Spectral Methods for Efficient Load Balancing Strategies

Robert Elsässer

Universität Paderborn
Institut für Informatik
Fürstenallee 11
33102 Paderborn
elsa@upb.de

Abstract: Die gleichmäßige Verteilung der auftretenden Rechenlast bei gleichzeitiger Minimierung der Kommunikation ist von entscheidender Bedeutung für die effiziente Auslastung der Ressourcen eines parallelen Systems. Eine große Zahl von Lastverteilungsverfahren wurde entwickelt und durch theoretische Analyse, durch Experimente und durch Integration in Anwendungen untersucht. Unter diesen Verfahren spielen die k-Partitionierungsverfahren und die Diffusionsverfahren eine prominente Rolle. Diese Arbeit stellt eine Zusammenfassung meiner Dissertation dar, und beschäftigt sich einerseits mit der Konstruktion und Analyse neuer spektraler Schranken für die k-Sektionsweite von Graphen und andererseits mit der Entwicklung verbesserter diffusionsbasierter Lastbalancierungsverfahren für verschiedene Graphklassen und Netzwerktopologien. Zudem wird gezeigt, dass die bekannten Diffusionsverfahren auf heterogene Netze übertragen werden können, wobei die Konvergenzgeschwindigkeit von der Konditionszahl einer zugehörigen gewichteteten Laplacematrix abhängt.

1 Einleitung

Eine der zentralen Problemstellungen bei der effizienten Nutzung paralleler Systeme ist die kostengünstige Abbildung der Rechen- und Kommunikationslast einer parallelen Berechnung auf das zugehörige Prozessornetzwerk. Ein Prozess kann allerdings nur dann effizient auf einem Parallelrechner bearbeitet werden, wenn der Prozess in viele Subprozesse unterteilt werden kann. Jeder Subprozess besitzt eine Rechenlast und es können bestimmte Abhängigkeiten zwischen diesen Subprozessen existieren. Der Prozess kann deshalb als ein sogenannter *Prozessgraph* dargestellt werden, wobei die Knoten dieses Graphen die einzelnen Subprozesse beschreiben und die Kanten des Graphen die Abhängigkeiten zwischen den zugehörigen Subprozessen modellieren.

Unser Ziel ist, die Gesamtberechnungszeit zu minimieren. Deshalb wollen wir zum einen die Gesamtlast im System gleichmäßig auf die Prozessoren verteilen und zum anderen die Kommunikation zwischen den Prozessoren minimieren. Mit anderen Worten, ist die Anzahl der Prozessoren im System k, dann sollte der Prozessgraph in k gleichgroße Teile zerlegt werden, so dass die Anzahl der Kanten zwischen den verschiedenen Teilgraphen minimiert wird. Dieses Problem ist unter dem Namen k-Partitionierungsproblem bekannt.

Eine Anwendungsklasse mit großer Bedeutung ist die der sich dynamisch verändernden, graphstrukturierten, datenparallelen Anwendungen. Probleme dieser Art treten in allen Bereichen der numerischen Simulation physikalischer Vorgänge auf, aber auch bei der Simulation dynamischer Systeme in der Mechatronik oder Produktionsplanung. In all diesen Anwendungen beschreiben *partielle Differentialgleichungen* das dynamische Verhalten eines physikalischen Vorgangs. Solche partielle Differentalgleichungen können in den meisten Fällen nicht analytisch gelöst werden. Deshalb transformiert man den kontinuerlichen Raum, der von dieser Differentalgleichung beschrieben wird, mit Hilfe einfacher geometrischer Objekte, wie Dreiecke oder Vierecke, in einen diskreten Raum. Am Ende dieser Transformation erhält man ein sogenanntes *Finite-Elemente Netz*, das ein lineares Gleichungssystem darstellt. Die Lösung der ursprünglichen partiellen Differentialgleichung läßt sich mit Hilfe des so gewonnenen Gleichungssystems approximieren.

Solche lineare Gleichungssysteme können effizient parallelisiert werden, indem man das Finite-Elemente Netz in der Anzahl der Prozessoren entsprechenden Teile zerlegt, um anschließend die einzelnen Teile auf die Prozessoren des Systems abzubilden. Dabei sollte man zwei Kriterien beachten: die gleichmäßige Aufteilung des Netzes und die Minimierung der Kommunikationsbeziehungen zwischen den einzelnen Prozessoren.

