r}_e^n l'(w(e^-, \mathbf{p})).$$

Note that this delay function is convex in the placement $\mathbf{p}$ of the gates, since it is a nonegative linear combination of compisitons of convex and linear functions ($\mathbf{p}_g$ is linear in $\mathbf{p}$).

Besides, we have prescribed arrival times on the boundary V'. These are the nodes which have either no entering or no leaving arc. Because of the structure of our constraints, this is equivalent to specifying lower limits on the arrival times at the sources of the graph and upper limits on the sinks. The set $V_p \stackrel{\text{def}}{=} V \setminus V'$ is called the set of **propagated**[5] nodes.

The careful reader may have noticed that the delay functions ignore the input pin capacitances. In fact, the contribution of the input pin capacitances to the delay can be computed in advance and added to the intrinsic delays. Therefore we omitted them to simpify our model.

To be found is a placement extending the preplacement, minimizing the (quadratic) netlength such that there exists a corresponding arrival time assignment respecting the delay inequalities.

Now we summarize the definition of our problem:

[5]Note that we do not make a distinction between prescribed and required arrival times. So nodes without leaving arcs with asserted required arrival times are not *propagated* by this terminology. This may seem piculiar at the first sight, but it is a consistent and usable point of view.

PLACEMENT WITH TIMING RESTRICTIONS

Instance:

- A gate graph $G = (V, E, C, \gamma)$. Let V' denote the set of nodes without either entering or leaving nodes in G.
- Convex and distance monotone netlength and wire-delay and objective estimations $l, l', l'' : P \longrightarrow \mathbb{R}_{\geq 0}$. We will further assume that l'' is smooth and strictly convex.
- A subset $C' \subseteq C$ and preplacement $\mathbf{p}' \in \mathbb{P}^{C'}$.
- Pin offsets $\mathbf{o} \in \mathbb{P}^E$.
- Gate- and net-resistances $\mathbf{r}^g, \mathbf{r}^n \in \mathbb{R}^E_{\geq 0}$.
- Intrinsic delays $\mathbf{i} \in \mathbb{R}^E$.
- Prescribed arrival times $\mathbf{a}' \in \mathbb{R}^{V'}$.

Goal: Find a placement $\mathbf{p} : C \longrightarrow \mathbb{R}^2$ extending the preplacement $\mathbf{p}'$ and corresponding arrival times $\mathbf{a}$ extending the prescribed arrival times $\mathbf{a}'$ minimizing the overall netlength

$$f(\mathbf{p}) \stackrel{\text{def}}{=} \sum_{v \in V} l''(w(v, \mathbf{p}))$$

subject to the timing constrains
$\mathbf{a}_{e^-} + d(e, \mathbf{p}) \leq \mathbf{a}_{e^+}$ for each arc $e \in E$.

Note that the objective function is convex and strictly convex in $\mathbf{p}$ and each constraint defines a convex area in the solution space $\mathbb{P}^{C_m} \times \mathbb{R}^{V_p}$. Therefore, each local minimum of the function is also a global minimum. Moreover, one can check that the placements of all optimum solutions are inside some offset adjusted bounding box (the bounding box extended by the maximum ocuring offset in each direction) of the preplaced gates. On the other hand, the delay inequalities imply that the arrival time vector $\mathbf{a}$ satisfies

$$\mathbf{a}_u + \sum_{e \in I} \mathbf{i}_e \leq \mathbf{a}_v,$$

if I is an arbitrary (u, v) path and u a source of the timing graph with prescribed arrival time. Similarly

$$\mathbf{a}_v \leq \mathbf{a}_u - \sum_{e \in I} \mathbf{i}_e$$

for an arbitrary v-u path I with sink u. This shows that all feasible solutions are located in an easy to compute compact box of the solution space.

The convexity of the problem motivates the usage of standard convex optimization methods introduced in Chapter 3

4.3.2 Overview of the Algorithm

To solve the timing driven placement probmem, we will apply the constrained subgradient method from Section 3.3 to the Lagrangian dual of the objective function. We will also see that the version with cyclic projections are useful in this situation.

Let f be the objective function of the timing driven placement problem. We assume that it arises from an arbitrary distance monotone convex netlength function. Later we will focus on the case of quadratic L_2 netlength.

First, dualize all constraints of the problem, so the Lagrangian function becomes:

$$Ł(\mathbf{a}, \mathbf{p}, \boldsymbol{\lambda}) = \sum_{v \in V} (l''(w(v, \mathbf{p})) + \sum_{e \in E} \boldsymbol{\lambda}_e (\mathbf{a}_{e^-} + d(e, \mathbf{p}) - \mathbf{a}_{e^+}) \tag{4.1}$$

The dual objective function is:

$$D(\boldsymbol{\lambda}) = \inf_{(\mathbf{a},\mathbf{p}) \in X} Ł(\mathbf{a}, \mathbf{p}, \boldsymbol{\lambda}),$$

where the infimum is taken over all arrival times and placement in compact box X which is known to contain the optimal solution. We also know that maximum of D over the nonnegative orthant is less or equal than the optimal value of the primal objective function (cf. Theorem 3.2.1).

We will assume that there is a feasible solution of the problem for which all delay constraints are satisfied with inequality. Since the objective function is assumed to be strictly convex and all constraints are linear inequalities, the Slater conditions (cf. Theorem 3.2.2) are fulfilled for this problem, implying that the duality gap is zero:

$$\sup_{\boldsymbol{\lambda} \in \mathbb{R}^E_{\geq 0}} D(\boldsymbol{\lambda}) = \inf_{(\mathbf{a},\mathbf{p}) \in X} f(\mathbf{a}, \mathbf{p}). \tag{4.2}$$

So we can apply the constrained subgradient algorithm from Section 3.3 to maximize $D(\boldsymbol{\lambda})$. In each step, we compute the duality gap, providing us not

only with a good stopping criterion, but also with an upper bound on the maximum distance from the optimum.

Of course, the essential work is to compute the dual function $D(\boldsymbol{\lambda})$ in each iteration of the algorithm. This is in general a hard task. However, in our special case, the Langrange function can be separated and minimized reasonably efficiently. This same idea was proposed in [Chen, Chu and Wong 1999] to solve the gate sizing problem.

4.3.3 Separating the Lagrange Function

The Lagrange function (4.1) can be separated:

$$Ł(\mathbf{a}, \mathbf{p}, \boldsymbol{\lambda}) = Ł_1(\mathbf{a}, \boldsymbol{\lambda}) + Ł_2(\mathbf{p}, \boldsymbol{\lambda}),$$

where

$$Ł_1(\mathbf{p}, \boldsymbol{\lambda}) = \sum_{e \in E} \boldsymbol{\lambda}_e d(e, \mathbf{p}) + \sum_{v \in V} l''(w(v, \mathbf{p}))$$

and

$$Ł_2(\mathbf{a}, \boldsymbol{\lambda}) = \sum_{e \in E} \boldsymbol{\lambda}_w (\mathbf{a}_{e^+} - \mathbf{a}_{e^-}).$$

It is easy to see, that for a particular $\boldsymbol{\lambda}$, the minimum of $Ł_2(\mathbf{a}, \boldsymbol{\lambda})$ is minus infinity unless $\boldsymbol{\lambda}$ satisfies the flow equality for the edges incident with each node with variable arrival time. Otherwise one could choose the arrival time at some violating node so that $Ł_2(\mathbf{a}, \boldsymbol{\lambda})$ becomes arbitrary small. On the other hand it is is a finite constant value if $\boldsymbol{\lambda}$ forms a flow on the propagated nodes of the timing-graph. Therefore, in order to optimize D by the constrained subgradient method, we are allowed to constrain $\boldsymbol{\lambda}$ to the convex set of nonnegative flows. In order to get the constrained subgradient algorithm from Section 3.3 work, we will have to project $\boldsymbol{\lambda}$ to the convex set of nonnegative flows on the arcs of the timing graph in each iteration.

4.3.4 Projection of the Lagrangian Multipliers

PROJECTION SUBPROBLEM

Instance:

- A graph $G = (V = V_p \cup V^- \cup V^+, E)$ without isolated nodes, where

$$V^- \stackrel{\text{def}}{=} \left\{ v \in V \mid \deg^-(v) = 0 \right\},$$
$$V^+ \stackrel{\text{def}}{=} \left\{ v \in V \mid \deg^+(v) = 0 \right\} \qquad \text{and}$$
$$V_p \stackrel{\text{def}}{=} V \setminus (V^- \cup V^+).$$

- Arc-weighting $\boldsymbol{\lambda} \in \mathbb{R}^E$.

Goal:: Find a nonnegative weighting $\boldsymbol{\lambda}' \in \mathbb{R}^E_{\geq 0}$ of the arcs minimizing

$$\|\boldsymbol{\lambda} - \boldsymbol{\lambda}'\|_2^2$$

subject to the flow conservation equalities on V_p:

$$\sum_{\substack{e \in E \\ e^- = v}} \boldsymbol{\lambda}'_e - \sum_{\substack{e \in E \\ e^+ = v}} \boldsymbol{\lambda}'_e = 0 \qquad \forall v \in V_p.$$

The computation of a projection to the set of nonnegative flows is a non-trivial task regarding that the gate graphs occuring in the layout process of todays chips have several millions arcs. The projection problem is well known from combinatorial optimization as the *minimum cost flow problem with quadratic cost function*. A polynomial algorithm based on scaling and successive piecewise linear approximation and run-time $O(c(\epsilon)|E||V|^2)$ was given in [Minoux 1984]. It is the special case of the more general class of *minimum cost flow problems with convex separable cost function* which have been studied extensively. The practically most efficient algorithm by [Ibaraki, Fukushima and Ibaraki 1991] is based on the generalized Newton method where the Hessian is replaced by approximation matrices which happen to be weighted combinatorial Laplacians of the timing graph with suitable weights.

However, exact projections have turned out to be computationally expensive in practice. Keep in mind that the projection is required only to keep the

Lagrangian multipliers (that measure the timing criticalities of the arc) near to the feasible region. The "real" optimization of the placement happens while solving the *placement subproblem* described in the next section. Since the superlinear scaling of the projection steps, it tends to dominate the run time of the whole algorithm even for moderately large designs. This problem has been noticed in [Chen, Chu and Wong 1999] since the projection was also used to solve the gate sizing problem by a similar method. They observed that the practical run time of the algorithm was about $O(|V|^{1.7})$ for the whole subgradient method. This is acceptable for instances up to 100000 nodes, but after that the run time degrades considerably. To cure this problem, several authors ([Muuss 1999] and [Sechen and Tennakoon 2002]) proposed some heuristics with linear run time but without theoretical guarantees of convergence.

This was the motivation to study a different variant of the constrained subgradient mehod in Section 3.5. The main idea is to project $\boldsymbol{\lambda}$ to the whole flow space and after that to the set of nonnegative vectors. Since both sets are polyhedral, we get by Theorem 3.6.8 that they intersect nicely and therefore by Theorem 3.7.4 the method described here converges as well.

Of course, it is to be demonstrated that $\boldsymbol{\lambda}$ can be very efficiently projected to the whole flow space. Here is the place where the combinatorial Laplacian enters the picture again.

One problem is that we have only a subset of nodes V_p for which the flow equalities have to hold. To make the problem more homogenous, we augment G by a dummy supernode s representing the reference 0 point of time. Additionally, for each node $v \in V^-$ an arc from s to v and for each node $v \in V^+$ an arc from v to s is added to the graph. In order to eliminate the arrival time constraints for the nodes in $V^- \cup V^+$, we add delay constraints for the newly added arcs incident with the dummy supernode: for the arcs $e = (v, s)$, we define the delay function over e by $d(e, \mathbf{p}) \stackrel{\text{def}}{=} -a_v$. For the arcs $e = (s, v)$ we put $d(e, \mathbf{p}) \stackrel{\text{def}}{=} a_v$. It is immediate that the new set of constraints[6] is equivalent to the original set of timing constraints. The resulting graph is denoted by $G' = (V' \stackrel{\text{def}}{=} V \cup \{s\}, E' \stackrel{\text{def}}{=} E \cup \tilde{E})$, where $\tilde{E}$ is the set of newly added arcs. Let $\mathbf{D}$ denote the node-arc incident matrix of G'. We assume that G' is connected (which is the typical case in practice), otherwise the problem can be separated into disjoint subproblems. Therfore, the rank of $\mathbf{D}$ is $|V'| - 1 = |V|$ and the deletion of any row of $\mathbf{D}$ results in a matrix with the same rank. Let $U \subseteq \mathbb{R}^{\tilde{E}}$

[6]Note that the arrival time of the supernode can be left variable.

denote the oriented cutset space of G' which is generated by the columns of $\mathbf{D}^T$. If we delete an arbitrary row of $\mathbf{D}$ (for example the one corresponding to the supernode s), then for the resulting matrix $\mathbf{D}_s$, $\mathbf{D}_s\mathbf{D}_s^T$ (a submatrix of the combinatorial Laplacian of G') is positive definite and $\mathbf{D}_s^T$ is a minimal generating matrix of U.

Given $\boldsymbol{\lambda} \in \mathbb{R}^E$, the task is to compute the orthogonal projection $P_{U^\perp}(\boldsymbol{\lambda})$ to the flow space. It is basic linear algebra that $P_U + P_{U^\perp}$ is the identity function of $\mathbb{R}^E$, that is

$$P_{U^\perp}(\boldsymbol{\lambda}) = \boldsymbol{\lambda} - P_U(\boldsymbol{\lambda}).$$

The orthogonal projection to U can be performed by solving

$$\mathbf{D}_s\mathbf{D}_s^T\mathbf{x} = \mathbf{D}_s\boldsymbol{\lambda}. \tag{4.3}$$

Then $\mathbf{D}_s^T\mathbf{x} = P_U(\boldsymbol{\lambda})$. Note that $\mathbf{D}$ is a sparse matrix having exactly two nonzero entries in each column, so for given $\boldsymbol{\lambda}$ and $\mathbf{x}$, $\mathbf{D}_s\boldsymbol{\lambda}$ and $\mathbf{D}_s^T x$ can be both computed in $O(|E| + |V|)$ time. $\mathbf{D}_s\mathbf{D}_s^T$ is a sparse matrix with at most $|V'| + 2|E'| \leq 3|V| + 2|E|$ nonzero entries so one iteration of the conjugated gradient method can be performed in $O(|E|+|V|)$. To obtain an exact solution of (4.3), $|V|$ iterations are needed. In pratice, however, only a small number of iterations (well under 100) are sufficient to get a very good approximative solution if one uses suitable preconditioning techniques. So the practically observable run-time of the ovarall projection scales in practice almost linearly with the size of the graph.

A slight annoyance induced by this method is that there are simple examples for which the Lagrangian multipliers are partially increased for timing feasible instances. The exact projection to $U^\perp_{\geq 0}$ guarantees that the Lagrangian multipliers never increase if the design is feasible in the current step. In the long run, the algorithm with cyclic projection converges as shown in Theorem 3.7.4, but the rate of convergence may be impaired by this phenomenon. Though it seems that the overall run-time improvement compared to an exact projection cancels this effect. An interesting idea to cure this shortcoming is to combine the subgradient method with the method of reflection-projection proposed in [Buschke and Kruk 2002] instead of the method of cyclic projections.

Of course the question arises **what** to project. The literature ([Chen, Chu and Wong 1999], [Langkau 2000] and [Sechen and Tennakoon 2002]) suggests for the similar gate-sizing problem that the arrival times are propagated by static timing analysis and the arc-slacks with respect to this propagation are added to the multipliers and used in the projection. It may be based on the

idea that the arrival time vector has to become primal feasible once the design gets feasible. However, it turns out that the propagated arrival times do not matter at all.

It is clear that

$$P_{U^\perp_{\geq 0}}(\boldsymbol{\lambda}) = P_{U^\perp_{\geq 0}}(P_{U^\perp}(\boldsymbol{\lambda})),$$

since $U^\perp$ is a linear subspace of $\mathbb{R}^E$. On the other hand, choosing a different arrival time vector means the addition of an element of U to $\boldsymbol{\lambda}$. After projecting to $U^\perp$, this contribution vanishes again. This means that one does not need to propagate at all to perform the timing optimization: ρd_e can be added to $\boldsymbol{\lambda}_e$ for each arc, where d_e is simply the delay over the arc. The projection of the Lagrangian multipliers already "propagates" the design in some sense. Note that we have transformed the boundary conditions on the arrival times at $V^+ \cup V^-$ into delays over the arcs of the newly inserted dummy node, so they affect the overall result of the projection.

Another interesting side-effect is that this method is able to optimize designs with variable clock arrival times without the necessity of performing clock-skew scheduling explicitly which in turn suggests that the clock-skew scheduling itself could be performed by a projection algorithm instead of the combinatorial ones in [Albrecht 2001], [Albrecht et al. 2002] and [Held 2001]. This is possible, but the main complication is that the delays must be projected to the space of nonnegative flows for which the run-times of best known algorithms scale superlinearly with the size of the graph (Projecting to the flow space could result in suboptimal scheduling even for timing-feasible designs). Methods based on combinatorial potential-balancing algorithms seem to run sufficiently fast in practice, so without additional benefits the use of such methods is not justified. Different analytical methods based on a constrained quadratic programming approach were proposed in [Kourtev and Friedman 1999] to improve robustness of the clock-scheduling.

