科学研究費補助金 特定領域研究

新世代の計算限界
— その解明と打破 —

平成17年度 成果報告書

平成18年 4月

領域代表者　岩間 一雄
京都大学 大学院 情報学研究科　教授

目 次

3.　各種資料

特定領域研究

「新世代の計算限界」ニュースレター　第 3 号　2005/7/11

はじめに

このニュースレターは，特定領域・新世代の計算限界のメンバーの情報交換と交流を目的とした情報発信誌です．毎回，いくつかの研究関連の記事と，特定領域のスケジュール・活動報告と，各研究者の活動予定などをお送りいたします．今回は，東北大学徳山豪先生の経験談と，6月に行われた全体会議，幹事会の報告を行います．

「群れ派」の研究の仕方　徳山 豪（東北大学）

最近は「引きこもり」や「ネットが友達」など一人でいるのが居るのがいいという方も結構いるらしいが，私は典型的な「群れ派」である．特定研究のおかげでいろいろな人の考えが聞けて，おしゃべりが出来るのは楽しくて仕方が無い．そういう場では少々躁状態になるので，声がでかくて「うざったい」（特に岩間さんや西関先生と一緒だと）が，暖かくお許しいただけるとありがたい．大体が教授室に1時間とじっとしていられないで学生研究室をうろうろしているので，学生には（だれも口に出して言わないが）うっとうしい存在である．そんな性格なので，一人でコツコツ本や論文を読んで勉強するのは苦手で，常に人との関りが幸運をもたらしている．宇野さんのリクエストもあり，そんな話（記憶が薄れているので，意図せずフィクションが入ったらごめんなさい）をしてみる．

食事とドライブ

楽しいおしゃべりの相手というと，浅野哲夫先生（二人の取り決めにより，「浅野さん」と書く）がなんといっても第一である．1988年にはじめての海外出張でイリノイ大学でのACMのSCG（計算幾何学シンポジウム）に行ったのがお付き合いのはじまり．日本でもお会いしたことはあったのだが，当時私は純粋数学（代数的群論）から計算機科学に転向して1年経たないほどの新米で，大先生相手に気後れして直接お話をしたことがなかった．
イリノイ大学のあるアーバナ・シャンペーンというところは平らな大地以外に何にも無いところで，しかも会議参加者は見知らぬ人ばかり．大体が食事をする場所も近くにない．うろうろしていると，うしろから「こんにちは」の声がして，黒いサングラスを掛けたヒゲの先生がニコニコしている．「タご

飯，どうします？　みんなと一緒に行きません？　僕，レンタカーしてるから．」うれしかったなあ．なんといっても人間は食べることが一番．　おまけに，後でIBMのWatson研究所でホストをしてくれたAlok Aggarwalや，研究での転換のきっかけをくれたMicha Sharirら有名人たくさんに紹介してもらった．結局会議期間中ずっと浅野さんの車に同乗したのだが，気に入ってもらえたらしく，「せっかくだから，一緒に研究しましょう．2次元のハッシュ問題だけど，興味ある？」この研究は後で「トポロジカルウォーク」というアイデアを生み，私の出世作（1991SCG）になったのだが，浅野さんとの出会いが無ければ今の半分も研究人生が楽しくなかっただろうと思う．

話のついでだが，この会議で，「ちょっといいか」といって，話しかけてきた二十歳くらいのパデュー大学の中国人学生が居た．テーブルで無理やり向かい合わせたかと思うと，いきなり「計算幾何学の並列アルゴリズムに将来性はあると思うか，返答は如何に」と質問してくる．なんだなんだと思いながら適当に答えていると，「Thank you，参考になった」といって握手してバイバイである．この生意気な学生はDanny Chen．当時はまだ論文も無い学生だったが，今ではトップクラスの研究者で，貴重な共同研究者である．