Einige der Methoden zur Lösung dieses Partitionierungsproblems modellieren zunächst das Finite-Elemente Netz als einen Graphen, in dem die Knoten die Elemente des Netzes beschreiben (in unserem Fall die Dreiecke oder Vierecke) und zwei Knoten genau dann über eine Kante verbunden werden, wenn die zugehörigen Netzelemente benachbart sind. Unser Ziel ist es, die Knotenmenge dieses Graphen in k gleichgroße Mengen zu zerlegen, so dass die Anzahl der Schnittkanten, die verschiedene Mengen voneinander trennen, minimiert wird. Diese minimale Anzahl der Schnittkaten einer balancierten k-Partitionierung heißt die k-*Sektionsweite* des Graphen.

Die oben beschriebene Vorgehensweise bei der Prozessorzuteilung entspricht einer statischen Methode. Falls bei der Applikation allerdings während der Laufzeit neue Prozesse generiert werden, oder die Prozesse ihre Kommunikationsanforderungen variieren, sind neue Plazierungsentscheidungen zu treffen.

Ein bekannter Ansatz besteht darin, zu diskreten Zeitpunkten auf die Veränderung des Prozessgraphen durch eine komplette Neu-Partitionierung desselbigen zu reagieren. Dies führt im allgemeinen zu einer kostenoptimalen Partitionierung. Demgegenüber steht allerdings ein hoher Aufwand zur Prozessmigration.

Als Alternative dazu läßt sich die Balance im System mit Hilfe der *dynamischen Lastbalancierung* wiederherstellen, die, aufbauend auf den bis dato getroffenen Platzierungsentscheidungen, versuchen, auf die neue Situation zu reagieren, indem eine unter gewissen Kostenkriterien zufriedenstellende Adaption der bisherigen Platzierung berechnet und durchgeführt wird.

In dieser Arbeit beschäftigen wir uns mit Lastbalancierungsalgorithmen, die in drei Phasen arbeiten. In der ersten Phase wird berechnet, *wieviel* Last über die Kanten des sogenannten *Quotient-Graphen* geschickt werden muss, oder anders formuliert, man berechnet einen balancierenden Fluss. Die Knoten des Quotient-Graphen stellen die einzelnen Teile des Finite-Elemente Netzes dar und zwei Knoten werden über eine Kante verbunden, wenn

die zugehörigen Teile des Netzes benachbart sind.

Der balancierende Fluss kann als die Lösung eines linearen Gleichungsystems angesehen werden [HBE98]. Wir verwenden in dieser Arbeit iterative Methoden zur Lösung eines solchen Gleichungssystems, die lokal auf dem Quotient-Graphen vorgehen, d.h. die Knoten des Graphen tauschen ihre Lastinformationen mit ihren Nachbarn iterativ aus, bis ein balancierter Fluss berechnet wird.

Sobald ein balancierender Fluss ermittelt ist, berechnet man in der zweiten Phase eine Scheduling der Lastelemente bzgl. des berechneten Flusses. Falls die aktuelle Last eines jeden Prozessors größer als die Gesamtlast ist, die er zu seinen Nachbarn zu schicken hat, kann die Balance in einem Schritt hergestellt werden und es wird keine Scheduling benötigt. In den meisten Fällen ist allerdings diese Situation nicht gegeben und eine Scheduling-Phase ist desahalb nötig.

In der dritten Phase werden einige Lastelemente ausgewählt und der Fluss wird verschickt. Bei der Auswahl der Elemente können weitere Optimierungskriterien beachtet werden, wie z.B. die Minimierung des Schnittes oder die Optimierung der Form der verbliebenen Subnetzwerke (siehe auch [DMM98, FMB95, JAMS89]).

In dieser Arbeit behandeln wir nur die erste Phase und beschreiben effiziente Algorithmen für die Berechnung eines balancierenden Flusses auf verschiedenen Graphklassen und Netzwerken.

2 Problembeschreibung und Ergebnisse

Dieses Kapitel besteht aus zwei Teilen. Der erste Teil beschäftigt sich mit dem Partitionierungsproblem von Graphen, sowie mit den von uns entwickelten neuen unteren Schranken für die k-Sektionsweite von Graphen. Im zweiten Teil dieses Kapitels behandeln wir die für die Lösung des Lastbalancierungsproblems entwickelten Diffusionsverfahren und beschreiben die von uns konstruierten Lastverteilungsalgorithmen.