4.3.5 The Placement Subproblem

In order to evaluate the dual function, the following function has to be minimized.

$$Ł_1(\mathbf{p}, \boldsymbol{\lambda}) = \sum_{e \in E} \boldsymbol{\lambda}_e d(e, \mathbf{p}) + \sum_{v \in V} l''(w(v, \mathbf{p}))$$

This is the classical placement problem with netweights. To solve this, we will make some additional assumptions about the netlength estimation functions

l, l' and l'': Let us assume that each of them can be written as a non-negative linear combination of some L_1 distances and squares of L_2 distances between pairs of placement locations of semi-adjacent gates (that is, gates adjacent to the same net):

$$l(e) = \sum_{(v,w)\in V\times V} \boldsymbol{\alpha}^e_{v,w}|\mathbf{p}_v - \mathbf{p}_w| + \boldsymbol{\beta}^e_{v,w}|\mathbf{p}_v - \mathbf{p}_w|^2.$$

The same assumptions are made about l' and l'' also. This means that Ł$_2$ becomes a function of the same form (with different $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ values):

$$Ł_2(\mathbf{p}, \boldsymbol{\lambda}) = \sum_{(v,w)\in V\times V} \boldsymbol{\alpha}_{v,w}|\mathbf{p}_v - \mathbf{p}_w| + \boldsymbol{\beta}_{v,w}|\mathbf{p}_v - \mathbf{p}_w|^2,$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ depend on $\boldsymbol{\lambda}$ linearly. So the task is to minimize a mixed non-negative linear combination of linear and quadratic distances between a set of nodes where the location of some nodes are prescribed. The justification for this model is based on the observation that the widely used Elmore delay estimation [Elmore 1948] is a quadratic function of the wire-length, but the power consumption and gate-delays depend linearly on the wire length. However, the pure linear objective is not strictly convex and the set of optimal solutions can be large. This is a disadvantage, since the stability of the placement algorithms is of importance: small changes in the netlist should have a limited effect. A mixed linear-quadratic

In the literature, the weighted rectilinear netlength minimization (without offsets) is handled under the label *multifacility location problem.*

The classical conjugated gradient method [Hestenes and Stiefel 1952] proved to be a very efficient for minimizing the quadratic netlength exactly or approximately ([Wipfler, Wiesel and Mlynski 1983], [Weismantel 1992], [Kleinhans et al. 1991], [Vygen 1996] and [Alpert, Kahng and Yao 1999]). The minimization of weighted rectilinear netlength can be written as a linear program whose dual is a minimum cost flow problem. This fact was already used in [Cabot, Francis and Stary 1970] to solve multifacility location problems, which is essentially the placement problem without disjointness constraints. Although minimum cost flow problems can be solved quite efficiently, the sheer size of the problems occuring in VLSI design (over ten million variables after transformation) motivates finding better performing specialized solutions. Another efficient special purpose algorithm is given in [Dax 1986], which is based on local refinement steps. The hard part of this algorithm is the optimality test, particularly for

the case of facilities with coinciding positions. An older naturally arising idea (Originally in [Weiszfeld 1937], but was rediscovered several times by different authors) is to iteratively approximate the linear case by quadratic functions.

4.3.6 Placement via Weiszfeld's Idea

In this section, we will discuss increasingly general versions of the mixed linear-quadratic netlength minimization problem and solve them using Weiszfeld's idea. That way, we will understand the historical background and the essence of the idea. At the end, we will arrive at the weighted mixed linear-quadratic netlength minimization problem.

In [Weiszfeld 1937] the so called *Fermat-Weber problem* was treated, which occurs frequently in Operations Research:

Given n points $p_1, \ldots, p_n \in \mathbb{P}$ in the plane and nonnegative weights $w_1, \ldots, w_n \in \mathbb{R}_{>0}$, find a point $x \in \mathbb{P}$ minimizing the following weighted sum of Euclidean distances:

$$f(\mathbf{x}) \stackrel{\text{def}}{=} \sum_{i=1}^{n} w_i \|x - p_i\|_2.$$

Algorithm of Weiszfeld

Input:

- Points $p_1, \ldots, p_n$ in the k-dimensional space.
- Weights $w_1, \ldots, w_n \in \mathbb{R}_{>0}$.

Output: A sequence $(x_n) \in \mathbb{R}^k$ of points.

① Choose an arbitrary initial point x_0 not coinciding with any of $p_1, \ldots, p_n$.
$l \leftarrow 0$

② Repeat

$$x_{l+1} \leftarrow \arg\min_{x \in \mathbb{R}^k} \sum_{i=1}^{n} w_i \frac{\|p_i - x\|_2^2}{\|p_i - x_l\|_2} \text{ and increase } l.$$

Of course, this problem is a very restricted special case of the general placement problem. However, we will see that Weiszfeld's idea can be generalized to a great extent. What we can already see is that the above approach does

not work if the iterated point x_k coincides with one of the p_i. To resolve this situation, first the points $p_1, \ldots, p_n$ are checked for optimality. If the optimal solution is not one of those points, then x_{k+1} is placed randomly in the bounding box of the $p_1, \ldots, p_n$ whenever x_k coincides with one of them. One can prove that the corrected method converges to an optimal solution with probability 1.

Note that the rectilinear case (where the weighted sum of rectilinear distances is to be minimized) is separable by coordinates, so only the one-dimensional case is to be solved. This is basically a median-computation task and can be computed very efficiently without Weiszfeld's idea.

However, Weiszfeld's idea generalizes to minimizing the linear L_2 netlength of a chip. We will consider the somewhat reduced problem where there is a bijection between the gates and the nodes. The more general problem with multiple output pins can be reduced to this case (see: [Struzyna 2004]). In this case we successively solve weighted quadratic placement problems and adjust the weights according to the distances in the current iteration. One should note that the convergence of this method may be excessively slow. Quadratic convergence was achieved in [Li 1996] and [Alpert et al. 1997b] by using the dual Newton method. It has excellent practical convergence, but the region of convergence (the set of initial solutions for which the method converges) is hard to determine. They propose using the plain Weiszfeld algorithm for the first steps. Another complication is that the system of linear equations involving the Hessian must be solved extremely accurately and therefore the overall run-time is quite high. We can also see that the method fails if the placements of adjacent vertices coincide in some iteration as it would require a division by zero. This is not as simple to solve as in the case of one single movable point. In [Alpert et al. 1997b] the so called *ϵ-regularization* was proposed: the Euclidean distance function $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$ is replaced by $\sqrt{\epsilon + \sum_{i=1}^{k}(x_i - y_i)^2}$ with some suitably chosen small epsilon. Although it introduces a slight inaccuracy, it smoothens the objective function and makes the Weiszfeld or the Newton method applicable. In the case of the Newton method, the problem is that with decreasing ϵ the condition of the Hessian gets worse. This makes it computationally expensive to solve the system of linear equations required by the Newton method.

In chip design, the rectilinear case is of great importance.

LINEAR WEIGHTED NETLENGTH MINIMIZATION

Instance:

- A gate graph $G = (V, E)$. Let V' denote an arbitary set of *preplaced* nodes such that V' intersects each connected component of G.
- Preplacement $\mathbf{p}' \in \mathbb{P}^{V'}$.
- Positive arc-weights $\mathbf{w} \in \mathbb{R}^E_{>0}$.
- Relative offsets $\mathbf{o} \in \mathbb{P}^E$.

Goal: Find a placement $\mathbf{p} \in \mathbb{P}^V$ extending $\mathbf{p}'$ and minimizing

$$d(\mathbf{p}) \stackrel{\text{def}}{=} \sum_{e \in E} \mathbf{w}_e \|\mathbf{p}_{e^-} + \mathbf{o}_e - \mathbf{p}_{e^+}\|_1.$$

We will assume that we can efficiently solve the quadratic weighted netlength minimazation problem, defined as follows:

QUADRATIC WEIGHTED NETLENGTH MINIMIZATION

Instance:

- A gate graph $G = (V, E)$. Let V' denote an arbitary set of *preplaced* nodes such that V' intersects each connected component of G.
- Preplacement $\mathbf{p}' \in \mathbb{P}^{V'}$.
- Positive arc-weights $\mathbf{w} \in \mathbb{R}^E_{>0}$.
- Relative offsets $\mathbf{o} \in \mathbb{P}^E$.

Goal: Find a placement $\mathbf{p} \in \mathbb{P}^V$ extending $\mathbf{p}'$ and minimizing

$$d(\mathbf{p}) \stackrel{\text{def}}{=} \sum_{e \in E} \mathbf{w}_e \|\mathbf{p}_{e^-} + \mathbf{o}_e - \mathbf{p}_{e^+}\|_2^2.$$

In fact, as already mentioned in the last section, the latter problem can be very efficiently solved by the method of conjugate gradients [Hestenes and Stiefel 1952] applied to a submatrix of the weighted combinatorial Laplacian (with weight

matrix diag($\mathbf{w}$)) which happens to be symmetric positive semidefinite (cf. [Wipfler, Wiesel and Mlynski 1983], [Kleinhans et al. 1991], [Weismantel 1992], [Vygen 1997] and [Alpert et al. 1997]).

The problem of linear netlength minimization can be solved using the idea of Weiszfeld: iteratively solving quadratic approximations of the original problem as noted in [Kleinhans et al. 1991] and [Struzyna 2004]. Again, a guarantee of convergence can only be given if the optimal positions of incident points never coincide and, moreover, the intermediate positions of incident points never coincide either. This condition is hard to check.

Now we turn to the more general problem of the mixed linear-quadratic weighted netlength minimization:

MIXED LINEAR-QUADRATIC WEIGHTED NETLENGTH MINIMIZATION

Instance:

- A gate graph $G = (V, E)$. Let V' denote an arbitary set of *preplaced* nodes such that V' intersects each connected component of G.
- Preplacement $\mathbf{p}' \in \mathbb{P}^{V'}$.
- Positive arc-weights $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^E_{>0}$.
- Relative offsets $\mathbf{o} \in \mathbb{P}^E$.

Goal: Find a placement $\mathbf{p} \in \mathbb{P}^V$ extending $\mathbf{p}'$ and minimizing

$$d(\mathbf{p}) \stackrel{\text{def}}{=} \sum_{e \in E} \left(\boldsymbol{\alpha}_e \|\mathbf{p}_{e^-} + \mathbf{o}_e - \mathbf{p}_{e^+}\|_2^2 + \boldsymbol{\beta}_e \|\mathbf{p}_{e^-} + \mathbf{o}_e - \mathbf{p}_{e^+}\|_1 \right).$$

The problem can be separated by the coordinates[7], so we only have to deal with the one-dimensional case, that is $\mathbf{p} \in \mathbb{R}^V$ and $\mathbf{o} \in \mathbb{R}^E$. To incorporate the Elmore delay more accurately, we should work with the square of the rectilinear distance instead of using the squared Euclidean distance. In fact, our exposition would work in that case too, but the analysis gets less elegant.

[7]We formulate the problem and solution in the two dimensional space, which is the relevant case to the application in chip design. It is clear now that the exposition works equally well without modification for higher dimensional spaces.

To the superficial viewer it may have seemed that Weiszfeld's idea consists in approximating the linear objective function by a quadratic one which attains the same value at $\mathbf{x}^k$. This is a mere coincidence. The real idea is to approximate it by a quadratic function whose *gradient* coincides with that of the objective function at $\mathbf{x}^k$. In the case of linear objective function, this does not make any difference for the algorithm, but the case of a mixed linear-quadratic objective requires a correct understanding of the idea.

To simplify our notation, we introduce the following abbreviations: for each $e \in E$ and $\mathbf{x} \in \mathbb{R}^{V \setminus V'}$:

$$r_e(\mathbf{x}) \stackrel{\text{def}}{=} |\mathbf{p}_{e^-} - \mathbf{p}_{e^+} + \mathbf{o}_e|, \text{where } \mathbf{p}_v = \begin{cases} \mathbf{p}'_v & \text{if } v \in V', \\ \mathbf{x}_v & \text{if } v \in V \setminus V'. \end{cases}$$

For $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{V \setminus V'}$, we also define

$$\psi_{\mathbf{x}}(\mathbf{y}) \stackrel{\text{def}}{=} \sum_{e \in E} \left(\frac{\boldsymbol{\alpha}_e}{2r_e(\mathbf{x})} + \boldsymbol{\beta}_e \right) r_e(\mathbf{y})^2$$

and the objective function is:

$$\phi(\mathbf{x}) \stackrel{\text{def}}{=} \sum_{e \in E} \boldsymbol{\alpha}_e r_e(\mathbf{x}) + \boldsymbol{\beta}_e r_e(\mathbf{x})^2.$$

GENERALIZED WEISZFELD ALGORITHM

Input:

- Graph $G = (V, E)$.
- $V' \subseteq V$ intersecting each connected component of G.
- Preplacement $\mathbf{p}' \in \mathbb{R}^{V'}$.
- Linear and quadratic weight vectors: $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^E_{>0}$.

Output: A sequence $(\mathbf{x}^n) \in \mathbb{R}^{V \setminus V'}$.

① Choose an arbitrary initial point $\mathbf{x}^0 \in \mathbb{R}^{V \setminus V'}$. $k \leftarrow 0$.

② Repeat

$$\mathbf{x}^{k+1} \leftarrow \arg \min_{\mathbf{x} \in \mathbb{R}^{V \setminus V'}} \psi_{\mathbf{x}^k}(\mathbf{x}) \text{ and increase } k.$$

Note that $\nabla\psi_{\mathbf{x}^k} = \nabla\phi$ at $\mathbf{x}^k$. We will prove the convegence of the above method for the case that $r_e(\mathbf{x}^k)$ never vanishes.

Theorem 4.3.3 *The generalized Weiszfeld algorithm converges to the minimum of ϕ if $r_e(\mathbf{x}^k) \neq 0$ for all $k \in \mathbb{N}$ and $e \in E$ and $r_e(\lim_{k\to\infty} \mathbf{x}^k) \neq 0$ for each $e \in E$.*

Proof: If $\mathbf{x}^{k+1} = \mathbf{x}^k$, then $\mathbf{x}^k$ is the unique minimum of the smooth, strictly convex function $\psi_{\mathbf{x}^k}$. Note that the convex function ϕ is smooth in some neighbourhood of each $\mathbf{x}$ satisfying $r_e(\mathbf{x}) \neq 0$ ($\forall e \in E$). The minimality of $\psi_{\mathbf{x}^k}(\mathbf{x}^{k+1})$ implies

$$0 = \nabla\psi_{\mathbf{x}^k}(\mathbf{x}^k) = \nabla\phi(\mathbf{x}^k),$$

which means that $\mathbf{x}^k$ is the minimum of the objective function ϕ. So if the sequence $(\mathbf{x}^k)$ gets stationary then it converges to the minimum of ϕ. Now we can assume that $\mathbf{x}^{k+1} \neq \mathbf{x}^k$. First, we prove that $\phi(\mathbf{x}^{k+1}) < \phi(\mathbf{x}^k)$. For this purpose, we define

$$\delta_e \stackrel{\text{def}}{=} r_e(\mathbf{x}^{k+1}) - r_e(\mathbf{x}^k).$$

Since $\psi_{\mathbf{x}^k}$ is a strictly convex function, $\mathbf{x}^{k+1}$ is its unique minimum. Therefore, $\mathbf{x}^k \neq \mathbf{x}^{k+1}$ implies $\psi_{\mathbf{x}^k}(\mathbf{x}^{k+1}) < \psi(\mathbf{x}^k)$. This means:

$$\sum_{e\in E}\left(\frac{\boldsymbol{\alpha}_e}{2r_e(\mathbf{x}^k)} + \boldsymbol{\beta}_e\right)\left(r_e(\mathbf{x}^k) + \delta_e\right)^2 < \sum_{e\in E}\left(\frac{\boldsymbol{\alpha}_e}{2r_e(\mathbf{x}^k)} + \boldsymbol{\beta}_e\right) r_e(\mathbf{x}^k)^2$$

$$\sum_{e\in E}\left(\frac{\boldsymbol{\alpha}_e}{2r_e(\mathbf{x}^k)} + \boldsymbol{\beta}_e\right)\left(r_e(\mathbf{x}^k)^2 + 2\delta_e r_e(\mathbf{x}^k) + \delta_e^2\right) < \sum_{e\in E}\left(\frac{\boldsymbol{\alpha}_e r_e(\mathbf{x}^k)}{2} + \boldsymbol{\beta}_e r_e(\mathbf{x}^k)^2\right)$$

$$\sum_{e\in E}\left(\boldsymbol{\alpha}_e\delta_e + \frac{\boldsymbol{\alpha}_e\delta_e^2}{2r_e(\mathbf{x}^k)} + \boldsymbol{\beta}_e(2r_e(\mathbf{x}^k)\delta_e + \delta_e^2)\right) < 0$$

$$\sum_{e\in E}\left(\boldsymbol{\alpha}_e\delta_e + \boldsymbol{\beta}_e(2r_e(\mathbf{x}^k)\delta_e + \delta_e^2)\right) < 0$$

$$\sum_{e\in E}\left(\frac{\boldsymbol{\alpha}_e}{r_e(\mathbf{x}^k) + \delta_e} + \boldsymbol{\beta}_e\right)\left(r_e(\mathbf{x}^k) + \delta_e\right)^2 < \sum_{e\in E}\left(\frac{\boldsymbol{\alpha}_e}{r_e(\mathbf{x}^k)} + \boldsymbol{\beta}_e\right) r_e(\mathbf{x}^k)^2$$

which shows

$$\phi(\mathbf{x}^{k+1}) < \phi(\mathbf{x}^k).$$

So the statement follows from Theorem 3.1.1 of Zangwill, since ϕ is a function of descent and each $\mathbf{x}^k$ is contained in the offset adjusted bounding box of the preplaced nodes. Using the smoothness of ϕ at $\mathbf{x}^* \stackrel{\text{def}}{=} \lim_{k\to\infty} \mathbf{x}^k$, the optimality of the solution follows. □

Of course the above theorem provides a good mathematical foundation for using the fixpoint algorithm. However, it ignores the singular points completely and poses a hard-to-check condition.