講演依頼は喜んで受けよう

STOC-FOCS-SODAとよく言うが，こういった会議に最初に論文を通すのが難しい．私の場合は，山下雅史先生と渡辺治先生のおかげである．といっても共同研究ではなく，駆け出しの研究者だった私に，「ランダムアルゴリズムについての一時間講演をお願い．絶対いい講演ができるから．」とおだてながら迫ったのがこのお二人（詳細は忘れたが，IEICE関係で，聴衆はずいぶんたくさん居た）．　当時ランダムアルゴリズムが世界的にブレイクし，反面日本ではあまり研究者が居なかったことは事実だが，ともかく自分でランダムアルゴリズムを設計したことが無い人間が「良い講演」をするのは難しいし，オリジナルな結果がまったく無いチュートリアルをするのは恥ずかしいと思った．講演期日までの間，柄にもなく一生懸命にさせられた結果が，割り当てや輸送問題を計算幾何学のランダムアルゴリズム手法で解く（問題自体は研究中だったが，ランダムというのは考えても居なかった）というもの．幸運なことにいい問題に当たった（1991SCG,1992SODA）が，講演を断っていたら運はめぐってこなかったろう．

人の話に首を突っ込むと

誰かが面白そうな（研究の）話をしているときに，話に加わりたいが邪魔をすると悪いなと思うかもしれない．IBMの東京基礎研究所で入社4年目くらいの頃，当時の上司の岩野さんと加藤直樹先生（これも加藤さんと呼ぶ）は頻繁に共同研究をしていた．ちょうど私の研究も軌道に乗って，翌年からアメリカ赴任という状況だったが，加藤さんが岩野さんを訪れて来た．ご挨拶でもしようと二人の話している会議室に（呼ばれもしないのに）行くと，なにやら面白げな話をしている．ぐずぐずしている間に，「徳山さんは専門家だから，聞いてよ」というのでありがたく参加すると，幾何学的な動的最小木とパラメトリックなマトロイドについての加藤さんの知識と発見の話題である．O(n^4)がマトロイドの理論で O(n^3) になるらしい．その場で考えてみると，計算幾何学のテクニックでもO(n^3) になるし，両方を組み合わせると O(n^2) にも出来るようである．わずか1時間ほどのDiscussionで出来た論文が，「動く点集合の最小木について」（1992FOCS）．さらに，関連するテーマで，たくさん（1995SODA,1999SODA,1999FOCS,2001SCG）論文を書いた．結論として，人の話には遠慮なく首を突っ込むべきである．

朝目覚めると

「ワーム食べませんか？」玉木さんと最初に会ったときに彼が言った言葉である．1992のFOCSで，ワームホールラウティングの講演をした直後，講演に使った（したがってOHP のガラスの上をのたくりまわった）グミキャンディーのミミズのおもちゃを半分にちぎって，片方を銜えながら．こん

な面白い人がIBMの研究所にやってきて，2年間同室だったのだから，いい研究ができないわけがない．まず手始めにということで，Micha Sharirと学会のlunchで一緒したときに「解くべきなのだが，まったく手掛りがない」といわれて挑戦し続けていた「放物線のレベル問題」について玉木さんに話すと，Lovaszのマッチングとカバーリングの双対定理という道具を使って，あららといううちに解けてしまったのはおどろいた（1995SCG）．私一人では3年間うまくいかなかったのに．　また或る時，国際会議で同室して，時差のせいで朝（？）3時に目覚めると，玉木さんなにやら思索にふけっている．「目が覚めましたか，ちょっと一緒に考えませんか？」．こちらは目覚めただけで起きたわけではないのだが，一緒につきあう．これで出来たのが1997年のISAACで最優秀論文賞をもらった研究（なんだか，私はサンチョ・パンザみたい）．

さて，玉木さんと私ではタイプはだいぶ違う．共通の趣味である囲碁を打つと良くわかるのだが，私は直感派で諦めが良く，玉木さんは論理（読み）を積み重ねないと納得しない．囲碁は個人ゲームだが，研究は協力して出来るので，違うタイプが重なると相乗効果で大変よろしい．考えてみると，「浅野－加藤－玉木－徳山」は4人ともタイプはそれぞれ違うように思う（どう違うかは，皆さんで判断してください）．浅野さんがリーダーシップを取って，この4人で合宿をしたりして，随分楽しく共同研究をしてきたものである．

研究のエピソードは書き出すと尽きないのだが，残りはまたの機会にしようと思う．

第2回全体会議を振り返って　浅野 哲夫 (JAIST)