2.1 Das k-Partitionierungsproblem

Die Lösung des k-Partitionierungsproblems ist schon für den Spezialfall der partitionierung in zwei Teilmengen (Bisektionsproblem) NP-vollständig [GJS76], und somit existieren keine effizienten Verfahren, die eine optimale Lösung für allgemeine Graphen berechnen. Für spezielle Graph-Klassen wurden eine Reihe von exakten Ergebnissen oder aber untere und obere Schranken für die Bisektionsweite ermittelt [MD97]. Die k-Sektionsweite hingegen ist sehr schwer zu berechnen, sogar für Graphen, für die die Bisektionsweite relativ einfach ermittelt werden kann.

Für die meisten Anwendungen wird eine balancierte k-Partitionierung durch den rekursiven Einsatz von effizienten Bisektionsheuristiken berechnet. Bei diesen Verfahren wird üblicherweise zwischen *globalen* und *lokalen* Heuristiken unterschieden. Die globalen

Methoden erhalten den Graphen als Input und berechnen eine balancierte Bisektionierung, während die lokalen Methoden neben dem Graphen eine balancierte Bisektionierung als Input erhalten und versuchen, diese bezüglich der Anzahl der Schnittkanten zu verbessern. Die leistungsfähigsten globalen Techniken sind *Inertial-*, *Spectral-*, und *Geometric-Partitioning* [DMP95, GGL93, HL95], während die effizientesten lokalen Methoden Varianten der *Kernighan-Lin* [KL70] und der *Helpful-Set* Heuristik sind [MP01].

Um die Qualität gegebener Verfahren zu beurteilen, ist es neben entsprechenden Tests auf Benchmark-Graphen insbesondere wichtig, Schranken für die k-Sektionsweite zu bestimmen. Diese Schranken können auch dazu verwendet werden, einige Branch & Bound Strategien zu verbessern, die dann eine gute k-Partitionierung für mittelgroße Graphen berechnen können.

Im weiteren werden wir die bekannten Verfahren zur Berechnung einer unteren Schranke für die k-Sektionsweite von Graphen vorstellen. Eine dieser Methoden besteht darin, eine Einbettung des volltändigen Graphen in einen gegebenen Graphen zu finden, wobei die Kantenauslastung über alle Einbettungen minimiert wird [Le92]. Diese Methode ist ursprünglich für die Berechnung einer Bisektionsschranke entwickelt worden, kann aber auf beliebige k-Partitionierungen verallgemeinert werden.

Man kann auch untere Schranken für die k-Sektionsweite von Graphen mit Hilfe von Methoden aus der algebraischen Graphentheorie entwickeln. Die klassische spektrale untere Schranke für die k-Sektionsweite ∇ eines Graphen $G = (V, E)$ mit n Knoten hat die Form

$$\nabla \geq \frac{n}{2k} \sum_{i=1}^{k} \lambda_i, \tag{1}$$

wobei $\lambda_1, \ldots, \lambda_k$ die k kleinsten Eigenwerte der Laplacematrix L des Graphen G sind [DH73].

Falkner, Rendl und Wolkowicz [FRW94] verwenden einen anderen Ansatz, um untere Schranken für die k-Sektionsweite eines Graphen zu berechnen. In diesem Ansatz wird das k-Partitionierungsproblem als ein semidefinites Programm beschrieben und man erhält dadurch die bekannte untere Schranke aus (1). In [RW95] werden dann Projektionsmethoden angewandt, um bessere Schranken zu berechnen.

Bolla [Bo93] berechnet neue untere spektrale Schranken für die k-Sektionsweite von Hypergraphen. Bolla und Tusnády betrachten spektrale Schrenken für die k-Sektion von gewichteten Graphen in [BT94], dabei wird allerdings keine balancierte Partitionierung vorausgesetzt.