4.3.7 The Overall Algorithm

Let us summarize the timing driven placement algorithm based on the exact projection:

TIMING DRIVEN PLACEMENT

Input: An instance of the *placement problem under timing restrictions* with mixed quadratic-linear objective, capacitance and arc-delay functions. A sequence $\rho_k \in \mathbb{R}_{>0}$ with $\rho_k \to 0$ and $\sum \rho_k \to \infty$.

Output: A sequence $(\mathbf{p}^k) \in \mathbb{P}^V$ extending the given preplacement.

① Augment the timing graph by the dummy supernode as in Section 4.3.4.

② $k \leftarrow 0$, $\boldsymbol{\lambda}^0 \leftarrow 0$.

③ Compute a placement $\mathbf{p}^k$ minimizing $Ł_1(\mathbf{p}, \boldsymbol{\lambda}^k)$ by a method presented in the preceeding section.

④ $\boldsymbol{\lambda}^{k+1} \leftarrow P_{U^\perp_{\geq 0}}(\boldsymbol{\lambda}_k + \boldsymbol{\rho}_k d(\mathbf{p}^k))$

⑤ Increase k and **go to** ③ **until** the duality gap exceeds the prescribed threshold.

The convergence (of a subsequence) and optimality of this method are consequences of that of the constrained subgradient method applied to the dual function D. The justification of the projection step comes from the fact that each optimal solution of the dual function $D(\boldsymbol{\lambda})$ must satisfy the delay inequalities. In fact, $D(\boldsymbol{\lambda}) = -\infty$, if $\boldsymbol{\lambda}$ is not a flow, therefore a restriction of the dual function is mandatory. In fact we maximize $D_1(\boldsymbol{\lambda}) = \inf Ł_1(\mathbf{p}, \boldsymbol{\lambda})$ over the set of nonnegative flows. However, $D_1(\boldsymbol{\lambda}) = D(\boldsymbol{\lambda})$ if $\boldsymbol{\lambda}$ forms a flow on the augmented timing graph. Theorem 3.3.2 still guarantees that $d(\mathbf{p}^k)$ is a subgradient of $D_1(\boldsymbol{\lambda}^k)$, justifying the above algorithm.

One can also opt to use the new alternating projection framework. In order to do so, only (the time consuming) projection $P_{U^\perp_{\geq 0}}$ in step ④ is to be replaced by $P_{\mathbb{R}_{\geq 0}} \circ P_{U^\perp}$. Theorem 3.5 implies that the method still converges to an

optimum of D_1 over the set of nonnegative flows, and therefore producing a sequence having a subsequence converging to the global optimum.

4.4 The Gate- and Wire-Sizing Problem

Besides placement, the individual sizes of the gates are of great importance to the timing behaviour and power consumption of a chip. There may be (and typically is) a multitude of possible physical realizations ("cells") for a given logical function (for example a two-way AND gate). Cells may differ in size, power-consumption and other physical or manufacturing parameters. The timing paramaters of a gate (its delays, the shape of the output signal, pin capacitances) depend essentially on its realization. The collection of all possible realizations on a chip is called "cell library". The cell library itself is relatively stable and used for the design of many different chips. However not all designs are allowed to use all available cells of a library. For example chips used in mobile equipments may omit cells with high leakage power in order to preserve battery life. Manufacturing costs can also be reduced by using a restricted library.

The width and other parameters of wires play a role in the timing estimation. An appropriate sizing the wires can improve the delay of nets, but the effect that can be achieved is usually much more limited than that of the gate-sizing since wire delays are typically neglectible for short interconnections, and the choice of thinnest wire width is optimal for most nets. However, with shrinking feature sizes, wire sizing and repeater insertion gains on importance.

4.4.1 Gate Delay Models

For a single logic function (like a 2-way NAND), available cells are typically divided into classes that differ only by size while their other parameters are fixed or scale well with their size. For example, balanced inverters may have similar delay characteristics for rising and falling signals whereas "standard" inverters have lower delays for the rising signals.

There are successful timing optimization tools (cf. [Schietke 1999]), that are agnostic about cell classes. Such tools rely exclusively on combinatorial methods: the effects of several different realizations are estimated and the one optimizing a fixed objective function is chosen. Sophisticated versions of this procedure generate realizations for a large set of gates simultaneously and a

subset of these realizations optimizing the objective function is applied simultanously. The advantage of this approach is that an analysis of the cell library is not required, no assumptions are necessary about the scalability of the cells. On the other hand side, the quality of the solution is impossible to be estimated and the runtime may increase with the number of cells in the library considerably.

Analytical methods promise cure for the above shortcomings of the combinatorial method. Such algorithms were proposed for example in [Fishburn and Dunlop 1985], [Chen, Chang and Wong 1996], [Cong and He, 1996], [Chu and Wong 1999] and [Chen, Chu and Wong 1999]. Almost all proposals make the assumption that the realization of each gate is chosen from the same class in which the cell characteristics scale perfectly with size. If the signal shapes are neglected and the library is assumed to be continuous (that is, if the gate sizes are chosen from a real interval) then the problem of finding a power optimal design meeting all timing constraints can be transformed to a constrained convex optimization problem (already recognized in [Fishburn and Dunlop 1985]). In order to see this, the delay model must be defined mathematically.

4.4.2 The Linear Delay Model

A common model for gate-delays is to assume that the delays of the gates depend linearly on the driven downstream capacitance and reverse proportionally on the size of the gate itself. This is justified by theoretical models describing the behaviour of transistors [Fishburn and Dunlop 1985]. That is $d_e = l_e + \frac{p_e c}{s_g}$ where d_e is the delay over propagation segment e inside gate g, l_e is the constant term of the delay (depending only on propagation segment e), c is the load (accumulated capacitance in the net) driven by the head of e and s_g is the size of g. The capacitances of the input pins of gates are assumed to scale linearly with the size of s_g of gate g. They contribute to the load of the predecessor (driving) stage.

For estimating the wire delays, the Elmore delay is sufficiently exact in prerouting optimization. Let us consider a piece of wire whose width w may vary between $\underline{\mathbf{b}}$ and $\overline{\mathbf{b}}$: the Elmore delay d over the wire is described by

$$d \stackrel{\text{def}}{=} \frac{r}{w}(\frac{cw}{2} + C),$$

where r and c are positive constants depending on the wire segment. (r and c depend on the length of the wire linearly, wires in different layers may have

different coefficients. Even in the same plane, isolated wires are assumed to have smaller capacitance, since primitive crosstalk estimations are typically included in the constants of the above formula.) C is the downstream capacitance which is the sum of all pin and wire capacitances driven by the piece of wire. Let us summarize this informally given model mathematically:

GATE- AND WIRE-SIZING INSTANCE

Instance:

- A directed graph $G = (V, E)$ called *timing graph*.
- A set of gate- and wire-objects (*features*) $F = F_g \cup F_w$.
- A feature-assignment $\varphi : V \longrightarrow F$. Arc $e \in E$ is called *wiring arc*, if $\varphi(e^+) \in F_w$, otherwise *belonging to gate* $\varphi(e^+)$ The set E_w of all wiring arcs is assumed to be a forest each connected component of which having a unique source node v with $\varphi(v) \in F_g$. φ is assumed to be a bijection on E_w, that is each wiring arc has a unique wire object.
- Let $\mathbf{p}, \mathbf{l} \in \mathbb{R}^E_{\geq 0}$ be the coefficients of the linear delay estimation function for each arc.
- $\underline{\mathbf{b}}, \overline{\mathbf{b}} \in \mathbb{R}^F_{>0}$ a minimum and a maximum size for each feature.
- $\mathbf{r} \in \mathbb{R}^E_{>0}$ capacitance ratios for the pin and wire capacitances.

Let us assume that we have an instance of the gate- and wire-sizing problem using the above notations. A size assignment vector $\mathbf{x} \in \mathbb{R}^F_{>0}$ satisfying $\underline{\mathbf{b}}_g \leq \mathbf{x}_g \leq \overline{\mathbf{b}}_g$ for each $g \in F$. To introduce the notion of *load* (or *downstream capacitance*) seen by node v, the set of features contributing to its load must be specified, which can be done by defining a partial order $\prec$ on the nodes of the timing graph: we say that node w is *seen by* v or simply $v \prec w$ if there is a directed path from v to w using exclusively wiring arcs.

The *downstream capacitance* $C(\mathbf{x})_v$ at node v with respect to $\mathbf{x}$ is given by:

$$C(\mathbf{x})_v \stackrel{\text{def}}{=} \sum_{\substack{e \in F \\ v \prec e^-}} \mathbf{r}_e \mathbf{x}_{\varphi(e^+)}.$$

That is the summation goes over all arcs whose tail is seen by node v. Delay

$d(e)$ over propagation arc e is computed by:

$$d(e) \stackrel{\text{def}}{=} \mathbf{l}_e + \frac{\mathbf{p}_e C(\mathbf{s})_{e^+}}{\mathbf{x}_{\varphi(e^+)}}.$$

4.4.3 Considering Signal Shapes

A serious problem of the above formulation is that the delay model is extremely simplified and therefore inaccurate, restricting its practical usability.

One of the main reasons of the inaccuracy is that the shape (cf. Section 4.2.4) of the signals is not taken into account by the model. A complete cure for this problem is not simple. The straightforward approaches for solving it have one of the following shortcomings:

- Makes the problem nonconvex.
- Introduces exponentially many variables.
- Uses overly pessimistic assumptions.

Now, we will consider a tractable but still a bit inaccurate model that does not exhibit any of the above drawbacks: the signal shape of a stage is assumed to influence only the delays of the immediate successor stages, but this effect is not propagated further, so the change of the signal shape does not influence the shape of the signals leaving the successors gates, only their delay. This assumption is justified by the observation that overall effect of signal shapes to the delay concentrates on the delay of the next stage: it decreases exponentially in the subsequent stages. The signal shapes are assumed to be estimated by a linear model (cf. Section 4.2.6), that is the shape of each signal is parametrized by a single real number: the slope of the linear estimation. The following additional assumptions are made:

(1) For fixed input signal shape, the slope of the output signal shape over a non-wiring arc depends linearly on the downstream capacitance seen by the head.

(2) For fixed input signal shape, the slope of the output signal shape over a non-wiring arc depends reverse proportionally on the size of the gate of the head.

(3) For fixed input signal shape, The slope of the output signal shape at the head of the wiring arc depends linearly on the delay over the arc.

(4) The delay over a wiring arc does not depend on the signal shape.

(5) The delay over a non-wiring arc depends linearly on the slope of the input signal shape at the tail.

(6) The coefficients of the linear functions in the above assumptions are nonnegative.

Formally, we can introduce variables for signal shapes $\mathbf{s}_v$ for each node, parameters $\boldsymbol{\mu}, \boldsymbol{\mu}' \in \mathbb{R}_{\geq 0}^{V_g}$, and $\boldsymbol{\nu} \in \mathbb{R}_{\geq 0}^{E_w}$ satisfying:

$$\mathbf{s}_v \stackrel{\text{def}}{=} \boldsymbol{\mu}_v \frac{C(\mathbf{x})_v}{\mathbf{x}_{\varphi(v)}} + \boldsymbol{\mu}'_v \qquad \forall v \in V_g \tag{4.4}$$

$$\mathbf{s}_{e^+} \stackrel{\text{def}}{=} \boldsymbol{\nu}_e d(e) + \mathbf{s}_{e^-} \qquad \forall e \in E_w \tag{4.5}$$

$$d(e) \stackrel{\text{def}}{=} \mathbf{l}_e + \frac{\mathbf{p}_e C(e^+)}{\mathbf{x}_{\varphi(e^+)}} \qquad \forall e \in E_w \tag{4.6}$$

$$d(e) \stackrel{\text{def}}{=} \mathbf{l}_e + \frac{\mathbf{p}_e C(e^+)}{\mathbf{x}_{\varphi(e^+)}} + \boldsymbol{\nu}_e \mathbf{s}_e \qquad \forall e \in E_g \tag{4.7}$$

$$\tag{4.8}$$

Fortunately, we can eliminate the newly introduced variables and parameters by introducing a new set $\mathbf{p}' \in \mathbb{R}_{\geq 0}^{E \times F \times F}$ of parameters: Our assumptions on the timing graph imply that for node $v \in V_w$ there is a node $\hat{v} \in V_g$ which is the unique source of the component of $G[E_w]$ containing v. (The source of the net containing v). For a node $v \in V$, denote $P(v)$ the arcs in the unique directed path connecting $\hat{v}$ and v. The delay over arc $e \in E_g$ with tail $v \stackrel{\text{def}}{=} e^-$ can be written as:

$$d(e) = \mathbf{l}_e + \frac{\mathbf{p}_e C(e^+)}{\mathbf{x}_{\varphi(e^+)}} + \boldsymbol{\nu}_e \left(\boldsymbol{\mu}_{\hat{v}} \frac{C_{\hat{v}}(\mathbf{x})}{\mathbf{x}_{\varphi(v)}} + \boldsymbol{\mu}'_{\hat{v}} + \sum_{\tilde{e} \in P(v)} \boldsymbol{\nu}_{\tilde{e}} \left(\mathbf{l}_{\tilde{e}} + \frac{\mathbf{p}_e C(\tilde{e}^+)}{\mathbf{x}_{\varphi(\tilde{e}^+)}} \right) \right)$$

So, the delay formula can be written uniformly for each arc:

$$d(e) = \mathbf{l}'_e + \sum_{f \in F} \sum_{\substack{f' \in F \\ f \neq f'}} \mathbf{p}'_{e,f,f'} \frac{\mathbf{x}_{f'}}{\mathbf{x}_f},$$

where $\mathbf{l}' \in \mathbb{R}^E_{\geq 0}$ and $\mathbf{p}' \in E \times F \times F$ can be extracted from the above formulas.

The goal of the gate- and wire-sizing problem is to determine a feasible power-minimal sizing of the gates such that all timing constraints are met. That is, the arrival times eqauls to the prescribed ones for the case of nodes without entering or leaving arcs and for each arc e of the timing graph holds

$$\mathbf{a}_{e^-} + d(e) \leq \mathbf{a}_{e^+},$$

that is the signals are delayed at each arc according to the supplied timing rules. The power-consumption of a gate or wire is assumed to be proportional to its size.

4.4.4 Problem Formulation

In this section an exact mathematic formulation of the generalized gate- and wire-sizing problem is given. We start with an instance of the form given in Section 4.4.2 with additional parameters $\boldsymbol{\nu}, \boldsymbol{\nu}', \boldsymbol{\mu}$ and $\boldsymbol{\mu}'$ to make it fit to handle the effect of changes in signal shapes. Then we can replace the parameter $\mathbf{p}, \mathbf{r}, \boldsymbol{\nu}, \boldsymbol{\nu}', \boldsymbol{\mu}$ and $\boldsymbol{\mu}'$ by the new uniform parameter set $\mathbf{p}'$ and $\mathbf{l}$ by $\mathbf{l}'$. Along the way, we drop several assumptions on the topology of the timing graph and the distinction between wires and gates. It turns out that the generalized problem can be still efficiently optimized.

Our new weakened assumptions can be subsumed in the following problem definition:

GENERALIZED GATE- AND WIRE-SIZING

Instance:

- A directed graph $G(V, E)$ called *timing graph*.
- A set of gate- and wire-objects (*features*) with mapping $\varphi : V \longrightarrow F$.
- $V_0 \subseteq V$ and prescribed arrival times $\mathbf{a} \in \mathbb{R}^{V_0}$.
- Lower and upper bounds $\underline{\mathbf{b}}, \overline{\mathbf{b}} \in \mathbb{R}^F_{>0}$ on the sizes of the features in F.
- Delay coefficients $\mathbf{l} \in \mathbb{R}^E$ and $\mathbf{p} \in \mathbb{R}^{E\times F\times F}_{\geq 0}$.

Goal: Determine $\mathbf{x} \in \mathbb{R}^F_{>0}$ and extend $\mathbf{a}$ to V minimizing the power consumption

$$P(\mathbf{x}) \stackrel{\text{def}}{=} \sum_{f\in F} \mathbf{x}_f.$$

subject to:

$$\forall e \in E : \mathbf{a}_u + d_e(\mathbf{x}) \leq \mathbf{a}_v \tag{4.9}$$

$$\forall f \in F : \underline{\mathbf{b}} \leq \mathbf{x}_f \leq \overline{\mathbf{b}}, \tag{4.10}$$

where

$$d_e(\mathbf{x}) \stackrel{\text{def}}{=} \mathbf{l}_e + \sum_{f\in F} \sum_{\substack{f'\in F \\ f\neq f'}} \mathbf{p}'_{e,f,f'} \frac{\mathbf{x}_f}{\mathbf{x}_{f'}}$$

4.4.5 Convexity

It is clear that the gate- and wire-sizing problem as formulated above is a posynomial program, so we can (following [Fishburn and Dunlop 1985]) transform it into a constrained convex program by a logarithmic tranformation of the size variables: Introduce variables $\boldsymbol{\xi}_f$ for each $f \in F$ so that $\mathbf{x}_f = \exp(\boldsymbol{\xi}_f)$. The new (convex) timing constraints are:

$$\mathbf{a}_{e^-} + \mathbf{l}_e + \sum_{f\in F} \sum_{f'\in F} \mathbf{p}_{e,f,f'} \exp(\boldsymbol{\xi}_f - \boldsymbol{\xi}_{f'}) \leq \mathbf{a}_{e^+},$$

for each edge $e \in E$. The objective function becomes $\sum_{f\in F} \exp(\boldsymbol{\xi}_f)$ which is a strictly convex function of $\boldsymbol{\xi}$.