去る6月16日から2日間に亘って開催された第2回の全体会議について報告します．昨年，特定研究がスタートして以来，平成16年10月に東北大学で開かれた第1回の全体会議，今年の2月に京都で開かれた国際会議に引き続いての会合ですが，最初の2回についてはまだ成果を出せる状況にないということもあって，招待講演を主とする構成になっていました．代表幹事の伊藤先生から，東京電機大学の築地先生と一緒に次回の全体会議を計画するように言われたとき，どのようなコンセプトにするか色々と考えました．まず考えたことは，特定研究にとって最も重要なことは何かでした．とにかく偉大な成果を出すことが要求されている訳ですが，特定研究としての成果というのは，単独では難しかった問題を特定研究のグループの力を借りて初めて解くことに成功したというものであることが望ましい訳です．つまり，うまくコラボレーションができたかどうかが問われる訳です．もちろん，理論のことですから，何度も集まったから必ず成果が出るという保証は何もありませんが，社会に対しては，少なくともコラボレーションがうまく行くように最大限の努力はしたということが見えなければならないと考えた訳です．では，そのためにはどうすればいいかですが，コラボレーションを成功させるためには，まず我々のグループの中でどんな成果があるのか，誰がどんな研究を行っているのかを知ることが肝腎です．ということで，まずは全ての研究班に自分たちの研究を紹介してもらおうという考えに至りました．もちろん，全員に話をしてもらうのが最適ですが，そうすると時間が2日では全く足りないので，ポスターの形式にしました．コラボレーションの切っ掛けは，日常会話に近い，ちょっとした会話の中から生まれるものだと個人的に信じているので（とにかく，興味ある共通の話題がないことにはやる気が起きないので），とにかくインフォーマルな会話の時間を大切にしたかったのです．

もう一つの視点は，私がこのコミュニティーが好きなのは，日本の社会にしては珍しく年齢の壁が高くないことです．理論をやっていると，どうしても若い人の集中力には負けるから，経験が豊富だというだけで威張ることができません．これが年齢を余り感じることなく誰とでも話せる雰囲気を醸し出しているのだと思います．いつの間にか自分も年寄りの部類に入ってしまっているのでしょう

が，気分はまだ20代です（言い過ぎです，ごめんなさい）．この雰囲気を大事にしたくて，偉い人にだけ講演をお願いするというスタイルを取りませんでした．

最後に，４人の方にだけ普段より長めの時間を用意して講演をしてもらいました．いずれも活躍が伝わってくる講演で，改めて本グループの質の高さを実感しました．今回の４人の講演者は私の一存で決めましたが，４人のご講演を聞いただけでも本グループの全体像が浮かび上がる，うまい人選になっていたのではないかと自画自賛しています．もちろん，講演を快諾して頂いた４人の講師の講演の質の高さが成功に導いたことは言うまでもありませんが．次回はどんな趣向で全体会議が企画されるか今から楽しみです．

2005年度第1回幹事会議事録　　事務局

日時：2005.6.16 12:00--13:50

場所：国立情報学研究所１２階会議室

[報告事項]

○ 教科書シリーズの進行状況（杉原，山下，室田，渡辺）

（仮題）アルゴリズムサイエンスシリーズ　全16巻　A5版 約200ページ

○ 電子ジャーナル（徳山）

・ECCCの成功（ドイツの大学）を見習ってECCCのアルゴリズム版を作る？

・維持が大変（お金と人手）

・本屋を巻き込めば継続可能だが

–> 全体討論で議論

○ 成果報告書（堀山）

・印刷体のものを各研究代表者に送る

○ 確率シンポジウム（徳山）

・資金は基本的に折半

○ ニューズレター（宇野・牧野）

・7月出版を目安に準備

[検討事項]