Wir zeigen, dass die untere Schranke in (1) nur für eine sehr kleine Anzahl von Graphen scharf ist. In den meisten Fällen muss man mit einer sehr großen Lücke zwischen dieser Schranke und der k-Sektionsweite rechnen. Betrachten wir als Beispiel den $\sqrt{n} \times \sqrt{n}$-Torus. Die 4 kleinsten Eigenwerte dieses Graphen sind $\lambda_1 = 0$ und $\lambda_2 = \lambda_3 = \lambda_4 = 2 - 2\cos(\frac{2\pi}{\sqrt{n}}) \approx 4\pi^2/n$. Daraus ergibt sich für die untere Schranke der Wert $\frac{3}{2}\pi^2$. Die 4-Sektionsweite nimmt allerdings den Wert $4\sqrt{n}$ an. Es existiert demnach eine quadratische Lücke zwischen der unteren Schranke und der k-Sektionsweite des Graphen. Wir berechnen neue spektrale untere Schranken, die anstatt $\sum_{i=1}^{k} \lambda_i$ den Ausdruck $\sqrt{\sum_{i=1}^{k} \lambda_i}$

verwenden und schließen die vorher beschriebene quadratische Lücke bis auf einen konstanten Faktor. In diesem Zusammenhang sei erwähnt, dass für gradbeschränkte planare Graphen sowie für zweidimensionale gitterartige Netze $\lambda_2 = O(1/n)$ ist [ST96] und somit die quadratische Lücke bei diesen Graphen ebenfalls auftritt. Wenn man allerdings die neuen Schranken betrachtet, dann wird diese quadratische Lücke auch für solche Graphen geschlossen.

Wir berechnen neue Schranken für die Summe $\sum_{i=1}^{k} \lambda_i$, indem wir die *Level-Struktur* einer k-Sektion betrachten. Eine Schicht dieser Level-Struktur besteht aus den Knoten, die den gleichen Abstand von den Schnittkanten besitzen. Unter Verwendung dieser Level-Struktur kann man eine neue Schranke berechnen, die vom Wachstum dieser Schichten abhängt. Formal definiert, sei $g : \mathbb{N} \rightarrow \mathbb{N}$ eine Funktion. Die Klasse der Level-Strukturierten Graphen $LS(g, \nabla, k)$ besteht aus den Graphen, die eine k-Sektionsweite ∇ haben und die Anzahl der Kanten zwischen der Schicht j und der Schicht $j+1$ in einer Teilmenge der entprechenden k-Partitionierung höchstens $\nabla g(j)$ beträgt. Wir können mit Hilfe der Level-Strukturierten Graphen neue Zusammenhänge zwischen der k-Sektionsweite und der Summe $\sum_{i=1}^{k} \lambda_i$ herleiten. Falls $\sum_{j=2}^{\infty} \frac{1}{g(j-1)}$ beschränkt ist, dann zeigen wir, dass die klassische spektrale untere Schranke in (1) um einen konstanten Faktor verbessert werden kann. In den anderen Fällen, wenn $\frac{n}{\nabla} \rightarrow \infty$, dann gibt es eine Konstante δ so dass

$$\nabla \geq \delta \left(\sum_{i=1}^{k} \lambda_i \right)^{\beta} n(1 - o(1)), \tag{2}$$

wobei $\beta \in [1/2, 1)$. Wir zeigen auch, dass diese Schranken asymptotisch scharf sind.

Im allgemeinen ist die Level-Struktur einer k-Sektion nicht bekannt. Wenn es sich allerdings um einen Graphen $G = (V, E)$ handelt, der einen maximalen Knotengrad d besitzt, dann ist G ein Graph aus der Klasse $LS(g, \nabla, k)$, wobei $g(j) = (d-1)^j$. Falls $\frac{n}{\nabla} \rightarrow \infty$, dann gilt:

$$\nabla \geq \frac{k(d-2)+2}{k(d-2)} \cdot \frac{\sum_{i=1}^{k} \lambda_i \cdot n}{2k} (1 - o(1)). \tag{3}$$

Es existieren auch weitere Graphklassen, für die mit Hilfe der hier eingeführten Methode neue verbesserte spektrale Schranken berechnet werden können.

Es werden auch einige Graphklassen betrachtet, die nicht zu den Level-Strukturierten Graphen gehören. Diese Graphen lassen sich als die m-te kartesische Potenz eines dichten regulären Graphen darstellen. Wir zeigen, dass die k-Sektionsweite ∇ der m-ten Potenz eines d regulären Graphen $G = (V, E)$ mit $d = |V| - l - 1 \geq 3|V|/4$ der Ungleichung

$$\nabla \geq \frac{(|V| - l - 1)\lfloor \log k \rfloor}{2} |V|^m \tag{4}$$

unterliegt. Man kann beweisen, dass die untere Schranke aus (4) asymptotisch scharf ist.