4.4.6 Duality

We denote by $V' \subseteq V \stackrel{\text{def}}{=} V \setminus V_0$ the set of nodes without prescribed (required) arrival times. Following [Chen, Chu and Wong 1999], we are going to use the constrained subgradient method (cf. Theorem 3.3.3 and Theorem 3.7.4) to optimize the dual objective function. We only dualize the timing constraints and do not touch the upper and lower bounds on the feature sizes.

$$Ł(\mathbf{a}, \boldsymbol{\xi}, \boldsymbol{\lambda}) = \sum_{f \in F} \exp(\boldsymbol{\xi}_f) + \sum_{e \in E} \boldsymbol{\lambda}_e (\mathbf{a}_{e^-} + d_e(\boldsymbol{\xi}) - \mathbf{a}_{f^+})$$

The Lagrangian function is minimized over

$$X = \left\{ (\mathbf{a}, \boldsymbol{\xi}) \in \mathbb{R}^{V'} \times \mathbb{R}^F \mid \boldsymbol{\xi}_f \in [\log \underline{\mathbf{b}}_f, \log \overline{\mathbf{b}}_f] \ \forall f \in F \right\}.$$

The dual objective function D is given by

$$D(\boldsymbol{\lambda}) \stackrel{\text{def}}{=} \inf_{(\mathbf{a}, \boldsymbol{\xi}) \in X} Ł(\mathbf{a}, \mathbf{p}, \boldsymbol{\lambda}).$$

Like in the case of the timing driven placement, the Lagrangian function can be separated:

$$Ł(\mathbf{a}, \boldsymbol{\xi}, \boldsymbol{\lambda}) = Ł_1(\boldsymbol{\xi}, \boldsymbol{\lambda}) + Ł_2(\mathbf{a}, \boldsymbol{\lambda}),$$

where

$$Ł_1(\boldsymbol{\xi}, \boldsymbol{\lambda}) = \sum_{f \in F} \exp(\boldsymbol{\xi}_f) + \sum_{e \in E} \boldsymbol{\lambda}_e d_e(\boldsymbol{\xi})$$

and

$$Ł_2(\mathbf{a}, \boldsymbol{\lambda}) = \sum_{e \in E} \boldsymbol{\lambda}_f (\mathbf{a}_{e^-} - \mathbf{a}_{e^+}).$$

The intersection (or projection) $X' \stackrel{\text{def}}{=} X \cap \mathbb{R}^F$ is a bounded region (product of finite intervalls). Therefore the following minimum is finite:

$$D_1(\boldsymbol{\lambda}) = \inf_{\boldsymbol{\xi} \in X} \sum_{v \in V} \exp(\boldsymbol{\xi}_{\varphi(v)}) + \sum_{e \in E} \boldsymbol{\lambda}_e d_e(\boldsymbol{\xi})$$

However,

$$D_2(\boldsymbol{\lambda}) = \inf_{\mathbf{a} \in \mathbb{R}^V} \sum_{e \in E} \lambda_e (\mathbf{a}_{e^-} - \mathbf{a}_{e^+}) = -\infty$$

unless the Lagrangian multipliers satisfy the flow equalities on the nodes V' of timing graph G in which case $D_2(\boldsymbol{\lambda})$ is constant. So again, we can constrain the subgradient method to the flow space, which requires the projection of the Lagrangian multipliers presented in Section 4.3.4.

The only open part of the algorithm is the minimization of $Ł_1(\boldsymbol{\xi})$.

4.4.7 Local Refinement

To minimize Ł$_1$, the method of local refinement was proposed in [Chen, Chu and Wong 1999]. (Also called *method of cyclic relaxation* in [Minoux 1986] Section 4.4.1) This method proved to be very efficient, since in a typical gate- and wire-sizing situation the size variables $\mathbf{x}_f$ interact weakly among themselves. The basic idea of the algorithm is that in each step the size of one gate is optimized while all others are fixed. After all gates are optimized, the same procedure is iterated over and over again until a sufficiently good approximation is achieved. The optimal size of a gate while fixing all others can be easily and efficiently computed.

LOCAL REFINEMENT

Input: An instance of the gate sizing problem, Lagrange multipliers $\boldsymbol{\lambda} \in \mathbb{R}^E_{\geq 0}$ on the edges of the gate graph. Feasible initial gate size assignment $\boldsymbol{\xi}^1$.

Output: A sequence $(\boldsymbol{\xi}^n : \mathbb{R}^F_{\geq 0}$ of gate size vectors converging to a minimum of Ł$_1(\boldsymbol{\xi}, \boldsymbol{\lambda})$.

① Let $f_1, \ldots, f_m$ be an arbitrary ordering of the featarues and $n = 1$.

② Let $\boldsymbol{\xi}^{n+1,0}(x) = \boldsymbol{\xi}^n$.

③ Let $\boldsymbol{\xi}^{n+1,i}$ be the size assignment that mimimizes Ł$_1(\boldsymbol{\xi}^{n+1,i}, \boldsymbol{\lambda})$ among all size assignments that satisfy

$$\boldsymbol{\xi}^{n+1,i}_{f_j} = \boldsymbol{\xi}^{n+1,i-1}_{f_j} \; \forall j \neq i.$$

④ If the last feature is reached ($i = m$) then set $\boldsymbol{\xi}^{n+1} = \boldsymbol{\xi}^{n+1,i}$. Otherwise increase i and **go to** ③.

⑤ Increase n and **go to** ②.

From Corollary 3.4.2 of Luo and Tse follows that the local refinement generates a sequence converging at least linearly to the optimal solution. Several authors reproved special cases of it: for graphs with tree topology the linear convergence [Chu and Wong 1999] of the local refinement was proved and the bounds for the error were given. This result was further generalized in [Langkau 2000]. Although these results are mainly redundant from a theoretical point of view,

they still yield valuable error bounds that are hard to derive from the general theory and can be used for stopping criterion in practice.

Now we will present a simpler proof of linear convergence of the local refinement for a greatly generalized version of the problem. It contains the case of gate and wire-sizing problem for arbitrary graphs with signal shape considerations without any further restrictions. The proof presented here is shorter and clearer than the ones given for the special situations in [Chu and Wong 1999] and yields similar error bounds.

4.4.8 Generalized Local Refinement

Now, the following class of problems is considered:

PROGRAM **P**

Instance: minimize $f(\mathbf{x}) \stackrel{\text{def}}{=} \sum_{i=1}^{n}\sum_{j=1}^{n} \mathbf{A}_{i,j}\frac{\mathbf{x}_i}{\mathbf{x}_j} + \sum_{i=1}^{n}\left(\mathbf{w}_i\mathbf{x}_i + \mathbf{q}_i\mathbf{x}_i^{-1}\right)$
subject to: $\underline{\mathbf{b}} \leq \mathbf{x} \leq \overline{\mathbf{b}}$,
where $\mathbf{A}_{i,j} \in \mathbb{R}^{n\times n}_{\geq 0}$, $\underline{\mathbf{b}}, \overline{\mathbf{b}}, \mathbf{w}_i \in \mathbb{R}^n_{>0}$ and $\mathbf{q} \in \mathbb{R}^n_{\geq 0}$ with $\underline{\mathbf{b}} < \overline{\mathbf{b}}$.

Obviously, this program generalizes the problem of minimizing Ł$_2$ from the previous section.

The logarithmic variable transformation $\mathbf{x} \mapsto \boldsymbol{\xi} \stackrel{\text{def}}{=} (\log(\mathbf{x}_i))_{i\in\{1,\ldots n\}}$ transforms the objective function to a strictly convex function $\tilde{f}$ of $\boldsymbol{\xi}$, so it has a unique minimum, when the size bounds are ignored.

In order to study the convergence of the constrained problem, we will need the concept of duality again and introduce Lagrangian multipliers $\underline{\boldsymbol{\lambda}}$ and $\overline{\boldsymbol{\lambda}}$ for the lower and upper bound constraints:

$$\tilde{\text{Ł}}(\xi, \underline{\boldsymbol{\lambda}}, \overline{\boldsymbol{\lambda}}) \stackrel{\text{def}}{=} \tilde{f}(\xi) + \underline{\boldsymbol{\lambda}}^T(\underline{\mathbf{b}} - \exp(\xi)) + \overline{\boldsymbol{\lambda}}^T(\exp(\xi) - \overline{\mathbf{b}}).$$

The dual function is

$$\tilde{D}(\underline{\boldsymbol{\lambda}}, \overline{\boldsymbol{\lambda}}) \stackrel{\text{def}}{=} \inf_{\xi\in\mathbb{R}^n} \tilde{\text{Ł}}(\xi, \underline{\boldsymbol{\lambda}}, \overline{\boldsymbol{\lambda}}).$$

The conditions of Theorem 3.2.2 can be easily checked for this program. Its consequence is that the duality gap between the primal and dual optimum

is zero. This translates back to the original program **P** before the variable transformation:

$$Ł(\xi, \underline{\boldsymbol{\lambda}}, \overline{\boldsymbol{\lambda}}) \stackrel{\text{def}}{=} \tilde{f}(\mathbf{x}) + \underline{\boldsymbol{\lambda}}^T(\underline{\mathbf{b}} - \mathbf{x}) + \overline{\boldsymbol{\lambda}}^T(\mathbf{x} - \overline{\mathbf{b}}),$$

$$D(\underline{\boldsymbol{\lambda}}, \overline{\boldsymbol{\lambda}}) \stackrel{\text{def}}{=} \inf\{\tilde{Ł}(\mathbf{x}, \underline{\boldsymbol{\lambda}}, \overline{\boldsymbol{\lambda}}) \mid \mathbf{x} \in \mathbb{R}^C_{\geq 0}\},$$

which has no duality gap either. The condition of Theorems 3.2.4 and 3.2.6 are fulfilled too, so one can state:

Lemma 4.4.1 *Vector* $\mathbf{x}$ *is an optimal solution of Program* **P** *if and only if the following two implications hold:*

$$\frac{\partial}{\partial \mathbf{x}_i} f(\mathbf{x}) < 0 \Longrightarrow \mathbf{x}_i = \underline{\mathbf{b}}_i$$

$$\frac{\partial}{\partial \mathbf{x}_i} f(\mathbf{x}) > 0 \Longrightarrow \mathbf{x}_i = \overline{\mathbf{b}}_i$$

□

The partial differentials are easily computed:

$$\frac{\partial}{\partial \mathbf{x}_i} f(\mathbf{x}) = \mathbf{w}_i + B_i(\mathbf{x}) - (\mathbf{q}_i + C_i(\mathbf{x}))/\mathbf{x}_i^2, \qquad \text{where} \tag{4.11}$$

$$B_i(\mathbf{x}) \stackrel{\text{def}}{=} \sum_{\substack{j \in \{1 \ldots n\} \\ j \neq i}} \mathbf{A}_{i,j} \mathbf{x}_j^{-1} \quad \text{and} \quad C_i(\mathbf{x}) \stackrel{\text{def}}{=} \sum_{\substack{j \in \{1 \ldots n\} \\ i \neq j}} \mathbf{A}_{i,j} \mathbf{x}_j \tag{4.12}$$

Note that B_i and C_i are free of variable x_i. Let T_i defined by:

$$T_i(x) \stackrel{\text{def}}{=} \sqrt{\frac{\mathbf{q}_i + C_i(\mathbf{x})}{\mathbf{w}_i + B_i(\mathbf{x})}} \tag{4.13}$$

Obviously:

$$\frac{\partial}{\partial \mathbf{x}_i} f(\mathbf{x}) = 0 \Longleftrightarrow \mathbf{x}_i = T_i(\mathbf{x}). \tag{4.14}$$

Let furthermore $\overline{T}$ defined by:

$$\overline{T}_i(x) \stackrel{\text{def}}{=} \max\left\{\min\{T_i(\mathbf{x}), \overline{\mathbf{b}}_i\}, \underline{\mathbf{b}}_i\right\}. \tag{4.15}$$

Now, Lemma 4.4.1 can be reformulated as:

Lemma 4.4.2 *Vector* $\mathbf{x}$ *is an optimal solution of Program* **P** *if and only if* $\mathbf{x}$ *is a fixpoint of* $\overline{T}$*, that is* $\overline{T}(\mathbf{x}) = \mathbf{x}$*.* □

This suggest the following algorithm to solve Program **P**:

GENERALIZED LOCAL REFINEMENT

Input: An instance of the program **P**. Feasible initial solution $\mathbf{x}^0$ (For example $\mathbf{x}^0 = \underline{\mathbf{b}}$).

Output: A sequence $(\mathbf{x}^i \in \mathbb{R}^n_{>0})_{i=0}^{\infty}$ of feasible gate size vectors converging to a minimum of f.

① **for each** $i \geq 0$: $\mathbf{x}^{i+1} \leftarrow \overline{T}(\mathbf{x}^i)$

Now we will show the linear convergence of the above algorithm.

To study convergence rate, it is convenient to define a new distance function $\delta : \mathbb{R}_{>0} \times \mathbb{R}_{>0} \longrightarrow \mathbb{R}_{\geq 0}$ for positive real numbers by letting $\delta(x, y)$ the smallest nonnegative ϵ for which the following inequalities hold:

$$\frac{1}{1+\epsilon} \leq \frac{x}{y} \leq 1 + \epsilon. \tag{4.16}$$

Note that $\delta(x, y) = \max(\frac{x}{y}, \frac{y}{x}) - 1 = \frac{\max(x,y)}{\min(x,y)} - 1$, which also implies:

$$|x - y| = \min(x, y) \left| \frac{\max(x, y)}{\min(x, y)} - 1 \right| = \min(x, y)\delta(x, y). \tag{4.17}$$

The following inequalities are straightforward for every $x, y, a \in \mathbb{R}_{>0}$:

$$\delta(\min(x, a), \min(y, a)) \leq \delta(x, y) \quad \text{and} \quad \delta(\max(x, a), \max(y, a)) \leq \delta(x, y). \tag{4.18}$$

Lemma 4.4.3 *Let x, x', β, ϵ and a be positive real numbers. Assume that $\delta(x, x') \leq \epsilon$ and $\max(\frac{1}{1+\frac{a}{x}}, \frac{1}{1+\frac{a}{x'}}) \leq \beta < 1$. Then: $\delta(x' + a, x + a) \leq \beta\epsilon$.*

Proof: We have $x \leq \beta(x + a)$ therefore:

$$x' + a \leq (1 + \epsilon)x + a \leq \epsilon\beta(x + a) + x + a = (1 + \beta\epsilon)(x + a)$$

The other inequality of 4.16 follows since the role of x and x' is symmetric. □

Let ρ' and ρ be defined by:

$$\rho' \stackrel{\text{def}}{=} \max_{i \in \{1,\ldots,n\}} \left(1 + \frac{\mathbf{w}_i}{B_i(\underline{\mathbf{b}})}\right)^{-1} \qquad \text{and} \qquad \rho \stackrel{\text{def}}{=} \max\left\{\sqrt{\rho'}, \frac{1+\rho'}{2}\right\} \tag{4.19}$$

Note that both ρ and ρ' are strictly less than 1.

Lemma 4.4.4 *For every positive ϵ holds $\sqrt{(1+\epsilon)(1+\rho'\epsilon)} \leq 1+\rho\epsilon$.*

Proof:

$$\sqrt{(1+\epsilon)(1+\rho'\epsilon)} = \sqrt{1+(\epsilon+\rho'\epsilon)+\rho'\epsilon^2} \leq \sqrt{1+2\rho\epsilon+\rho^2\epsilon^2} = 1+\rho\epsilon \quad \square$$

Lemma 4.4.5 *Let $\mathbf{x}$ and $\mathbf{x}' \in \mathbb{R}^n_{\geq 0}$ two feasible gate size vectors. Let $\epsilon \stackrel{\text{def}}{=} \max_{i\in\{1,\ldots,n\}} \delta(\mathbf{x}_i, \mathbf{x}'_i)$, then for all $g \in C$ holds: $\delta(\overline{T}(\mathbf{x}')_i, \overline{T}(\mathbf{x})_i) \leq \rho\epsilon$.*

Proof: It is obvious by the choice of ρ that the assumptions of Lemma 4.4.3 are fulfilled for

$$x = B_i(\mathbf{x}),\ x' = B_i(\mathbf{x}'),\ a = \mathbf{w}_i,\ \beta = \rho'$$

So we get the inequality $\delta(B_i(\mathbf{x}) + \mathbf{w}_i, B_i(\mathbf{x}') + \mathbf{w}_i) \leq \rho'\epsilon$ which implies by definition of δ and Lemma 4.4.4:

$$\begin{aligned} T_i(\mathbf{x}') &= \sqrt{\frac{\mathbf{q}_i + C_i(\mathbf{x}')}{B_i(\mathbf{x}') + \mathbf{w}_i}} \leq \sqrt{\frac{(1+\epsilon)\,(\mathbf{q}_i + C_i(\mathbf{x}))}{\frac{1}{1+\rho\epsilon}(B_i(\mathbf{x}') + \mathbf{w}_i)}} \\ &= \sqrt{(1+\epsilon)(1+\rho'\epsilon)}T_i(\mathbf{x}) \leq (1+\rho\epsilon)T_i(\mathbf{x}) \end{aligned}$$

and similarly $\frac{1}{1+\rho\epsilon}T_i(\mathbf{x}') \leq T_i(\mathbf{x})$ which means $\delta(T_i(\mathbf{x}), T_i(\mathbf{x}')) \leq \rho\epsilon$. The statement for the bounded values $\overline{T}$ is now an easy consequence of inequalities 4.18. $\square$

We define

$$\Delta \stackrel{\text{def}}{=} \max_{i\in\{1,\ldots,n\}} \frac{\overline{\mathbf{b}}_i - \underline{\mathbf{b}}_i}{\underline{\mathbf{b}}_i} \tag{4.20}$$

Theorem 4.4.6 *Let $(\mathbf{x}^j)_{j=1}^{\infty}$ be the sequence of size vectors generated by the local refinement method. The sequence $(\mathbf{x}^j)$ converges to a minimum $\tilde{\mathbf{x}}$ of Problem $\mathbf{P}$. The error can be estimated by:*

$$\left|\frac{\mathbf{x}_i^j - \tilde{\mathbf{x}}_i}{\tilde{\mathbf{x}}_i}\right| \leq \frac{(\Delta+1)\Delta\rho^j}{1-\rho} \tag{4.21}$$

First we show:

$$\max_{i\in\{1,\ldots,n\}} \delta(\mathbf{x}_i^j, \mathbf{x}_i^{j+1}) \leq \Delta\rho^j. \tag{4.22}$$

It is clear for $0 = 1$ by the definition of Δ and because $\mathbf{x}_i$ satisfies $\underline{\mathbf{b}}_i \leq \mathbf{x} \leq \overline{\mathbf{b}}_i$ for each $i \in \mathbb{N}$ and $i \in \{1,\ldots,n\}$. For general $i \in \mathbb{N}$, it follows by an easy induction from Lemma 4.4.5 which effectively states that $\max_{i\in\{1,\ldots,n\}} \delta(\mathbf{x}_i^{j+1}, \mathbf{x}_i^{j+2}) \leq \max_{i\in\{1,\ldots,n\}} \rho\delta(\mathbf{x}_i^j, \mathbf{x}_i^{j+1})$.