○NHC国際会議に続いて，今年度も国際会議を開催するか？

・若手の育成を重視し，サマースクール（またはスプリングスクール）を実施

・前半スクール後半チュートリアル？

–> 全体討論で議論

〇次回の全体会議の場所，担当者

・名古屋大，平田先生

・11／21，22予定

〇林先生のコメント

・審査が厳しくなっている．

– きちんと評価ー＞反映　という傾向．

– 中間審査の結果で減額されたり，最終報告の結果で次期プロジェクトに反映されたりするようになってくる．

・この特定は情報基盤を支える重要なものである．

・現実的な解法，具体的な成果が求められる．

・領域の連携が重要：単なる集まりでは無い（連携することでどんな結果が得られたか？目標に対してどこまで到達したか？）

・「情報学」が終わるのでこの特定が情報系の代表格になる

・グランドチャレンジを期待する．

「新世代の計算限界」イベントカレンダー　＋　事務連絡

特定領域研究も2年目に入り，段々と盛り上がってきて事務局も嬉しい忙しさです．春学校，ミニプロジェクトなど新しい試みも始まります．皆様，これからもご協力の程よろしくお願いします．

★ 報告書の作成について皆様のご協力有難うございました．作成方法やフォーマットなどについてご意見をお持ちでしたら，伊藤，堀山までお寄せ下さい．

★ ミニ研究集会はどなたがどんな思いつきで企画されても結構ですから，どしどし行って下さい．総括班から費用のサポートもいたします．

全体会議 （予定）　11/21(月),22(火) 名古屋大学

列挙合宿 （宇野毅明・中野眞一・上原隆平）

3日間，泊り込みで列挙・数え上げを中心に組み合わせアルゴリズムに関する雑談をします．

9/27(火)13:00 – 29(木)11:30 群馬県伊香保町 群馬大学セミナーハウス

7/11(月)-15(金) ICALP, ポルトガル・リスボン
7/11(月)-15(金) IFORS, ハワイ （参加者: 伊藤大雄, 巳波）
7/18(月)-21(木) ランダムネスと計算に関わるワークショップ 仙台国際センター (参加者: 元木光雄)
7/19(火)-22(金) Combinatorial Pattern Matching　韓国済州島
7/25(月)-27(水) LAシンポジウム　福岡県宗像市
8/15(月)-17(水) WADS, カナダ・ウォータールー
8/16(火)-29(金) COCOON, 中国・昆明
8/29(月)-9/2(金) MFCS, ポーランド・グダニスス
10/1(土)-5(水) CP2005　スペイン・バルセロナ　（参加者: 元木光雄）
10/3(月)-6(木) ESA, スペイン・イビザ
10/21(金)-10/22(土) 第17回RAMPシンポジウム 青森県弘前市 シティ弘前ホテル
10/23(日)-25(火) FOCS 2005 アメリカ・ピッツバーグ
12/7(水)-12/9(金) I-SPAN2005 アメリカ・ラスベガス
12/19(月)-21(水) ISAAC, 中国・海南島
1/22(日)-1/24(火) SODA2006　アメリカ・マイアミ

このニュースレターについて

ニュースレター各号は電子メールで配布する予定です．短い記事や連絡事項は全て掲載しますが，長い記事，イベントの詳細などはwebページに掲載する予定です．webページには詳細まで全てを載せた完全版を掲載して，目次，あるいは各記事の末尾のURLを参照すると，web版の同じ記事を参照できるようにいたします．

記事は，各回，1つの研究課題に担当をお願いする予定です．各研究課題で2000-4000字程度，研究に関わる記事を書いていただければと思います．通常，このようなニュースレターでは，研究成果を報告するのが一般的だと思われますが，この特定領域では「研究者の交流」に焦点を当てたいため，「研究の成果以外」の記事を面白く解説していただければと思います．例えば，最近参加した国際会議の情報を，どのようなものが流行っていたか，何が面白かったか，などの主観的な解説を交えて報告したり，最近考えている問題，あるいはオープン問題を，この辺までは解けるがここがうまくいかない，といった解説を交えて紹介する，という形です．

また，研究者間の交流を促進するため，各研究者の，国内外の会議への出席予定を集約して掲載していこうと考えています．研究者の交流には，顔をあわせる回数を増やすことが肝要です．他の研究者の参加予定がわかれば，会議への出席のモチベーションを高めることにもつながり，それがディスカッションや研究成果を生むきっかけにもなるでしょう．特定領域メンバーの皆さんには，自分のわかる範囲で，国内外の会議・研究会の情報と，自分の参加予定を教えていただければと思います．