2.2 Lokale Iterative Lastbalancierungsalgorithmen

Wie bereits erwähnt beschäftigen wir uns mit Lastbalancierungsmethoden, die in drei Phasen vorgehen. Dabei konzentrieren wir uns auf die erste Phase, in der ein balancierender Fluss berechnet wird. Sei $G = (V, E)$ ein Graph mit n Knoten und sei w_i die Last eines Knotens $i \in V$. Unsere Aufgabe besteht darin, einen balancierenden Fluss über die Kanten des Graphen zu berechnen, so dass nach der dritten Phase jeder Knoten i die Last $\overline{w_i} = \sum_{j=1}^{n} w_i/n$ besitzt. Wir setzen voraus, dass während der Iterationen keine Last generiert oder konsumiert wird und dass das Netz sich nicht verändert.

Es ist bekannt, dass das Lastbalancierungsproblem mit Hilfe eines linearen Gleichungssystems beschrieben werden kann [HBE98]. In der Literatur findet man viele Methoden, die erst den Vektor $w - \overline{w}$ berechnen, um anschließend das Gleichungssystem zu lösen (siehe [HB95]).

Wir setzen aber voraus, dass die Prozessoren eines parallelen Systems nur auf die Lastinformation ihrer Nachbarn zugreifen dürfen. Aus diesem Grund betrachten wir ausschließlich lokale iterative Algorithmen, die einen balancierenden Fluss berechnen. Es gibt zwei unterschiedliche Methoden, mit denen sich solche Algorithmen realisieren lassen: *Diffusion* und *Dimension-Exchange* [Cy89]. Diffusion setzt voraus, dass in jedem Schritt jeder Prozessor des Netzes die Last gleichzeitig mit all seinen Nachbarn balanciert, während bei Dimension-Exchange jeder Prozessor in einem Schritt die Last nur mit einem seiner Nachbarn balancieren kann.

Diffusionsschemata wurden von Cybenko [Cy89] eingeführt. Wenn w_i^k die Last des Knotens $i \in V$ eines Graphen $G = (V, E)$ nach k Iterationsschritten darstellt, dann gilt:

$$w_i^k = w_i^{k-1} - \sum_{\{i,j\} \in E} \alpha_{i,j}(w_i^{k-1} - w_j^{k-1}). \tag{5}$$

In den meisten Fällen wurde angenommen, dass für alle $(i, j) \in E$, $\alpha_{i,j}$ den selben Wert annimmt.

Es gibt viele Arbeiten, die den Zusammenhang zwischen der Konvergenzgeschwindigkeit von Diffusionsverfahren und der Konditionszahl der zugehörigen ungewichteten Laplacematrix analysieren (z.B. [DFM99]). Es ist bekannt, dass eine große Konditionszahl dieser Laplacematrix eine schnelle Konvergenz garantiert, während eine kleine Konditionszahl nur eine langsame Konvergenz zuläßt. In [DFM99] wird bewiesen, dass alle bekannten Diffusionsverfahren den selben Fluss berechnen und dass dieser Fluss minimal bzgl. der l_2-Norm ist. In der selben Arbeit wird zudem gezeigt, dass die Diffusionsverfahren für kanten-gewichtete Graphen verallgemeinert werden können, wobei der berechnete gewichtete Fluss l_2-minimal bleibt.

Es existiert hingegen eine sehr kleine Anzahl von Arbeiten, die inhomogene Schemata betrachten. Das Ziel dieser Schemata besteht darin, den Kanten eines Graphen bestimmte Gewichte zuzuweisen, so dass die Konditionszahl der zugehörigen Laplacematrix über alle Laplacematritzen mit der gleichen Adjazenzstruktur maximiert wird. In [DMN97] verwenden die Autoren ein semidefinites Programm und zeigen, dass mit Hilfe dieser Methode ein

polynomielles Approximationsschema für die Approximierung der optimalen Kantengewichte entwickelt werden kann. In dieser Arbeit werden zusätzlich noch mehrere Graphen mit optimalen Kantengewichten angegeben. Unter Verwendung dieser Methode kann man allerdings die optimalen Kantengewichte nur für kleine Graphen effizient approximieren.