By Equality 4.17 and Inequality 4.22, we obtain:

$$|\mathbf{x}_i^j - \mathbf{x}_i^{j+1}| = \min(\mathbf{x}_i^j, \mathbf{x}_i^{j+1})\delta(\mathbf{x}_i^j, \mathbf{x}_i^{j+1}) \leq \overline{\mathbf{b}}_i \Delta \rho^j,$$

So we have for every $k > j$ (since $\rho < 1$):

$$|\mathbf{x}_i^k - \mathbf{x}_i^j| \leq \sum_{l=j}^{k-1} |\mathbf{x}_i^{l+1} - \mathbf{x}_i^l| \leq \sum_{l=j}^{k-1} \overline{\mathbf{b}}_i \Delta \rho^l \leq \overline{\mathbf{b}}_i \Delta \frac{\rho^j}{1-\rho}$$

which shows that $\mathbf{x}_i^j$ is a Cauchy sequence for every g, and therefore $(\mathbf{x}^j)$ converges to some vector $\tilde{\mathbf{x}}$ for which holds:

$$\left| \frac{\mathbf{x}_i^j - \tilde{\mathbf{x}}_i}{\tilde{\mathbf{x}}_i} \right| \leq \frac{\overline{\mathbf{b}}_i \Delta \rho^j}{\underline{\mathbf{b}}_i (1-\rho)} \leq \frac{(\Delta+1)\Delta\rho^j}{1-\rho}$$

showing the error estimation in the statement of the theorem.

The optimality of $\tilde{\mathbf{x}}$ is a consequence of Lemma 4.4.2, the convergence of $(\mathbf{x}^j)$ and the continuity of $\overline{T}$: $\overline{T}(\tilde{\mathbf{x}}) = \lim_{i=0}^{\infty} \overline{T}(\mathbf{x}^j) = \lim_{i=1}^{\infty} \mathbf{x}^j = \tilde{\mathbf{x}}$. □

4.4.9 Convergence Rate and Error Estimations

One can clearly change the above proofs, so that they apply to the original local refinement presented in Section 4.4.7 where $\mathbf{x}_i$ is optimized relative to the already updated variables and not at once relative to their value in the last global iteration. The proof does not need any further assumptions: it would complicate the notation a bit, but without any essential change to the proof. Theoretically, the same rate of convergence can be established in this case too, but the practical speed of convergence of the algorithm may increase and it can be implemented by an algorithm with reduced memory consumption (only one vector is kept instead of two). Therefore, real life implementations use this version of the local refinement.

Let us discuss the assumptions of Theorem 4.4.6. ρ was defined in 4.19 by

$$\rho' \stackrel{\text{def}}{=} \max_{i \in \{1,\ldots,n\}} \left(1 + \frac{\mathbf{w}_i}{B_i(\underline{\mathbf{b}})}\right)^{-1} \qquad \text{and} \qquad \rho \stackrel{\text{def}}{=} \max\left\{\sqrt{\rho'}, \frac{1+\rho'}{2}\right\}$$

The higher is ρ (or ρ' – the nearer it gets to 1), the slower is the convergence of the local refinement. In the special case of gate- and wire-sizing, $\mathbf{w}_i = 1$ and B

decreases monotonically with the gate and wire sizes. Generally one can say: a reduction of the minimum gate and wire sizes means slower convergence. An increase in the Lagrangian multiplier makes the proven convergence rate worse. Good news is that these parameters do not depend on the topology of the chip. So, for a given library of building blocks (gates and wires), and bounded Lagrangian multipliers the convergence rate can be estimated without knowing the chip. The error estimation in [Langkau 2000] depends on the minimum wiring capacitance of nets and the maximum fanout occuring on the design. This is not necessary anymore.

However the error estimation in [Chu and Wong 1999] and [Langkau 2000] may be sharper than the one presented in the previous section, when applied to the special cases handled by them. Forunately the proof of Theorem 4.21 can be changed so that it yields exactly the same error estimation but with a clearer proof. In fact, one can easily change the proof of Lemma 4.4.5 so that it works with

$$\rho \stackrel{\text{def}}{=} \max_{1\in\{1,\dots,n\}} \left\{ \max \left\{ \left(1 + \frac{\mathbf{w}_i}{B_i(\underline{\mathbf{b}})}\right)^{-1}, \left(1 + \frac{\mathbf{q}_i}{C_i(\overline{\mathbf{b}})}\right)^{-1} \right\} \right\}$$

The difference in the proof of 4.4.5 is that Lemma 4.4.3 is applied to

$$x = C_i(\mathbf{x}),\ x' = B_i(\mathbf{x}'),\ a = \mathbf{q}_i,\ \beta = \rho$$

as well. So we get

$$T_i(\mathbf{x}') = \sqrt{\frac{\mathbf{q}_i + C_i(\mathbf{x}')}{B_i(\mathbf{x}') + \mathbf{w}_i}} \leq \sqrt{\frac{(1+\rho\epsilon)\,(\mathbf{q}_i + C_i(\mathbf{x}))}{\frac{1}{1+\rho\epsilon}(B_i(\mathbf{x}') + \mathbf{w}_i)}} \leq (1+\rho\epsilon) T_i(\mathbf{x}),$$

So, one can show that $\delta(T_i(\mathbf{x}), T_i(\mathbf{x}')) \leq \rho\epsilon$. and therefore the same estimates hold for $\bar{T}_i$. However the achievable improvement is bounded: the number of estimated iterations to achieve a prescribed precision can be at most halved with respect to the original estimation.

One should note that the estimated rate of convergence is quite far from the practically observable. Differences of factor $10-40$ are not seldom. Tests show that $5-10$ iterations are fairly enough for current chips with millions of gates.

To have a practical stopping criterion, one can perform online estimations of the error. In [Langkau 2000] the following method was proposed: Compute two sequences $(\mathbf{x}^{(j)})$ and $(\mathbf{y}^{(j)})$ with initial solutions $\mathbf{x}^{(0)} \stackrel{\text{def}}{=} \underline{\mathbf{b}}$ and $\mathbf{y}^{(j)} \stackrel{\text{def}}{=} \overline{\mathbf{b}}$. It

is easy to show that the sequence $\mathbf{x}_i^{(j)}$ increases monotically while $\mathbf{y}_i^{(0)}$ decreases for each index i. So the following error estimation holds:

$$|\mathbf{x} - \tilde{\mathbf{x}}| \leq |\mathbf{x}^{(j)} - \mathbf{y}^{(j)}|.$$

This method has several drawbacks:

- Local refinement is typically used in subsequent iterations of the subgradient method. The above method prevents the usage of a solution for initial solution in the next round of local refinement.
- It doubles the runtime, as we have to calculate two sequences instead of one.

The first shortcoming is quite serious since the reuse of solution in the next round has immense impact on the overall run-time. To cure this shortcoming, the following error estimation can be used: Compute Δ_j in iteration j of the local refinement by

$$\Delta_j \stackrel{\text{def}}{=} \max_{i \in \{1,\ldots,n\}} \delta(\mathbf{x}_i^{j+1}, \mathbf{x}_i^{j+2}).$$

Obviously, the proof of Theorem 4.21 implies that

$$\left| \frac{\mathbf{x}_i^j - \tilde{\mathbf{x}}_i}{\tilde{\mathbf{x}}_i} \right| \leq \frac{\overline{\mathbf{b}}_i \Delta_j \rho^j}{\underline{\mathbf{b}}_i (1-\rho)}$$

which is typically much tighter than the apriori estimation.

4.4.10 Multistage Signal Shape Propagation

The following question arises naturally:

Can we change the local refinement method so that the signal shapes do not only influence the next stage but their effect is propagated further to a fixed number of subsequent stages?

The problem is that the incoming signals are typically merged at the outputs and only the latest one is kept according to the principle of static timing analysis. The shape of the resulting signal depends on which signals are kept and which are thrown away. If one considers signal merging directly in the formulation of the gate-sizing problem then we have to deal with the following consequences depending on the chosen slew merging strategy (cf. [Lee et al. 2001] and [Vygen 2001]):

- If the signal shape of the latest arriving signal is kept, then the problem gets irreparably nonconvex (even discontinuous).

- If the worst signal shape is kept, or some linear combination of the signal shapes depending on the arrival times, then the resulting problem is posynomial, but the Lagrangian function becomes inseparable, prohibiting an easy use of the subgradient method.

However it is not necessary to merge the signals at each stage. We are free to propagate them to subsequent stages remembering their shape. In fact it even improves the accuracy of the timing anaysis at the cost of additional memory consumption and increase of run-time. However, the effect of the propagation can easily be simulated by adding new propagation segments for the signal paths for which the effect of signal change is to be estimated more precisely. It is not hard to see that the problem of gate- and wire-sizing formulated in Section 4.4.4 is general enough to incorporate these newly added arcs without changing the general form of the program.

Of course, it is computationally infeasible to apply this method to all possible paths in the design. The gain is also questionable since the effect in the change of signal shapes on the delay decreases exponentially. However, for a small subset of critical local situations, for which the decrease is not fast enough, it can mean higher quality solutions. In fact the approach can be extended to generate arcs temporarily if their necessity is recognized.

4.5 Real Life

As already mentioned before, the timing behaviour of the gates may significantly differ from our idealized model. The effect of slew change is only one source of mismatch. The gate libraries are typically discrete and support a finite number of realizations. A more serious problem is that gates with equivalent logical function may be divided into separate classes, each of which scales reasonably well with size, but the classes itself have different delay characteristics.

These problems may explain why approaches based on special delay assumptions are not widely used in the industry in general purpose optimization tools despite their theoretical and practical efficiency.

4.5.1 Timing Optimization Framework

Since there is no exact mathematical model for all aforementioned phenomena which are based partly on manufacturing constraints, one is forced to find general heuristics that work well and efficiently in most cases, but whose optimality is questionable. Now, a framework is sketched that is general and robust enough to handle most everyday chips without much intervention of the designer.

TIMING OPTIMIZATION FRAMEWORK

Input: The netlist of a real world design

Output: Optimized netlist with as few timing violations as possible and minimized power consumption.

① Perform an initial timing analysis on the netlist.

② Update the Lagrangian multipliers according to the timing.

③ Generate new linear delay models based on the timing of the actual netlist.

④ Optimize the weighted sum (according to the Lagrangian multipliers) of power and delay on the chip.

⑤ Perform timing analysis.

⑥ **go to** ② if the improvement is not negligible.

⑦ Do some postprocessing that does not fit well in ④.

Note that expect for postprocessing the design is changed exclusively in ④: local refinement (cf. Section 4.4.7) may be performed here to optimize gate sizing, a full-scale or partial placement may be performed according the actual Lagrangian multipliers, repeater trees can be replaced or more complex logic changes can be done. All these operations may be applied subsequently, possibly iterated multiple times, each time step ④ is reached.

The most important difference between ④ and ⑦ is that in step ④ a trade-off between delay and power-consumption is considered, in step ⑦ a small number of critical hot spots are optimized to remove remaining timing violations disregarding the power consumption. This is typically needed, since at the beginning of the physical design process for harder instances the timing closure (that is timing-feasibility) is often hopeless to be achieved immediately without deeper logic or specification changes and the subgradient method performs

provably optimally only if all timing constraints can be satisfied (with strict inequality). This is typically not the case. The post optimization may perform some local-refinement like strategies to optimize gate-sizes for slack, but also some deeper logic changes based on the information of the latest timing analysis (cf. [Rautenbach, Szegedy and Werber 2003[a-c]).

4.5.2 Lazy Projection

The projection of the Lagrange multipliers was treated in Section 4.3.4. An iterative algorithm with quadratic convergence (based on [Ibaraki, Fukushima and Ibaraki 1991]) was presented. This algorithm performs reasonably well on designs with a moderate number of gates but does not scale well to larger sizes because of its superlinear run-time. Practical tests have already shown in [Chen, Chu and Wong 1999] that even for a pure gate-sizing algorithm without additional features, optimization spends most of its time in the projection of the multipliers. This seems to be unacceptable, especially since this step of the process does not change the design and serves only to yield a rough quantification of the relative criticality of parts of the design. So, it seems plausible that the run-time can be improved by using less accurate estimations of the projection. This can be done by using the newly developed variant of the subgradient method using cyclic projections (cf. Theorem 3.7.4). Doing so, a projection to the flow-space should be computed followed by a projection to the quadrant of nonnegative vectors. This is much faster than a projection to the space of nonnegative flows, but has the same guarantees of convergence.

[Muuss 1999] proposed the following very fast method for doing the job for acyclic timing graphs. Its basic idea is to project the flow locally optimally at each node of the timing graph. The drawback of the method is that it could not be justified theoretically and a guarantee of convergence could not be given. Nevertheless, it works without problems and it is very efficient in practice.

An outflow vector $\mathbf{f}$ at the nodes with required arrival times is maintained and updated in each iteration of the subgradient method. Note that the flow value of the Lagrangian multipliers becomes zero at each node without prescribed (required) arrival time as required by the subgradient method.

LAZY PROJECTION BY KARSTEN MUUSS

Input:

- A Timing graph $G = (E, V)$ with arbitrary real weights $\boldsymbol{\lambda} \in \mathbb{R}^E$.
- Arrival time vector $\mathbf{a} \in \mathbb{R}^V$ and required arrival times $\mathbf{r} \in \mathbb{R}^{V'}$ for a subset $V' \subseteq V$ of the nodes.
- Outflow $\mathbf{f} \in \mathbb{R}^V_{\geq 0}$ with supp $f \subseteq V'$.
- Nonnegative $\delta \in \mathbb{R}_{\geq 0}$.

Output: Nonnegative arc weights $\boldsymbol{\lambda}' \in \mathbb{R}^V_{\geq 0}$ and $\mathbf{f}' \in \mathbb{R}^V_{\geq 0}$ satisfying the flow equalities

$$\sum_{e \in v^-} \boldsymbol{\lambda}'_e - \sum_{e \in v^+} \boldsymbol{\lambda}'_e = \mathbf{f}'_v \qquad \forall v \in V \setminus V'$$

and

$$\mathbf{f}'_v = \begin{cases} \mathbf{f}_v + \delta(\mathbf{a}_v - \mathbf{r}_v) & \text{if } v \in V' \\ 0 & \text{otherwise} \end{cases}.$$

① **for each** node $v \in V$ in reverse topological order **do**
Compute

$$\mathbf{f}'_v \stackrel{\text{def}}{=} \mathbf{f}_v + \begin{cases} \delta(\mathbf{a}_v - \mathbf{r}_v) & \text{if } v \in V' \\ 0 & \text{otherwise} \end{cases}$$

and nonnegative weigths $\boldsymbol{\lambda}'_e$ on the incoming arcs e of v such that:

$$\sum_{e \in v^-} \boldsymbol{\lambda}'_e = \sum_{e \in v^+} \boldsymbol{\lambda}'_e + \mathbf{f}'_v$$

and $\sum_{e \in v^-} ||\boldsymbol{\lambda}'_e - \boldsymbol{\lambda}_e||_2^2$ is minimized.

Note that using the reverse topological processing order guarantees that $\boldsymbol{\lambda}'_e$ is already computed for each arc e leaving v when v is processed.

The local projection of the Lagrangian multipliers can be performed very fast by successively solving simple linear equations. The practical run-time of the algorithm is linear in the size of the timing graph. The outflow vector is stored and updated in each step together with the Lagrangian multipliers.

Modified versions of the above heuristic are thinkable. For example the introduction of a supernode for the nodes with required arrival time may be benificial to smooth the effect of the update. Another possibility is to project the multipliers forward instead of backward or alternately in both directions.

Several of these variants were tested in practice, but the overall effect is hard to predict as it may change from instance to instance. Sometimes a variant performs a bit better in practice even than the exact projection, but it is an open question whether a theoretical guarantee for the subgradient method using one of these "lazy" updates can be given.

One modification of the original method emerged as generally very useful: the original scheme tends to forget criticalities as fast as it learned them. A cure for this behaviour is to update the criticalities at the endpoint by a nonlinear function of the slack s: for example, we add αs to the outflow if $s \leq 0$ and βs if $s > 0$ where $0 < \beta < \alpha$. This method effectively prevents the criticality of a path from being forgotten immediately after the timing violation at the endpoint is removed. This trick improved the practical performance of the algorithm enormously.

One must also note that although the above algorithm requires an acyclic graph, it can be adapted to arbitrary timing graphs if a suitable subset of arcs is snipped for the method. This implies that the flow inequalities are violated on some nodes, but it does not seem to impair the practical performance of the algorithm. A particularly interesting case consists in variable clock arrival times. In this case, the timing analysis must incorporate a clock skew scheduling in each iteration of the subgradient method. This can be efficiently solved using modern clock skew scheduling algorithms that can work without too much run-time or memory overhead (cf. [Held 2001] and [Held et al. 2003]).