この他，個人からの寄稿を募集いたします．100-1000字程度で，情報宣伝されたいことを自由な形式で書いて送っていただければ，掲載いたします．メールで配布する関係上，テキスト形式のものしか扱えませんが，そこはご了解お願いいたします．

次回は10月ごろを予定しています．

★ ニュースレター編集委員では，皆様からのご意見をお待ちしております．編集方針や内容の追加など編集全体にかかわることから細かいことまで，幅広いご意見をお願いいたします．

■■ 新世代の計算限界 ニュースレター ■■

編集委員長 宇野 毅明 uno@nii.jp（問合せ先）

副編集委員長 牧野 和久 makino@sflab.sys.es.osaka-u.ac.jp

特定領域研究

「新世代の計算限界」ニュースレター　　第 4 号　　2006/1/27

はじめに

このニュースレターは，特定領域・新世代の計算限界のメンバーの情報交換と交流を目的とした情報発信誌です．毎回，いくつかの研究関連の記事と，特定領域のスケジュール・活動報告と，各研究者の活動予定などをお送りいたします．今回は，名古屋大学高木直史先生の研究科紹介と，11月に行われた全体会議，幹事会の報告を行います．

中心？隙間？それとも辺境？　　高木　直史（名古屋大学）

■ 集積システム論

主要大学の中では少し遅れをとりましたが、名古屋大学でも2003年４月に情報系の大学院研究科として、情報科学研究科が設立されました。計算機数理科学専攻、情報システム学専攻、メディア科学専攻、複雑系科学専攻、社会システム情報学専攻の５つの専攻からなっています。本特定領域研究には、計算機数理科学専攻の平田先生、柳浦先生（昨年１０月に転入）も参加されています。（藤戸先生は豊橋技術科学大学に転出されました。）私と高木一義先生は情報システム学専攻の集積システム論講座（大講座：３研究室）に所属しています。

集積システム論というと、半導体回路工学等をイメージされるかもしれませんが、集積回路で構成されるシステムが対象であり、いわゆるシステムLSIや組込みシステムのハードウェア（アーキテクチャ）やソフトウェアに関する研究を行う学問と考えています。現在、年間100億個以上のマイクロプロセッサ（4ビット以上）が出荷されていますが、そのうち、パソコンやワークステーション等の汎用のコンピュータに用いられるのは2％程度であり、ほとんどが組込み用に使われています。実際、あらゆる家電製品にマイクロプロセッサが組み込まれており、一般家庭で40～50個はあるといわれています。また、自動車にも多く使われており、高級車では70～80個使われています。

以前は、集積回路を部品として汎用コンピュータを作るために、汎用機のアーキテクチャ、汎用機のOS、汎用機のソフトウェア、そして部品としてのIC設計の研究が行われていました。しかし、今は、コンピュータ（プロセッサ）は部品となり、メモリや種々の周辺回路とともにボード上、さらには、一つのLSI上に集積されています。

集積（組込み）システムのアーキテクチャ、組込み用プロセッサのアーキテクチャ、組込みOS、組込みソフトウェア、そしてシステムとしてのLSI設計の研究が必要です。

私は、集積システム論は、電気・情報分野の「中心」に位置すると思い（たく）、実際、産業界はシステムLSIや組込みソフトウェアの高度技術者を必要としているのですが、残念ながら、我国の大学では電気電子の分野と情報の分野の「隙間」になっています。企業では、半導体技術者等を社内再教育でこの分野に振り向けているのが現状です。

■ 集積システムの設計支援とハードウェアアルゴリズム

これまでは、本特定領域研究の皆様にはピンと来ない話だったかと思います。しかし、実は、集積システムの設計過程は、組合せ最適化問題の宝庫です。処理のどの部分をハードウェアで行いどの部分をソフトウェアで行うか等を決めるシステムレベルの設計から、LSIのレイアウトまで、最適化問題を解く作業の連続です。最適化の対象は、計算時間や面積に加え、消費電力が重要になっています。ちょっと意外かもしれませんが、よく現れる処理は、ソフトウェアで行うよりハードウェアで行った方が消費電力が少なくて済みます。製造手順の最適化もあります。少し前の話になりますが、IBMの岩野さんらが、プリントボードに多数のホールを開ける作業を、巡回セールスマン問題の解法を応用して効率化されたことを覚えておられる方もおられると思います。