Wir analysieren kantensymmetrische Graphen und zeigen, dass die Konditionszahl der Laplacematrix dieser Graphen genau dann maximiert wird, wenn jede Kante das gleiche Gewicht erhält. Dieses Ergebnis löst ein in [DMN97] beschriebenes offenes Problem. Wir betrachten auch Cayley Graphen und zeigen, dass die von dem selben Generator erzeugten Kanten das gleiche Gewicht haben müssen, damit die Konditionszahl der zugehörigen Laplacematrix maximiert wird. Für die kartesischen Produkten von Graphen verbessern wir unter Verwendung dieser Methode die Konvergenzgeschwindigkeit bekannter Diffusionsverfahren, indem wir den Kanten dieser Graphen bestimmte Gewichte zuweisen. Zugleich berechnen wir optimale Kantengewichte für Cube-Connected Cycles und Tori und verbessern die Laufzeit der Diffusionsschemata für Butterfly, De Bruijn und Gitter.

Wir beschäftigen uns auch mit heterogenen Netzwerken, die aus Prozessoren mit verschiedener Leistungsfähigkeit oder mit verschiedener Speicherkapazität bestehen. In einem heterogenen Netzwerk sollte man die Last nicht gleichmäßig verteilen, sondern proportional zu den Leistungsfähigkeit der Prozessoren. Langsame Prozessoren sollten weniger Last erhalten, schnellere hingegen mehr Last, damit man sicherstellt, dass jeder Prozessor die gleiche Zeit für die Abarbeitung seiner Last aufwendet. Formal beschrieben, sei n die Anzahl der Knoten in einem Graphen $G = (V, E)$, wobei jeder Knoten i ein Gewicht c_i besitzt. Unsere Aufgabe besteht darin, einen balancierenden Fluss über die Kanten des Graphen zu berechnen, so dass hinterher der Knoten i die Last

$$\overline{w_i} = \frac{\sum_{j=1}^{n} w_j}{\sum_{j=1}^{n} c_j} c_i \tag{6}$$

erhält. In diesem Fall benötigt jeder Knoten die gleiche Zeit, um seine Last abzuarbeiten und Idle-Zeiten werden dadurch vermieden. Dieses Problem stellt eine Verallgemeinerung des Lastbalancierungsproblems auf homogenen Netzwerken dar, wobei in homogenen Netzen jeder Knoten das Gewicht $c_i = 1$ besitzt. Es wird gezeigt, dass alle bekannten Diffusionsverfahren auf heterogene Netze übertragen werden können, wobei die Konvergenzgeschwindigkeit von der Konditionszahl einer zugehörigen gewichteten Laplacematrix abhängt. Wir setzen allerdings auch in diesem Fall voraus, dass die Topologie fest ist und dass während den Iterationsschritten keine Last generiert oder konsumiert wird.

Diese Arbeit beschäftigt sich außerdem mit einem neuen Lastbalancierungsstrategie für kartesischen Produkte von Graphen. Beispiele für solche Graphen sind $n \times n$ Gitter (das kartesische Produkt zweier Pfade der Länge n) oder $n \times n$ Tori (das kartesische Produkt zweier Kreise der Länge n). Dieses Verfahren trägt den Namen *Alternating-Direction Verfahren* und ist eine Kombination zwischen Diffusion und Dimension-Exchange. In diesem Modell, balanciert in einem Schritt jeder Prozessor seine Last mit seinen Nachbarn entlang der ersten Komponente und dann anschließend entlang der zweiten Komponente des Produkts. Unter Verwendung dieser Methode verbessert man die Laufzeit der einfachen Diffusionsschemata um Faktor 2. Der vom Alternating-Direction Verfahren berechnete Fluss ist im Gegensatz zu den bekannten Diffusionsverfahren nicht l_2-minimal und kann

sogar unbegränzt groß ausfallen. Ein solches Problem läßt sich vermeiden, indem man die Komponenten während der Iterationsschritte alternierend verwendet. Dieses modifizierte Schema trägt den Namen *Mixed Direction Iterative Scheme*.

In [DFM99] wurde ein optimales Verfahren eingeführt, das in $m-1$ Iterationsschritten die Last im System komplett ausgleicht, wobei m die Anzahl der unterschiedlichen Eigenwerte der Laplacematrix des Graphen darstellt. Der von diesem Schema berechnete Fluss ist l_2-minimal. Dieses Verfahren wurde so entwickelt, dass die Lastunterschiede im System kontinuerlich fallen und dass auf die numerische stabilität geachtet wird. Wir stellen ein einfacheres optimales Verfahren vor, das auch nur $m-1$ Schritte benötigt, um die Last komplett zu balancieren. Dieses Schema läßt sich wesentlich einfacher realisieren als das vorher beschriebene Verfahren, ist aber numerisch deutlich instabiler.