4.5.3 Adaptive Gradient Scaling

The proof of the convergence of the subgradient method using the divergent series rule does not give any hints about the convergence rate. Typically, estimations about the convergence rate of subgradient methods can only be given if the condition of the problem is known (cf. [Minoux 1986] Chapter 4.3). This is not only a theoretical problem, since a wrong step-size can considerably impair the performance of the algorithm.

Here a new heuristic rule will be presented without proof of convergence guarantees. The basic idea is that if the scalar product of two successive gradients is

negative, then the step size was too high since the gradient must have "turned back". If the gradient is about zero then the scaling factor size for the first step was approximately right, if it is highly positive then the scaling factor was too low. This suggests that the scaling factor should be corrected according to the angle of successive subgradients.

This intuition immediately gives rise to the **Adaptive one step lookahead scaling factor adjustment** method:

Before beginning with the subgradient method, we fix a threshold $t \in [-1, 0]$, a positive α and a natural number m.

Assume that we are in iteration k and process $\mathbf{x}^k \in \mathbb{R}^n$, the last gradient direction $\mathbf{g}^k \in \mathbb{R}^n$ is computed with norm $\|\mathbf{g}^k\| = 1$ (if the subgradient was 0 then no adjustment is needed) and the current scaling factor is $\rho \stackrel{\text{def}}{=} \rho^k$. Perform an additional *lookahead* gradient computation at $\mathbf{x}^k + \rho\mathbf{g}^k$ resulting in the next temporary gradient direction vector $\mathbf{g}$ of length 1. Compute

$$p \stackrel{\text{def}}{=} \langle \mathbf{g}^k, \mathbf{g} \rangle$$

and set

$$\rho \leftarrow \exp(\alpha p)\rho.$$

If $p > t$ then we decide to keep $\mathbf{g}$ as the next $\mathbf{g}_{k+1}$ gradient direction otherwise we recompute it by setting it to the subgradient of f at $\mathbf{x}^k + \rho\mathbf{g}^k$. Note that ρ has changed since the last computation. Continue with the scaling process. However, this scale-down process inside one iteration could loop infinitely at a suboptimal $\mathbf{x}^k$. To avoid this, we abort the loop after at most m local iterations. After the scaling down is ready, we set

$$\mathbf{g}^{k+1} \stackrel{\text{def}}{=} \mathbf{g} \qquad \mathbf{x}^{k+1} \stackrel{\text{def}}{=} \mathbf{x}^k + \rho\mathbf{g}^{k+1} \qquad \rho^{k+1} \stackrel{\text{def}}{=} \rho$$

and proceed with the next iteration of the subgradient method.

The method has several advantages: it discards the result of the gradient computation quite rarely and adapts quickly to the local form of the function. A problem is that ρ is global for each direction. The typical situation is that different optimization directions require different step sizes. Space dilatation techniques (cf. [Minoux 1986] Chapter 4.3) promise to cure this, but they require storing a transformation matrix with quadratic size in the number of the variables and therefore infeasible in our situation. So we propose a simplified method where we only consider diagonal dilatation matrices.

The modification is that instead of one global ρ we maintain a full vector $\boldsymbol{\rho} \in \mathbb{R}^n$. Its components are updated in each iteration by:

$$\boldsymbol{\rho}_i \leftarrow \boldsymbol{\rho}_i \exp(\alpha p |\mathbf{g}_i^k|)$$

The next point of the subgradient method is defined by

$$\mathbf{x}^{k+1} \stackrel{\text{def}}{=} \mathbf{x}^k + \langle \boldsymbol{\rho}, \mathbf{x}^k \rangle \mathbf{g}^k.$$

This approach allows more accurate adaption to the form of the objective function at an almost neglectible run-time overhead.

4.5.4 Update of the Delay Models

The use of gate-sizing and timing driven placement methods requires a linear delay model. In fact using a fixed set of linear coefficients may lead to reasonably optimized chips but a significant portion of the cycle time can be lost by inaccuracies and mismatch between these models and the more accurate reference timing engine. Most industrial implementations performing optimization in an analytical way use fixed models with questionable success.

An elegant and simple solution seems to be the update of delay parameters in each iteration of the subgradient method. Modern timing engines typically permit to query the $\frac{\partial \text{delay}}{\partial \text{load}}$, $\frac{\partial \text{delay}}{\partial \text{in-slope}}$ $\frac{\partial \text{out-slope}}{\partial \text{load}}$, $\frac{\partial \text{out-slope}}{\partial \text{in-slope}}$ values for each propagation segment at any measurement point. These derivatives can be used to extract linear delay models matching the local situation exactly. As the subgradient method proceeds the variance of the capacitive load and signal slope decreases and therefore the accuracy of the linear delay models increases.

Another problem is posed by the discreteness of the cell library. It proved to be useful to introduce *virtual cells* to make the library continous. A virtual cell consists of a real cell and a linear coefficient μ close to 1. The size of a virtual cell the is μs where s is the size of the real cell. The introduction of virtual cells smoothens the problem and leads to better convergence as opposed to rounding the gate sizes in each iteration of the sugradient method. Of course, it requires a sufficiently sophisticated timing engine that can perform timing analysis on designs with virtual cells.

4.5.5 Local Refinement in Practice

The local refinement method which has been introduced theoretically can be used in practice without change and obtains fairly good results without further tweaking when dynamically updated linear timing models are used and if the cell classes contain only scalable cells.

If logically equivalent cells are partitioned into cell classes with different characteristics, then the following hybrid analytical-combinatorial approach can be used:

For each set of logically equivalent cells, we choose a default class which is used for most gates of the design. In each iteration of the subgradient method, a small subset of gates is marked as critical. A good measure for the criticality of a gate is the sum of Lagrangian multipliers on each arc leaving it. A prescribed percentage of the most critical gates can be computed in linear time in the number of gates by a median algorithm. For each critical gate g, all the locally optimal realizations from each cell class can be selected for g. Inside a cell class, one can use the analytical formula (4.15) to quickly compute the relatively optimal size. Among all solutions the one minimizing the objective function $Ł_1$ is chosen.

Analogously, one can integrate more complex operations in the local refinement like the replacement of full invertertrees or replacement of some subset of the gates or resynthesis of larger pieces of combinatorial logic if it decreases $Ł_1$.

Although the local refinement in its pure form is quite fast, it can take considerable time. The following practical trick can reduce the run-time significantly while preserving provable and practical convergency:

Introduce a label $\in$ {**fixed,unfixed**} for each gate. Before the first iteration of the local refinement the value of the label at each gate is set to **unfixed**. Every time gate g is processed, its label is set to **fixed**. If the relative change of the iteration was above a prescribed threshold, then all neighbours of g (all those gates significantly affected by the change of g) are set to **unfixed**. The proof of Theorem 4.21 gives an explicit method to compute the required threshold to guarantee a specified accuracy. In practice, the run-time of the local refinement could be reduced using this trick by a factor of 3 to 10 (increasing with the specified accuracy). As a rule of thumb, the current implementation computes a suffeciently precise solution in half of the time of a timing analysis. Of course this excellent speed could not be achieved by a timing engine without well optimized support for virtual cell assignment.

4.5.6 The Method of Gain Restriction

In most cases, for the designers (users of the timing optimization framework) power optimality is only a secondary objective. The most important criterion is that the optimizer finds a solution with as few timing violations as possible and such solutions should be found as quickly as possible. The subgradient method in its original form converges quite slowly, indeterministically and reacts very sensitively to the change of the parameters like the step size.

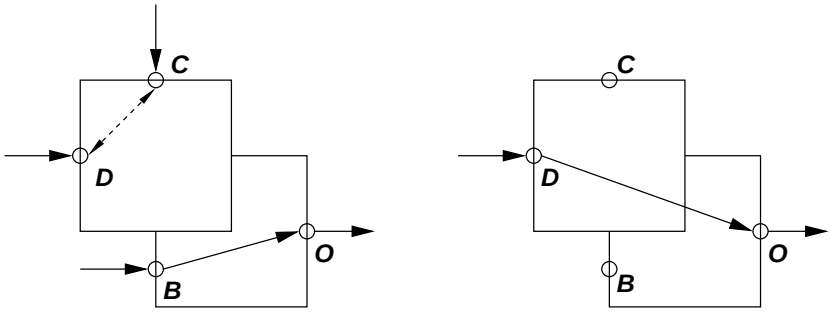
The following heuristical method to improve the behaviour of the algorithm was tested and found to be extremely efficient:

We do not only prescribe minimum and maximum size for the gates, but maximum gains. The *gain* is the ratio of the gate's output load and its size. A small gain guarantees that the delays of all arcs of the gate are not too high and the signal shapes remain of good quality. Of course, low gains for all gates are unacceptable and would cause unnecassarily high power consumption (it could even impair the timing behaviour). So the maximum allowed gains are increased for uncritical gates in subsequent iterations.

The gain restrictions are considered in the local refinement. Every time a gate is processed, its lower bound depends on the gain restriction. If the maximum allowed gain is g and the driven load is l then the minimum bound is set to $\max\left\{\max(\underline{b}, l/g), \overline{b}\right\}$, where $\underline{b}$ and $\overline{b}$ denote the absolute lower and upper bounds on its size, respectively. The so modified local refinement may not converge linearly, since the loads depend on the sizes of all driven gates, which change during the process. However, a suitably tuned gain-restricition may accelerate the convergence considerably and, what is more important, it makes it more robust: in the non-restricted version, after several iterations previously completely uncritical paths can suddenly become highly critical and it is hard to find a feasible solution. The method of gain restriction prevents this unwanted behaviour.

4.5.7 Interaction with Clock Skew Scheduling

As we already mentioned, the method of Lagrangian relaxation can be easily applied in connection with clock skew scheduling: simply the arrival times at the clock pins should be left variable. However sometimes a slight modification of the timing graph is necessary to incorporate cycle adjustments into the model. The following figure illustrates the applicaton of this techinque in the presence of master-slave memory cells.



On the right hand side, we can see the timing graph induced by the clocking of a master-slave memory cell. The dashed line indicates a late mode test between the master clock pin C and the data input pin D.

The signal arriving at the slave clock pin B is exactly the inverse of the periodic clock signal arriving at C. The timing arc between clock pin B and the data output pin O indicates that the release of the signal leaving the memory cell at O is controlled by the clock signal at B. We assume that we can shift the (coupled) arrival time at the clock pins B and C arbitrarily. This means that the best way is to choose the clocking so that the clock closes the latch exactly when the data signal arrives at D. However, due to the coupled arrival time of the signals at C and B, this imposes a late mode constraint on the release time of the signal at ouput pin O: This can be modeled by a single propagation arc from D to O (see left hand side) whose delay is the sum of the setup time at D, the delay of the propagation segment BO minus the cycle time of the clock plus additional safety margins (for example, due to clock jitter). The delay is normally negative, since the cycle time dominates all positive terms.

After performing this substitution at each memory cell of the design, we typically arrive at a timing graph with directed cycles. Still, we can apply the subgradient method on it to optimize the gate sizes or the placement. The subgradient method converges provably since both the projection and the local refinement works on graphs with oriented cycles.

In order to update the delay models, we have to propagate on the design. This can be done by the clock skew scheduling methods presented in [Albrecht 2001], [Albrecht et al. 2002], [Held 2001] and [Held et al. 2003].

4.5.8 Interaction between Timing and Placement

A natural question is whether one could provably optimally place and size the gates of a design simultaneously. The problem is that while the gate-sizing

can be transformed to a convex problem and the timing driven placement with convex netlength estimation is a convex problem, their combination is not convex. It can not be ruled out that it could be transformed to a convex problem, but it is not easy if possible at all.

However, the following heuristic is applicable:

TIMING DRIVEN PLACEMENT WITH GATE-SIZING

Input: An instance of the combined gate sizing and placement problem.

Output: A non-overlapping placement with resized gates.

① Set the Lagrangian multipliers to 0.

② Iterate local refinement and weighted placement alternatingly with respect to the given Lagrangian multipliers. The placement is fixed in the local refinement and the gate-sizes are fixed for the placement.

③ Retime the chip and update the delay models.

④ Recompute the Lagrangians by some projection method (cf. sections 4.3.4 and 4.5.2).

⑤ **go to** ② if the improvement is not neglectible.

⑥ If partitioning is needed then perform one and go to ②.

⑦ Perform detailed placement.

Of course, the stopping criterions in ② and ⑤ have to be chosen suitably. Since the duality gap is hard to compute, theoretically not justified conditions must be used.

Another complication is that the gate sizes change during the process, so previously underfull regions can become overfull due to increased cell sizes. This problem typically can be ignored and handled in the detailed placement.

The Timing Driven Placement algorithm can be easily combined with blow-up heuristics [Brenner and Rohe 2002] to improve routability.

The result of the algorithm is a non-overlapping placement. The quality of the solution is hard to estimate. However this is also true for all known quadratic placement methods with partitioning, even those disregarding the timing constraints.

The theoretical foundations of the above algorithm are much firmer then other netweighting heuristics currently in use. The reason is that if the gate sizes are

fixed and no partitioning is performed than the method converges provably. The weights are computed using the information about the driving strength of all relevant timing arcs.

4.6 BonnTime

The BonnTime timing optimization framework is developed in the Research Institute for Discrete Mathematics, University of Bonn, within cooperation project with IBM. The main developers of the BonnTime project are Stephan Held, Jürgen Werber, Christoph Bartoschek and the author.

The BonnTime timing optimization tool is written in C++ and consists of several different well-encapsulated parts. The main components are:

- A complete self-contained timing engine.
- A clock-skew scheduling algorithm.
- A timing optimization framework as described in section 4.5.1.
- A repeater tree optimizer.

BonnTime is designed to be integrated with other BonnTools like the BonnPlace and IBM-internal CAD tools. It can be compiled as a standalone executable, but is typically used as a dynamically loadable library (DLL) and can be scripted in the command language Tcl. BonnTime can work with timing rules conforming to the IEEE 1481-1999 CDC/DCM standard [IEEE 1999].

4.6.1 Architecture of the Timing Engine

In the preceding sections, the task of performing a static timing analysis was assumed to be a trivial, efficiently performable task. However, on real life chips with real cell libraries, it is very involved. Here we do not go into the details of the implementation, just give a superficial overview of the main features, architectural choices and how the timing engine of BonnTime differs from the ones currently in industrial use.

The timing engine has a modular architecture consisting of well separated, exchangeable components communicating exclusively over clearly specified object-oriented interfaces.

The timing engine supports plugins for different functions. For example, the timing rules, wiring estimations, propagation-strategy and assertion reader can be registered and removed during runtime. This can be useful for optimization algorithms requiring different levels of accuracy for controlling the behaviour of the timing engine, and the Tcl interface to the plugins empowers the designer to change the behaviour of the timing engine according to their needs.

The timing engine is designed to be used in a shared memory multithreaded environment. Different timing engines can be instantiated at the same time for different and overlapping parts of the same chip. The timing engines do not use any global variables, and they can have separate, disjoint memory heaps and loggers to prevent run-time degradation caused by pending locks.

These are the main parts of the timing engine:

- **Placement manager:** Controls and caches queries about placement. Responsible for the legalization and placement queries of gates and pins.
- **Physics manager:** Controls and caches queries about physical information like wire delay estimations, wiring planes, wire classes, and other physical parameters like delay coefficients, voltage and temperature.
- **Rule manager:** Responsible for delay queries, and topology information of the timing cells. It is the gateway to the CDC/DCM delay rules.
- **Assertion manager:** One of the most involved parts of the timing engine. Responsible for handling all user defined timing constraints like asserted and required arrival times and signal shapes, physical information, but also more sophisticated types of assertions like those for multicycle and false paths. It contains a full fledged phase manager.
- **BTD Assertion reader:** Export and import of raw (uninterpreted) assertions. It is used to convert assertions of IBM-internal format to BTD (Bonn Timing Data) format.
- **Graph Manager:** Responsible for building up and changing the topology the timing graph. It manages the matching between the netlist and the timing graph. The most complicated part of this module is the subnetwork support which can be used to cut out, analyse, change and reinsert arbitrary pieces of a timing graph. The extracted part is a completely separate timing engine that can be changed and timed independently in a thread-safe way.

- **Propagator:** The component reponsible for the timing analysis on a single pin of the design. This is not as simple as it looks at first sight, since the timing information on a single point has a considerable internal structure: it holds early/late-mode arrival time and signal shape information of each phase propagated over the pin. Moreover the delay information on the timing arc must be adjusted and stored considering all assertions and other types of adjustment. It also manages additional information like delay and slew gradients.

- **Propagation strategy:** Controls the validation and invalidation of timing information on the nodes. Since a change in the netlist typically influences only the timing of a small part of the chip, it is desirable to invalidate only the influenced part of the design. A recomputation is performed lazily: on a query of some timing information only the necessary parts analysed again. Different optimization algorithms may require different strategies for optimal efficiency, so the propagation strategy is realized by a plugin.

- **Logic manager:** Controls queries about the logical and timing equivalency of cells and virtual gate sizes.

- **Report manager:** Able to write all kinds of handy reports.

- **Scripting interface:** Registers scripting (currently Tcl) commands that allow a flexible external control of the timing engine.

4.6.2 Assertions in BonnTime

The timing engine contained in BonnTime is capable of handling all kinds of design specifications (so called *assertions*) needed for the timing of real world chips. This does not only mean specifications of (required) arrival times at the boundary of timing graph and obligatory physical parameters like temperature or voltage and primary output capacitances, but a multitude of different adjustments to the propagated values of the design. There are simpler assertions like an addition of some value to the arrival time of a node, but there are also more complicated ones ranging from the inclusion of completely new timing nodes and arcs with specified attributes to the exclusion or adjustment of signal paths passing certain points.