本特定領域研究には、「ハードウェアアルゴリズムの評価に関する研究」という課題で参加させて頂いています。ハードウェアアルゴリズムの評価は、それの基づく回路の性能によって行います。そのためには、回路の安定したモデルが必要です。従来、組合せ回路モデル上での段数および素子数、さらには、配線領域も考慮したVLSIモデル上での面積が評価基準として用いられています。段数は計算時間の指標ですが、現実には、論理ゲートでの遅延よりも配線での遅延の方が大きくなっており、配線遅延を含めた計算時間を評価できる回路モデルと評価基準の確立が必要だと考えています。また、消費電力を評価するための回路モデルと評価基準も必要だと考えています。
近年、LSIの配線層数が増えてきており、さらに、先日、東北大学でLSIチップを貼り合わせた3次元LSIが開発されました。3次元のVLSIモデルも興味深い研究課題ではないかと思います。

我々の研究は、本特定領域研究では「辺境」に位置しますが、皆さんにとっても、いろいろ面白い課題があると思います。

第3回全体会議の報告　　平田富夫（名古屋大学）

昨年の11月21日、22日に開かれました第3回全体会議についてご報告いたします。

これまで科研の集会というと研究発表のみというスタイルが多かったと思いますが、前回6月の第2回全体会議では、メンバーのコラボレーションを促進することを主目的として各研究班を紹介するポスターセッションがもたれました。（そのコンセプトは前回の全体会議企画担当の浅野哲夫先生の記事（ニュースレター第3号）をご覧ください。）

第3回の全体会議もこのコラボレーション路線を引き継ぎました。今回はオープン問題を持ち寄り共同でチャレンジすることをメインにしましたが、この企画は、第2回全体会議の直後に浅野哲夫先生と岩間先生にご相談して決めたものです。皆さんがお持ちのオープン問題で全体会議に話題提供できるものを募集したところ5人の先生方が応じてくださいました。この場を借りてこの5人の先生方にお礼申し上げます。私の印象としては、だれもがすぐ理解でき、しかもアイデアが出せそうな問題で非常によい問題だったと思います。中には他のメンバーの結果と結びついて発展した話題もあったようです。このような試みは短期的な成果を求めるのが目的ではありませんが、これを機会にひとつでも成果に結びつくものが出てくれば今回の全体会議は大成功

といえます。

２日目の講演はSODA、FOCSで発表を予定しているメンバーを中心に、私と柳浦先生が興味をもっている話（は皆さんも興味があるだろうと勝手に考えて）で人選させてもらいました。内容はレベルが高く楽しめたのではないかと思います。ご講演いただいた５人の先生方にこの場を借りてお礼申し上げます。

場所は初日が野依記念学術交流館で、２日目がIB電子情報館大講義室でした。野依記念学術交流館は野依良治先生のノーベル賞受賞を記念して建てられた瀟洒な建物です。カンファレンスホール（２００席）と会議室８室（１室１０名～２０名、間仕切り撤去可）があり、オープン問題セッションの後のディスカッションにはうってつけでした。この会場は人気があり早くからの予約が必要ですが、今回はたまたま１日だけあいていたので使わせてもらいました。

６月に次回会議の日程を決めたにもかかわらず、私の不手際で会議の直前まで案内をだすことができませんでした。伊藤先生はじめ事務局の先生方にはご心配をおかけしました。幸い柳浦先生が１０月に名古屋大学に着任されてからは進捗著しく無事会議を終えることができました。ご協力いただいた皆様に感謝します。

2005年度第2回幹事会議事録　　事務局

日時：2005.11.21, 16:00--17:30

場所：名古屋大学　野依学術交流館　カンファレンスホール

[活動報告]