Ein solches Schema läßt sich vor allem in Systemen einsetzen, in dem Kommunikation zwischen beliebigen Paaren von Prozessoren möglich ist. Damit allerdings hohe Kommunikationskosten vermieden werden, erlauben wir für jeden Prozessor mit nur einer begrenzten Anzahl von anderen Prozessoren zu kommunizieren. Mit anderen Worten, wir verwenden eine sogenannte virtuelle Topologie, um die Kommunikationsstruktur im Netzwerk festzulegen. Auf dieser virtuellen Topologie muss zusätzlich noch Lastbalancierung schnell zu realisieren sein. In diesem Zusammenhang entwickeln wir Netzwerktopologien, die einen kleinen Knotengrad aufweisen und eine relativ geringe Anzahl von unterschiedlichen Eigenwerten besitzen. Mit Hilfe dieser Topologie erreicht man beide Ziele: Zum einen läßt sich die Kommunikation im Netzwerk auf das Nötige begrenzen und zum anderen kann man mit Hilfe des vorher beschriebenen optimalen Verfahrens die Last schnell ausgleichen.

3 Zusammenfassung und Ausblick

Wir haben neue Zusammenhänge zwischen den spektralen und strukturellen Eigenschaften von Graphen entdeckt und mit Hilfe dieser Zusammenhänge konnten wir neue spektrale Schranken für die k-Sektionsweite von Graphen entwickeln. Wir haben desweiteren asymptotisch optimale Schranken für die k-Sektionsweite von kartesischen Produkten dichter regulärer Graphen berechnet.

Auf dem Gebiet der Lastbalancierungsalgorithmen haben wir zunächst verbesserte Verfahren auf kartesischen Produkten von Graphen untersucht. Wir haben gezeigt, dass unter Anwendung der sogenannten Alternating-Direction Verfahren auf solchen Graphen die üblichen Diffusionsverfahren um Faktor 2 verbessert werden können. Allerdings ist der resultierende Fluss nicht mehr l_2-optimal, wie dies für Diffusionsverfahren gilt.

Danach haben wir uns mit der Konstruktion optimaler Diffusionsmatritzen zu gegebenen Graphen beschäftigt. Dabei heißt eine Diffusionsmatrix optimal für G, wenn das durch sie definierte Diffusions-Verfahren unter allen Matritzen mit der Verbindungsstruktur von G am schnellsten konvergiert. Es wurde gezeigt, dass in einer optimalen Diffusionsmatrix kantensymetrischer Graphen alle Kantengewichte gleich sind und dass in Cayley-Graphen alle Kanten, die durch denselben Generator erzeugt werden, das gleiche Gewicht haben.

Darüber hinaus wurden die optimalen Gewichte für Cube-Connected-Cycles berechnet.

Zum Schluss haben wir gezeigt, dass Diffusionsverfahren auf heterogene Netze übertragen werden können. Die Konvergenzgeschwindigkeit wird durch die Konditionszahl der gewichteten Laplacematrix bestimmt und auch die l_2-Optimalität und ein entsprechendes Verhältnis zwischen zweitem Eigenwert und Kantenexpansion gilt weiter.

In dieser Arbeit haben wir vorausgesetzt, dass die Topologie fest ist, und dass die Berechnungen synchron laufen. Wir können einfache Diffusionsschemata auch in dynamischen Netzen anwenden, die Konvergenz wird allerdings nur dann garantiert, wenn die Dynamik in der Topologie stark begrenzt wird. Es wäre aber sicherlich interessant zu analysieren, in wieweit diese Einschränkung aufgeweicht werden darf. Eine weitere wichtige Frage ist, ob die effizienteren Diffusionsverfahren in dynamischen Netzen ebenfalls angewandt werden können.

Literatur

[Bo93] Bolla, M.: Spectra, euclidean representations and clusterings of hypergraphs. *Discrete Mathematics*. 117:19–40. 1993.

[BT94] Bolla, M. und Tusnády, G.: Spectra and optimal partitions of weighted graphs. *Discrete Mathematics*. 128:1–20. 1994.

[Cy89] Cybenko, G.: Load balancing for distributed memory multiprocessors. *Journal of Parallel and Distributed Computing*. 7:279–301. 1989.