Unfortunately, these assertions were added to other timing engines in a successive way on the request of designers over the years, so industrially used timing

engines allow for hundreds of slightly different atomic assertion types whose number is increasing even today.

In order to cope with future assertions, instead of ad hoc mechanisms, the BonnTime engine uses a more structured approach: the assertions are decomposed into more basic building blocks that can be handled in a well-defined orthogonal manner. Each BonnTime assertion consists of the following components:

subject	attribute	operation	flags	value

For example: If the required arrival time for the rising edge for phase p at node v is overriden by the value 1.0, this is specified by the following assertion:

subject	attribute	operation	flags	value
phase p at node v, rising	arrival time	override	limit	1.0

Of course, this is a very simple example. The subject can be specified in a lot of different ways: for example, it may be the more abstract "head of arc e" which is different to the head node of arc e, since it makes a difference to adjust a propagated signal *before* it is merged into the head node instead of the already merged signal. Priority specification between phase renames and other assertions complicates the picture further. The subject may be as complex as the set of endpoints of paths for which a specified predicate holds. Of course, the implementation of handling such complicated assertions requires sophisticated approaches under the hub as well.

The advantage of the decomposed approach is that once an aspect is enhanced (for example a new type of subject is added), the combinations with all orthogonal aspects (like operation) are available immediately. For example, once "endpoints of paths passing certain points" are accepted as subject, any operations that work on nodes will work on them too.

4.6.3 Cell Classes in BonnTime

One of the most distinguishing features of BonnTime is a full scale support for virtual cell sizes and mechanisms to allow fast estimations of the effect of changing of some cells.

As we have seen in Section 4.4.7, one of the most time-consuming parts of the timing optimization is the local refinement step which has to update all cells of the design several times in one iteration of the subgradient method. The

number of cells on current designs is in the millions, so the resize step should be performable as fast as possible.

The naive approach is to simply update the cell assignment in the netlist and let the timing engine handle the update of the timing graph by callbacks. Unfortunatly, the update of the netlist is a costly operation itself, and the propagation of its effect to the timing graph consumes considerable run-time.

The solution implemented in BonnTime is much faster than this standard procedure: it allows for changing the cell assignment *without updating a netlist*. This means that full matching between the timing graph and the netlist is temporarily violated, but a lot of run-time is saved. In fact, the change of the cell associated with the gate can be updated in BonnTime by the change of one single pointer. However, this requires that the gate is switched to a so called "virtual mode" that allows such changes. This change makes the analysis marginally slower, since it adds an extra level of indirection to the query of timing information: the actual gate that the virtual cell is based on must be queried at each occasion where the timing analysis needs any cell-specific information. At the first sight it may look as costly as updating all this information once for each change. In fact, tricky datastructures guarantee that cell-specific information can be queried almost as fast as in the normal mode, and significant run-time can be saved compared to a real update. These data structures are built up once when the database for cell classes is initialized; switching a gate to virtual mode can then be performed efficiently.

The management of the virtual cell assignment handles the virtual gate sizes: since cell libraries are typically discrete, but analytical methods require continuous libraries, BonnTime allows for continuous scaling between existing cells. Of course, these sizes are not realizable and cannot be written back to the netlist, but have to be rounded finally. However their intermediate use is indispensable for achieving practical convergence. Virtual cell sizes require deep support by the timing engine. For the analysis, each time a delay and slew (slope of the signal shape) function of the timing rule is called, the values must be adjusted according to the size and capacitances of the input pins (required for computing the wiring capacitance) must be scaled as well. This does not happen just once, but each time the influenced quantity is queried.

This support allows the BonnTime timing optimization framework to spend only about half as much time in local refinement as in the timing analysis.

4.7 Experimental results

The gate-sizing algorithm was implemented and tested within the BonnTime environment. The implementation contains the implementation tricks described in the last section and an additional combinatorial postprocessing optimizing the slack directly on the most critical gates of the design. The criticality of a gate is measured by the sum of the Lagrangian multipliers on the arcs leaving the gate. The postoptimization step is extremely simple: it processes a prescribed low percentage of the most critical gates in reverse topological order. This step increases the power consumption slightly, but sometimes it improves the worst slack by some hundred picoseconds on ASIC designs with 130-180 nm feature-sizes and long critical paths.

The methodology of the test was the following:

(1) The design was placed by the BonnPlace quadratic placement tool.

(2) It was optimized by the traditional DelayOpt optimization suite, which includes gate-sizing and repeater-tree insertion. (All existing repeater-trees were removed.)

(3) The sizes of the gates were reset to the minimum possible in their class.

(4) BonnTime gate-sizing was applied to the design.

After the procedure, the power consumption and worst slack of the results of steps (2) and (4) were compared. The run-time of the gate-sizing is of interest too, however it cannot be fairly compared with that of the combinatorial optimization tool, since DelayOpt performs complex operations and it inserts repeater trees too. However in each of the tests, BonnTime was at least ten times faster (even if DelayOpt was restricted to perform only gate-sizing).

A further restriction was that for the gate-sizing only a subset of logically equivalent resize operations were allowed. For example, a change between balanced and inbalanced gates was artificially prohibited, even in the postoptimization phase. This was necessary since the tool has no built-in library analysis feature and implicitly assumes a linear scaling of the cells.

All chips were processed with the same parameter set and only seven iterations of the subgradient method were performed.

The power column refers to the power consumption of the sizeable cells compared to the original solution. It is computed by adding the input capacitances of all sizeable cells. Clock-tree, wiring and large macros were ignored.

The Δslack column refers to the change of worst slack with respect to the solution found by the combinatorial tool. A positive value means improvement.

The runtime was measured on an IBM S85 workstation with 600Mhz RS6KIII processors using the industrial timing rules supplied in executable form. More than one third of the run-time was spent in the evaluation of these models. The optimization and timing propagation was not multithreaded. The run-times are given in hours:minutes.

name	technology	cycle time	#gates	power	Δslack	run-time
Matthias	180nm	8ns	480K	88%	38ps	1:28
Werner	180nm	4ns	770K	59%	420ps	0:51
Bernhard	180nm	4ns	560K	92%	75ps	0:38
Max	180nm	7.5ns	440K	71%	-12ps	0:24
Ulrich	130nm	6.6ns	2070K	76%	250ps	3:12
Hanno	130nm	3.3ns	530K	89%	-20ps	0:45
Alex	130nm	2.5ns	750K	87%	30ps	1:44
Walter	130nm	1.6ns	120K	108%	90ps	0:33

The slack is sometimes slightly improved. The increase in power-consumption is due to the applied convergence-accelerating heuristics. A pure implementation sometimes yields significantly lower power-consumption, but the number of iterations to achieve an acceptable slack increases to more than 50. The power-overhead of the heuristics depend on the ratio of critical gates in the designs. There were examples of highly critical macros where the chip became overfull if all these tricks were turned on.

Conclusions

In this work, we have inspected the weighted combinatorial Laplacian of graphs in different contexts. This matrix proved to be not only theoretically intriguing, but also very useful for designing practical algorithms for combinatorial and analytical optimization problems. Different weightings of the Laplacian can highlight important connections and operations between the two most basic algebraic invariants of graphs: the cycle- and cocycle-subspace.

All applications of the combinatorial Laplacian have something in common: they are always connected to the orthogonal projection operator to the cycle- or cocycle-space. However, these operators are not directly defined by the Laplacian matrix.

For base fields of characteristic 0, these projections always exist. In nonzero characteristics, they exist only if the bicycle-space of the cocycle-space vanishes (that is, the induced scalar product is nondegenerate), otherwise the matroid defined by the bicycle-space describes those minimal subsets of the "edges" whose contraction results in a space with nondegenerate scalar product. The dimension of the bicycle-space is the corank of the Laplacian matrix minus the corank of the graph. However, in characteristic 2 symplectic weightings of a graph play an important role: if there is a projection matrix with respect to the so-weighted cocycle-space, it is necessarily symmetric and alternating and defines a Δ-matroid that describes those edge-sets whose contraction results in a space with nondegenerate scalar-product. Under the symplectic weightings of the edges, the most generic ones (whose weighted Laplacian defines the "richest" Δ-matroid) are distinguished. A graph is factor-critical if the so weighted cutset-space is nondegenerate (with respect to the induced scalar product). These results extend to matroids.

In VLSI design, weighted quadratic netlength minimization, which is a key step of the timing-constrained placement problem, can also be viewed as a projection to the weighted cocyle-space of the gate-graph. Therefore it is

not surprising that the Laplacian matrix of the gate-graph has been used for decades to minimize the quadratic netlength. We also demonstrated that this can be used as a subroutine in an algorithm minimizing the weighted mixed linear-quadratic netlength, a problem which arises naturally as a subproblem of the timing driven placement problem. The approach presented in this work is not merely slack-driven, but considers the individual driver-strengths of the involved gates and the Elmore delays of the wires.

Generally, in the timing-constrained optimization, the projection step to the cycle-space of the augmented timing graph turned out to be crucial. A good intuitive explanation for this phenomenon was given in Section 4.3.4 where it became clear that the projection step is closely related to the timing propagation: in fact, it distributes the slack evenly without an explicit static timing propagation. This raises hopes that the robustness of clock-skew-scheduling could be improved by similar projection techniques.

As an open problem the question remains whether the combined placement and gate-sizing problem can be optimized efficiently if the disjointness constraints for the cells are ignored. Although both special cases (the timing driven placement and the gate-sizing problem) can be transformed to convex problems and can be solved by the dual subgradient method, their combination does not seem to be so well-behaving. Still there is hope that a global optimum can be found reasonably fast.

Appendix A

Notation

$\mathbb{R}$	Set of real numbers
$\mathbb{P}$	The two dimensional Euclidean plane $\mathbb{R}^2$
$\mathbb{R}_{\geq 0}$	Set of nonnegative real numbers
$\mathbb{R}_{>0}$	Set of positive real numbers
$\mathbb{N}$	Set of natural numbers (excluding 0)
GF(q)	Finite field with q elements
$k[x_1, \ldots, x_n]$	Polynomial ring over field k in indeterminates $x_1, \ldots, x_n$
$k(x_1, \ldots, x_n)$	Field of rational functions over field k in indeterminates $x_1, \ldots, x_n$
R/I	Factor ring of ring R with respect to ideal I
$Q(R)$	Quotient field of integral domain R
$M[S]$	Matroid induced by the subset S of edges
$M \setminus S$	Matroid resulting from the deletion of subset S of edges
M/S	Matroid resulting from the contraction of subset S of edges
$M \div S$	Matroid resulting from the subdivision of subset S of edges
M^*	Dual of matroid M
$\mathrm{rk}\,(M)$	Rank of matroid M
$U \setminus S$	(E, K)-space resulting from the deletion of subset S of edges
U/S	(E, K)-space resulting from the contraction of subset S of edges
$U \div S$	(E, K)-space resulting from the subdivision of subset S of edges
$U^\perp$	Orthogonal space to (E, K)-space U
$\mathcal{M}(U)$	Matroid associated with space U
$\mathrm{S}(U)$	Symplectification of space U
$\mathcal{B}(U)$	Bicycle space $(U \cap U^\perp)$ of space U
$\beta(U)$	Dimension of the bicycle space of U
$\boldsymbol{\chi}_S$	Characteristic vector of $S \subseteq E$ in (E, K)-space U
$A[S]$	Principal submatrix induced by the the subset S of columns
$A[S_1, S_2]$	Submatrix induced by the subset S_1 of rows and S_2 of columns
$\mathrm{rk}\,(A)$	Rank of matrix A
$\mathcal{M}(A)$	Column matroid of matrix A

$G[S]$	Subgraph induced by the subset S of nodes
G/S	Graph resulting from the contraction of the subset S of arcs
$G \setminus S$	Graph resulting from the deletion of the subset S of arcs
e^-	Tail of arc e
e^+	Head of arc e
v^-	The set arcs e with $e^+ = v$
v^+	The set arcs e with $e^- = v$
$\deg^-(v)$	The indegree of node v: $\lvert v^- \rvert$.
$\deg^+(v)$	The outdegree of node v: $\lvert v^+ \rvert$.
$T(G)$	Analogon of the Tutte matrix of graph G
$D(G)$	Node-edge incidence matrix of graph G
$\mathcal{M}(G)$	Cycle matroid of graph G

π_X	L_2-Projection onto $X \subseteq \mathbb{R}^n$.
$d(\mathbf{x}, \mathbf{y})$	Euclidean distance between two points in $\mathbb{R}^n$.
$d(X, \mathbf{x})$	Euclidean distance between a set and a point in $\mathbb{R}^n$.
$d(X, Y)$	Euclidean distance of sets X and Y in $\mathbb{R}^n$.

Bibliography

[Albrecht 2001] C. Albrecht, *Optimierung der Zykluszeit und der Slackverteilung und globale Verdrahtung,* (in German) Ph.D. thesis, University of Bonn, 2001.

[Albrecht et al. 2002] C. Albrecht, B. Korte, J. Schietke and J. Vygen, Maximum Mean Weight Cycle in a Digraph and Minimizing Cycle Time of a Logic Chip. *Discrete Applied Mathematics* **123** (2002), pp. 103-127.

[Alon 1986] N. Alon, Eigenvalues and expanders, *Combinatorica* **6,** (1986), pp. 83-96.

[Alpert et al. 1997a] C.J. Alpert, T.Chan, D.J.-H. Huang, I. Markov and K. Yan, Quadratic Placement Revisited, *Proceedings of the 1997 ACM DAC*, (1997), pp. 752-757.

[Alpert et al. 1997b] C.J. Alpert, T. Chan, D.J.-H. Huang, A.B. Kahng, I.L. Markov, P. Mulet and K. Yan, Faster minimization of linear wirelength for global placement *Proceedings of the 1997 international symposium on Physical design* (1997) pp. 4-11.

[Alpert, Kahng and Yao 1999] C.J. Alpert, A.B. Kahng and S.-Z. Yao, Spectral partitioning: The more eigenvectors, the better, *Discr. Appl. Math* **90**, (1999), pp. 3-26.

[Anderson and Morley 1971] W.N. Anderson jr. and T.D. Morley, Eigenvalues of the Laplacian of a graph, *Technical Report TR 71-45*, University Maryland 1971.

[Arora 1996] S. Arora, Nearly linear time approximation schemes for Euclidean TSP and other combinatorial problems, in *Proc 38. Ann. Symp. on Foundations of Comput. Sci.*, (1997), pp. 554-563.

[Bazaraa, Sherall and Shetty 1993] M. S. Bazaraa, H. D. Sherall and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, John Wiley & Sons, New York 1993.

[Bouchet 1988] A. Bouchet: Representability of Δ-matroids, Combinatorics, *Proc. 7th Hung. Colloq., Eger/Hung. 1987*, Colloq. Math. Soc. János Bolyai **52** (1988), pp. 167-182.

[Boyle and Dykstra 1986] J. P. Boyle and R. L. Dykstra, A method of finding projections onto the intersection of convex sets in Hilbert spaces, in: *Advances in Order Restricted Statistical Inference* **37**, Lecture Notes in Statistics, Springer Verlag, Berlin, Germany, 1986, pp. 28-47.

[Brandes et al. 2000] U. Brandes, G. Shubina, R. Tamassia and D. Wagner, "Fast Layout Methods for Timetable Graphs," in: *Graph Drawing* (2000), pp. 128-138.

[Brenner and Rohe 2002] U. Brenner and A. Rohe, An Effective Congestion Driven Placement Framework, *ISPD 2002,* pp. 6-11.

[Brenner and Vygen 2002] U. Brenner and J. Vygen, Worst-Case Ratios of Networks in the Rectilinear Plane *Networks* (2002), **38**(3), pp. 126-139.

[Burnstein and Youssef 1985] M. Burstein, M.N.Youssef, Timing Influenced Layout Design, *Proc. DAC IEEE ACM 1985,* pp. 124-130.

[Buschke and Kruk 2002] H.H. Bauschke, S.G. Kruk, *The method of reflection-projection for feasibility problems with an abtuse cone* Technical Report, Oakland University (2002).

[Cabot, Francis and Stary 1970] A.V. Cabot, R.L. Francis and M.A. Stary. A Network Flow Solution to the Rectilinear Distance Facility Location Problem *AIIE Transactions* **2** (1970), pp. 132-141.

[Chen, Chang and Wong 1996] C. Chen, Y.-W. Chang and M. D. F. Wong, Fast Performance-Driven Optimization for Buffered Clock Trees Based on Lagrangian Relaxation, *Proc. ACM/IEEE Design Automation Conference*, (1996) pp. 405-408.

[Chen, Chu and Wong 1999] C. Chen, C.C.N. Chu and M.D.F. Wong, *Fast and Exact Simultaneous Gate and Wire Sizing by Lagrangian Relaxation*, Proc. IEEE Trans. Computer-Aided Design, Vol. **18**, (1999), 1297 ff.

[Cheney and Goldstein 1959] W. Cheney and A. Goldstein, Proximity maps for convex sets, *Proceedings of the AMS* **10** (1959), 448-450.

[Chu and Wong 1999] C. C. N. Chu, M. D. F. Wong, Greedy Wire-Sizing is Linear Time, IEEE Trans. Computer-Aided Design, Vol. **18** (1999), 398 ff.

[Cong and He, 1996] J. Cong, L. He, Simultaneous Transistor and Interconnect Sizing using General Dominance Property, Proc ACM, SIGDA Workshop Physical Design, (1996), pp. 34-39.

[Coullard and Hellerstein 1996] C.R. Coullard and L. Hellerstein, Independence and port oracles for matroids, with an application to computational learning theory, *Combinatorica* **16** No.2 (1996) pp. 189-208.

[Cunningham and Geelen 1997] W.H. Cunningham and J.F. Geelen, The optimal path-matching problem, *Combinatorica* **17** (1997), pp. 315-337.