〇ミニ研究集会

・組合せゲーム・パスル（9/12, 京大）

・複雑ネットワーク・ウェブグラフ（9/13, 関学）

・列挙アルゴリズム（9/27--29, 群馬大）

・Complexity（10/11, 東工大）

〇ポスドク

・西村治道（10月～, 京大）

〇招聘研究者

〇国際会議渡航補助

・FOCS（河原林[東北大]）

〇事務員

・宮本希（８月～）

○国際会議 ランダムネス ［渡辺］

7/18--21, 東北大学

[進捗状況報告／討論]

○教科書シリーズ ［杉原］

順調。脱稿2件

○ニュースレター ［牧野・宇野］

・これまでに3回発行。4回目は来年の1月。

・年に2回程度の予定で進めたい。->了承 (必要ならば、号外を適宜。)

○電子ジャーナル [定兼]

・引き続き、検討を続ける

・協力が必要な場合は、後日に相談

○春学校、ワークショップ (2/27(月) ～ 3/3(金))

・オーガナイザ・プログラム：浅野孝夫

・ローカルアレンジ：西野・垂井

・2/27(月) ～ 3/1(水) 春学校

– 電通大図書館 AV教室 (301号室, 306号室)

– 講師は5名

・3/2(木) ～ 3/3(金) ワークショップ

– 調布クレストンホテル

・春学校 … 初学者の道案内を目指す (定員 90名)

(※ 参考: 春学校に参加させてみたい学生数 … 会場だけで 30名以上)

近隣の諸国にもアナウンスを流す

・春学校/ワークショップではregistration fee を徴収する (高額ではない)

○ミニプロジェクト

・ゲーム [伊藤]

－ミニ研究集会を開催

－ 今後、スクールみたいなのを開催したい

・Complexity [垂井]

－ミニ研究集会を開催、今後も開催予定

－ 学生にも声をかけたい

・ジオメトリ [加藤]

－ ３月後半、学生対象の mini school

－ 講師 加藤・浅野・徳山・(David Avis)

・列挙 [中野]

－ミニ研究集会を開催

〇電子情報通信学会 特集号 ［増山］

・来年６月に、和文論文誌、特定の解説論文の特集

〇ICALP ワークショップ

・SATの厳密解法関係のミニワークショップの提案（渡辺先生）

－ヨーロッパの人々とも相談する必要あり

（後日、渡辺先生によって申請、採録された。）

〇電子情報通信学会 英文論文誌 ［和田］

・来年の８～１０月頃、〆切は2月頃

・基本は研究論文 (解説論文も ok かも)

・もう少し検討

〇予算の使途

・国際会議発表のための海外渡航の補助

〇次回の全体会議

・九州地区（山下、定兼）

・2006年、５月か６月頃

〇電子情報通信学会 全国大会 企画 ［山下］

・3月24日(金) ～ 27日(月) 国士舘大学

・若手(主に博士課程の学生)のためのシンポジウム

・参加費と旅費は、総括班から支援

〇次回以降の春学校/ワークショップ

・来年度からは、時期を夏休みぐらいにしたい

・早急に内容の相談を始める必要がある

〇EATCS記事募集 [牧野]

・特定内外で、記事があれば、牧野まで

〇特定領域としての成果

・どんな集まりで、どんなアイデアが出て、どんな進展・研究があったか

どんなに細かいことでも良いので、事務局まで教えてほしい

〇次の特定領域

・来年は、調査研究の申請をする

・そのために、今の特定領域で、領域として集まった成果を挙げる

「新世代の計算限界」イベントカレンダー　＋　事務連絡

成果報告書の執筆依頼

今年度の成果報告書を領域として作成する時期となりました. 詳細は別途メール・ウェブで御連絡いたしますが, 昨年度と同じ要領で, A01 から C11 の研究課題ごとに「研究活動報告」と「発表文献リスト」を御執筆いただけますようお願い申し上げます. なお, 提出〆切は 3月31日(金) です.

http://keisan-genkai.lab2.kuis.kyoto-u.ac.jp/members/reports.html

旅費の補助

総括班予算より国際会議等の旅費の補助を行います。補助希望者は伊藤までお申し出ください。特に締め切りなどは設定しておりません。

・ジオメトリ合宿　(加藤直樹 浅野哲夫、徳山豪、David Avis)