[DFM99] Diekmann, R., Frommer, A., und Monien, B.: Efficient schemes for nearest neighbor load balancing. *Parallel Computing*. 25(7):789–812. 1999.

[DH73] Donath, W. und Hoffman, A.: Lower bounds for the partitioning of graphs. *IBM J. Res. Develop.* 17:420–425. 1973.

[DMM98] Diekmann, R., Meyer, D., und Monien, B.: Parallel decomposition of unstructured FEM-meshes. *Concurrency: Practice and Experience*. 10(1):53–72. 1998.

[DMN97] Diekmann, R., Muthukrishnan, S., und Nayakkankuppam, M.: Engineering diffusive load balancing algorithms using experiments. In: *Solving Irregulary Structured Problems in Parallel, (IRREGULAR)*. LNCS 1253. S. 111–122. 1997.

[DMP95] Diekmann, R., Monien, B., und Preis, R.: Using helpful sets to improve graph bisections. In: Hsu, D., Rosenberg, A., und Sotteau, D. (Hrsg.), *Interconnection Networks and Mapping and Scheduling Parallel Computations.* volume 21 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. S. 57–73. AMS. 1995.

[FMB95] Farhat, C., Maman, N., und Brown, G.: Mesh partitioning for implicit computations via iterative domain decomposition. *Int. J. Num. Meth. Engrg.* 38:989–1000. 1995.

[FRW94] Falkner, J., Rendl, F., und Wolkowicz, H.: A computational study of graph partitioning. *Mathematical Programming*. 66:211–239. 1994.

[GGL93] George, A., Gilbert, J. R., und Liu, J. W. H.: Graph theory and sparse matrix computations. *The IMA Volumes in Math. and its Appl.* 56. 1993.

[GJS76] Garey, M., Johnson, D., und Stockmeyer, L.: Some simplified NP-complete graph problems. *Theoretical Computer Science*. 1:237–267. 1976.

[HB95] Hu, Y. und Blake, R.: An optimal dynamic load balancing algorithm. Technical Report DL-P-95-011. Daresbury Lab., UK. 1995.

[HBE98] Hu, Y., Blake, R., und Emerson, D.: An optimal migration algorithm for dynamic load balancing. *Concurrency: Practice & Experience*. 10(6):467–483. 1998.

[HL95] Hendrickson, B. und Leland, R.: An improved spectral graph partitioning algorithm for mapping parallel computations. *SIAM J. Sci. Comput.* 16(2):452–469. 1995.

[JAMS89] Johnson, D., Aragon, C., McGeoch, L., und Schevon, C.: Optimization by simulated annealing: An experimental evaluation; part 1, graph partitioning. *Operations Research*. 37(6):865–893. 1989.

[KL70] Kernighan, B. und Lin, S.: An effective heuristic procedure for partitioning graphs. *The Bell Systems Technical J.* S. 291–307. 1970.

[Le92] Leighton, F.: *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann Publishers. 1992.

[MD97] Monien, B. und Diekmann, R.: A local graph partitioning heuristic meeting bisection bounds. In: *8th SIAM Conf. on Parallel Processing for Scientific Computing*. 1997.

[MP01] Monien, B. und Preis, R.: Bisection width of 3- and 4-regular graphs. In: *Mathematical Foundations of Computer Science, (MFCS), LNCS 2136*. S. 524–536. 2001.

[RW95] Rendl, F. und Wolkowicz, H.: A projection technique for partitioning the nodes of a graph. *Annals of Operations Research*. 58:155–179. 1995.

[ST96] Spielman, D. A. und Teng, S.-H.: Spectral partitioning works: Planar graphs and finite element meshes. In: *Proc. of the 37th Conf. on Foundations of Computer Science, (FOCS)*. S. 96–105. 1996.

Robert Elsässer wurde 1972 in Neumarkt, Siebenbürgen geboren. Nach seinem Informatikstudium an der Universität Gesamthochschule Paderborn hat er die Abschlussprüfung 1998 mit der Note “sehr gut” bestanden. Anschließend war er in der Arbeitsgruppe Monien tätig und promovierte 2002 mit Auszeichnung an der Universität Paderborn. Ab Oktober 2002 ist er wisseschaftlicher Assistent. Er nimmt, sobald es die gesetzlichen Bestimmungen erlauben, seinen Ruf zum Juniorprofessor im Institut für Informatik an der Universität Paderborn wahr.