[Dax 1986] W. Dax, An efficient algorithm for solving the rectilinear multifacility location problem, *IMA J. Numer. Anal.* **6**, (1986), pp. 343-355.

[Donath and Hoffman 1973] W.E. Donath and A.J. Hoffman, Lower bounds for the partitioning of graphs, *IBM J. Res. Dev.* **17**, (1973), pp. 420-425.

[Doyle and Snell 1984] P. G. Doyle, J. L. Snell, *Random Walks and Electric Networks* The Mathematical Association of America, Washington D. C., 1984.

[Drineas et al. 1999] P. Drineas, A. Frieze, R. Kannan, S. Vempala and V. Vinay, Clustering in large graphs and matrices, *Proceedings of the 10th annual ACM-SIAM symposium on discrete algorithms. Baltimore, MD, USA, January 17-19, 1999. Philadelphia*, (1999), 291-299.

[Dykstra 1983] R. L. Dykstra, An algorithm for restricted least squares regression, *J. Amer. Stat. Assoc.* **78**, (1983), pp. 837-842.

[Edmonds and Karp 1972] J. Edmonds and R. Karp, Theoretical improvements on algorithmic efficiency for network flow problems *J. Assoc. Computing Machinery* **19**, (1972), pp. 248-264.

[Elmore 1948] W.C. Elmore, The transient response of damped linear networks with particular regard to wide-band amplifiers, *Journal of Applied Physics* **19**, (1948), pp. 55-63.

[Ermoliev 1966] Y. M. Ermoliev, Methods for solving nonlinear extremal problems, *Kibernetika* **2/4**, (1966), pp. 1-17 (in Russian). Translated in *Cybernetics* **2/4**, (1966), pp. 1-14.

[Fisher 1981] M. L. Fisher, The Lagrangian relaxation method for solving integer programming problems, *J. Manage. Sci.* **27**, (1981), pp. 1-18.

[Fishburn and Dunlop 1985] J. P. Fishburn, A. E. Dunlop, TILOS: a posynomial programming approach to transistor sizing. *Proceedings of the IEEE International Conference on Computer-Aided Design 1985,* pp. 326-328.

[Fiedler 1973] M. Fiedler, Algebraic connectivity of graphs, *Czechoslovak Mathematical Journal* **23**, (1973), pp. 298-305.

[Frank 1993] A. Frank, Conservative Weightings and Ear-Decompositions of Graphs, *Combinatorica* **2**, (1993) pp. 247-274.

[Frank 1999] A. Frank, T. Jordán and Z. Szigeti, An Orientation Theorem with Parity Conditions, *Proceedings of 7th Conference on Integer Programming and Combinatorial Optimization, Graz 1999*, pp. 183-190.

[Frank and Király 1999] A. Frank and Z. Király, Parity Constrained k-Edge-Connected Orientations, *Proceedings of 7th Conference on Integer Programming and Combinatorial Optimization, Graz 1999*, pp. 191-201.

[Garey and Johnson 1977] M. R. Garey, D. S. Johnson, The Rectilinear Steiner Tree Problem is $NP-$Complete, *SIAM Journal on Applied Mathematics* **32**, (1977), pp. 826-834.

[Geelen 1998] J. Geelen, An Algebraic Matching Algorithm, *Technical Report, University of Waterloo* (1998).

[Geelen and Iwata 2003] J.F. Geelen, S. Iwata, *Matroid matching via mixed skew-symmetric matrices,* manuscript (2003).

[Godsil 1993] C.D.Godsil, *Algebraic Combinatorics*, Chapman and Hall, 1993.

[Goemans and Rendl, 1999] M.X.Goemans, F.Rendl, "Semidefinite Programming in Combinatorial Optimization," in: *H. Wolkowicz, R.Saigal and L. Vandenberghe (Eds.): Handbook of Semidefinite Programming: Theory, Algorithms and Applications,* Chapter 12, Kluwer Academic Publishers, 1999.

[Grady 2004] L. Grady, *Space-Variant Computer Vision: A Graph-Theoretic Approach* Ph.D. thesis, Boston University 2004.

[Guattery and Miller 2000] S. Guattery and G. L. Miller, Graph Embeddings and Laplacian Eigenvalues, *SIAM Journal on Matrix Analysis and Applications* Volume **21** Number **3**, (2000), pp. 703-723.

[Han 1988] S. P. Han: A succesive projection method, *Math. Programming* **40**, (1988), 1-14.

[Wolfe and Crowder 1974] M. , P. Wolfe and H. P. Crowder, Validation of subgradient optimization *J. Math. Program.* **6**, (1974), 62-88.

[Held 2001] S. Held, *Algorithmen für Potenzial-Balncierungs-Probleme und Anwendungen im VLSI-Design* (in German), Diploma Thesis, University of Bonn, 2001

[Held et al. 2003] S. Held, B. Korte, J. Maßberg, M. Ringe and J. Vygen, Clock Scheduling and Clocktree Construction for High Performance ASICs. *Proceedings of the IEEE International Conference on Computer-Aided Design 2003,* pp. 232-239.

[Hendrickson and Leland 1995] B. Hendrickson and R. Leland, An improved spectral graph partitioning algorithm for mapping parallel computations, *SIAM J. Sci. Comput.* **16** No.2, (1995), pp. 452-469.

[Hestenes and Stiefel 1952] M. R. Hestenes and E. Stiefel, Methods of Conjugate Gradients for Solving Linear Systems, *Journal of Research of the National Bureau of Standards* **49**, (1952), pp. 409-439

[Hurst, Cong and Kuehlmann 2004] A.P. Hurst, P.Cong and A. Kuehlmann, *Physical Placement Driven by Sequential Timing Analysis,* Manuscript.

[Ibaraki, Fukushima and Ibaraki 1991] S. Ibaraki, M. Fukushima and T. Ibaraki, Dual-based Newton methods for nonlinear minimum cost network flow problems, *J. Oper. Res. Soc. Japan* **34**, No.3 (1991), pp. 263-286.

[IEEE 1999] IEEE Delay and Power Calculation Working Group, *1481-1999 IEEE Standard for Integrated Circuit (IC) Delay and Power Calculation System* IEEE (1999).

[Jackson and Kuh 1989] M. A. B. Jackson and E. S. Kuh, Performance-Driven Placement of Cell Based IC's, *Proceedings of the 26th ACM/IEEE Design Automation Conference 1989,* pp. 307-375.

[Jaeger 1983a] F. Jaeger, Symmetric representations of binary matroids, *Ann. Discrete Math.* **17**, (1983), pp. 371-376.

[Jaeger 1983b] F. Jaeger, Graphes de cordes et espaces graphiques (in French), *Eur. J. Comb.* **4**, (1983) pp. 319-327.

[Kaveh and Bondarabady 2000] A. Kaveh and H. A. Rahimi Bondarabady, Finite element mesh decomposition using complementary Laplacian matrix. *Commun. Numer. Methods Eng.* **16** No.6, (2000), pp. 379-389.

[Kleinhans et al. 1991] J. M. Kleinhans, G. Sigl, F. M. Johannes and K. J. Antreich, GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization, *IEEE Transaction on Computer Aided Design of Integrated Circuits and Systems,* **10**, (1991), pp. 356-365

[Kourtev and Friedman 1999] I.S. Kourtev, E.G. Friedman, Clock Skew Scheduling for Improved Reliability via Quadratic Programming, *Proceedings of the ICCAD 1999*, pp. 219-223.

[Lang 1971] S. Lang, *Algebra* Addison Wesley, 1971.

[Langkau 2000] K. Langkau, *Gate-Sizing in VLSI-Design,* (in German), Diploma thesis, University of Bonn (2000).

[Larsson, Patriksson and Strömberg 1996] T. Larsson, M. Patriksson and A.-B. Strömberg, Conditional subgradient optimization - theory and applications, *European J. Oper. Res.* **88**, (1996), pp. 382-403.

[Lee et al. 2001] J. Lee, D.L. Ostapko, J. Soreff and C.K. Wong, On the signal bounding problem in timing analysis, *Proc. ICCAD 2001*, pp. 507-512.

[Li 1996] Y. Li, A Newton Accelaration of the Weiszfeld Algorithm for Minimizing the Sum of Euclidean Distances, *Technical Report, Cornell University* (1996).

[Lovász 1972] L. Lovász, A Note on Factor-Critical Graphs, *Studia Sci. Math. Hungar.* **7**, (1972), pp. 287-290.

[Lovász 1979] L. Lovász, On Determinants, Matchings and Random Algorithms, in *Fundamentals of Computing Theory* (L.Budach, Ed.), Akademia Verlag, Berlin, 1979.

[Lovász 1986] L. Lovász and M.D. Plummer, *Matching Theory,* Annals of Discrete Mathematics, 29. North-Holland Mathematics Studies, 121. Amsterdam 1986.

[Lovász 1993] L. Lovász, "Random Walks on Graphs: A Survey," in *D. Miklós, V. T. Sós and T. Szőnyi (Eds.): Combinatorics, Paul Erdős is Eighty (Volume* **2***)*, Bolyai Society, Budapest, 1993, pp. 353-397.

[Luo and Tseng 1992] Z.Q. Luo and P. Tseng, On the Convergence of the Coordinate Descent Method for Convex Differentiable Minimization, *J. Opt. Theory and Apps.* **72** No. 1, (1992), pp. 7-35.

[Love 1969] R.F. Love, Locating Facilities in Three Dimensional Space by Convex Programming, *Nav. Res. Log. Quart.* **16**, (1969), pp. 503-516.

[Lubotzky 1994] A. Lubotzky, *Discrete Groups, Expanding Graphs and Invariant Measures* Birkhäuser, Basel 1994.

[Maßberg 2002] J. Maßberg, *Clocktreedesign mit optimalen Ankunfszeitintervallen* (in German), Diploma thesis, University of Bonn, 2002.

[Marr and Hildreth 1980] D. Marr, E.C. Hildreth, Theory of edge detection, *Proceedings of the Royal Society of London,* **207**(1167), (1980), pp. 187-217.

[Meshaby and Hadaegh 2001] M. Meshaby and F.Y. Hadaegh, Formation flying control of multiple spacecraft via graph, matrix inequalities and switching, *AAIA Journal of Guidance, Control and Dynamics* **24**(2), (2001), pp. 369-377.

[Minoux 1984] M. Minoux, A polynomial algorithm for minimum quadratic cost flow problems, *Eur. J. Operational Res.* **18**, (1984), pp. 377-287.

[Minoux 1986] M. Minoux, *Mathematical Programming*, John Wiley & Sons, New York, 1986.

[Muuss 1999] Karsten Muuss, *Personal communication* (1999).

[Nakanishi 1971] N. Nakanishi, *Graph theory and Feynman integrals,* Mathematics and its Applications, Vol. **11.** New York-London-Paris: Gordon and Breach Science Publishers 1971.

[Polyak 1967] B. T. Polyak, A general method of solving extremum problems, *Doklady Akademii Nauk SSSR* **174/1**, (1967), 33-36 (in Russian). Translated in *Soviet Mathematics Doklady* **8/3**, (1967) pp. 593-597.

[Polyak 1969] B. T. Polyak, Minimization of unsmooth functionals, *USSR Computational Mathematics and Mathematical Physics* **9**, (1969), pp. 14-29.

[Pothen, Simon and Liou 1990] A. Pothen, H. D. Simon and K. P. Liou, Partitioning sparse matrices with eigenvectors of graphs, *SIAM Journal on Matrix Analysis and Applications*, **11** (1990), pp. 430-452.

[Ratzlaff and Pillage 1994] C.L. Ratzlaff and L. T. Pillage, RICE: Rapid interconnect circuit evaluation using AWE, in: *IEEE Trans. CAD*, June 1994, **13**(6), pp. 763-776.

[Rautenbach, Szegedy and Werber 2003a] D. Rautenbach, C. Szegedy and J. Werber, Asymptotically optimal boolean circuits for functions of the form $g_{n-1}(g_{n-2}(\dots g_2(g_1(x_1,x_2),x_3)\dots,x_{n-1}),x_n)$ given input arrival times, *Technical Report 03931*, University of Bonn (2003).

[Rautenbach, Szegedy and Werber 2003b] D. Rautenbach, C. Szegedy and J. Werber, Delay optimization of linear depth boolean circuits with prescribed input arrival times, *Technical Report 03932*, University of Bonn (2003).

[Rautenbach, Szegedy and Werber 2003c] D. Rautenbach, C. Szegedy and J. Werber, Fast circuits for functions whose inputs have specified arrival times, *Technical Report 03933*, University of Bonn (2003).

[Rockafellar 1997] R.T. Rockafellar, *Convex Analysis* Princeton University Press, 1997.

[Sapatnekar et al. 1993] S. S. Sapatnekar, V. B. Rao, P. M. Vaidya and S.-M. Kang, An Exact Solution to the Transistor Sizing Problem for CMOS Circuits Using Convex Optimization, *IEEE Trans. Computer-Aided Design*, Vol. 12, (1993) pp. 1621-1627.

[Schietke 1999] J. Schietke, *Timing-Optimierung beim physikalischen Layout von nicht-hierarchischen Design hochintegrierter Logikchips*, Ph.D. thesis, University of Bonn (1999).

[Schwartz 1980] J. Schwartz, Fast probabilistic algorithms for verification of polynomial identities, Journal of the ACM **27**, (1980), pp. 701-717.

[Sechen and Tennakoon 2002] C. Sechen and H. Tennakoon, Gate sizing using Lagrangian relaxation combined with a fast gradient-based pre-processing step, in: *Proceedings of the 2002 IEEE/ACM International Conference on Computer-aided Design*, pp. 395-402.

[Sigl, Doll and Johannes 1991] G. Sigl, K. Doll and F.M. Johannes, Analytical Placement: A linear or a quadratic Objective Function? *Proceedings of the 28th ACM/IEEE Design Automation Conference 1991*, pp. 427-432

[Slotine and Wang 2003] J.J.E. Slotine and W. Wang, A Study of Synchronizations and Group Cooperation Using Partial Contraction Theory, in: *Block Island Workshop on Cooperative Control,* Kumar V. Editor, Springer Verlag 2003.

[Srinisavan, Chaudary and Kuh 1992] A. Srinisavan, K. Chaudhary and E.S. Kuh, RITUAL: A performance-driven placement algorithm, *IEEE Transactions on Circuits and Systems* vol. **37**, (1992), pp. 825-839.

[Struzyna 2004] M. Struzyna, *Analytisches Placement im VSLI-Design* (in German) Diploma thesis, University of Bonn, 2004.

[Szegedy 1999] C. Szegedy, A Representation of the Ear-Matroid, *Technical Report 99878*, University of Bonn (1999)

[Szegedy and Szegedy 2004] B. Szegedy and C. Szegedy, Symplectic spaces and ear-decompositions of matroids, *Combinatorica*, (to appear)

[Szigeti 1994] Z. Szigeti, *Conservative Weightings of Graphs,* PhD Thesis, Eötvös Loránt University Budapest (1994).

[Szigeti 1996] Z. Szigeti, On a Matroid Defined by Ear-Decomposition of Graphs, *Combinatorica* **16,** (1996) pp. 233-241.

[Szigeti 2001] Z. Szigeti, On generalizations of matching-covered graphs, *Eur. J. Comb.* **22**, No.6, (2001), pp. 865-877.

[Tsay and Koehl 1991] R.-S. Tsay and J. Koehl, An Analytic Net Weighting Approach for Performance Optimization in Circuit Placement, *Proceedings of the 28th ACM/IEEE Design Automation Conference,* (1991), pp. 620-625.

[Tsay and Kuh 1991] R.-S. Tsay and E.Kuh, A Unified Approach to Partitioning and Placement, *IEEE Transaction on Circuits and Systems* **38** No.5, (1991), pp. 521-633.

[Tutte 1947] W. T. Tutte, The factorization of linear graphs. (English) *J. Lond. Math. Soc.* **22**, (1947), pp. 107-111.

[Vladimirescu 1994] A. Vladimirescu, *The SPICE Book,* John Wiley & Sons, New York, 1994.

[Vygen 1996] J. Vygen, *Plazierung in VLSI-Design und ein zweidimensionales Zerlegungsproblem* (in German), Ph.D. thesis, University of Bonn, 1996

[Vygen 1997] J. Vygen, Algorithms for large-scale flat placement. In *ACM/IEEE Design Automation Conf.* (1997) pp. 746-751.

[Vygen 2001] J. Vygen, *Theory of VLSI Layout,* Habilitation Thesis, University of Bonn (2001).

[Warme, Winter and Zachariassen 2000] D. M. Warme, P. Winter and M. Zachariassen, "Exact algorithms for plane Steiner tree problems: A computational study," *Advances in Steiner trees, D. Z. Du, J.M. Smith and J.H. Rubinstein (Eds.),* Kluwer, Boston, 2000, pp. 81-116.

[Weismantel 1992] R. Weismantel, Plazieren von Zellen: Theorie und Lösung eines quadratischen 0/1-Optimierungsproblem, (in German), Ph.D. thesis, TR 92-3, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1992.

[Weiszfeld 1937] E. Weiszfeld, Sur le point pour lequel la somme des distances de n points donnes est minimum. (in French), *Tohoku Math. J.* **43**, (1937), pp. 355-386.

[Welsh 1976] D.J.A. Welsh, *Matroid Theory* Academic Press, 1976.

[Wipfler, Wiesel and Mlynski 1983] G.J. Wipfler, M. Wiesel and D.A. Mlynski A Combined Force and Cut Algorithm for Hierarchical VLSI Layout, *Proc. 20th ACM/IEEE Design Automation Conference 1983,* pp. 124-125.

[Zippel 1979] R. Zippel, Probabilistic algorithms for sparse polynomials, In *International Symposium on Symbolic and Algebraic Computation*, Vol. 72 of *Lecture Notes in Computer Science*, (1979), Berlin, pp. 216-226.