2006年3月15日昼から17日まで

コミュニティ嵯峨野（京都府勤労者研修センター）（http://www.com-sagano.com/）

〒616-8372 京都市右京区嵯峨天龍寺広道町3-4

・COMP-NHC 学生シンポジウム（田中圭介）

2006年3月24日(金)～26 日(日)のいずれか一日、国士舘大学世田谷キャンパス(東京都世田谷区)

電子情報通信学会 2006 年総合大会 シンポジウム講演 (B) DS-1「COMP-NHC 学生シンポジウム」

（総合大会は2006年3月24(金)～27(月)）

3/27(月)-29(水) EWCG'06 (22nd European Symposium on Computational Geometry) Delphi, Greece
3/31(金)-4/2(日) Algebra and Combinatorics 2006 Manila, Philippines
4/24(月)-25(火) 第19回 回路とシステム軽井沢ワークショップ 軽井沢プリンスホテル西館
5/21(日)-23(火) STOC'06 Seattle, Washington, USA
6/5(月)-7(水) SoCG'06 Sedona, Arizona, USA
6/20(火)-22(木) The Second International Conference on Algorithmic Aspects in Information and Management 香港
6/22(木)-24(土) WG2006 (グラフアルゴリズムの会議です) ノルウェー, ベルゲン
6/25(日)-28(水) INFORMS International Hong Kong 2006 香港
7/3(月)-5(水) SIROCCO2006 Chester, United Kingdom
7/6(木)-8(土) SWAT 2006 リガ (ラトビア)
7/9(日)-16(日) ICALP 2006 ベネチア (イタリア)
7/30(日)-8/4(金) ISMP2006 Rio de Janeiro, Brazil
8/15(火)-18(金) COCOON 2006 台北（台湾）
8/22(火)-24(木) IFIP/TCS 2006 Santiago, Chile
9/11(月)-15(金) ALGO 2006 チューリッヒ (スイス)

このニュースレターについて

ニュースレター各号は電子メールで配布する予定です．短い記事や連絡事項は全て掲載しますが，長い記事，イベントの詳細などはwebページに掲載する予定です．webページには詳細まで全てを載せた完全版を掲載して，目次，あるいは各記事の末尾のURLを参照すると，web版の同じ記事を参照できるようにいたします．

記事は，各回，1つの研究課題に担当をお願いする予定です．各研究課題で2000-4000字程度，研究に関わる記事を書いていただければと思います．通常，このようなニュースレターでは，研究成果を報告するのが一般的だと思われますが，この特定領域では「研究者の交流」に焦点を当てたいため，「研究の成果以外」の記事を面白く解説していただければと思います．例えば，最近参加した国際会議の情報を，どのようなものが流行っていたか，何が面白かったか，などの主観的な解説を交えて報告したり，最近考えている問題，あるいはオープン問題を，この辺までは解けるがここがうまくいかない，といった解説を交えて紹介する，という形です．

また，研究者間の交流を促進するため，各研究者の，国内外の会議への出席予定を集約して掲載していこうと考えています．研究者の交流には，顔をあわせる回数を増やすことが肝要です．他の研究者の参加予定

がわかれば，会議への出席のモチベーションを高めることにもつながり，それがディスカッションや研究成果を生むきっかけにもなるでしょう．特定領域メンバーの皆さんには，自分のわかる範囲で，国内外の会議・研究会の情報と，自分の参加予定を教えていただければと思います．

この他，個人からの寄稿を募集いたします．100-1000字程度で，情報宣伝されたいことを自由な形式で書いて送っていただければ，掲載いたします．メールで配布する関係上，テキスト形式のものしか扱えませんが，そこはご了解お願いいたします．

次号は6月ごろを予定しています．

★ ニュースレター編集委員では，皆様からのご意見をお待ちしております．編集方針や内容の追加など編集全体にかかわることから細かいことまで，幅広いご意見をお願いいたします．

■■ 新世代の計算限界 ニュースレター ■■

編集委員長 宇野 毅明 uno@nii.jp（問合せ先）

副編集委員長 牧野 和久 makino@sflab.sys.es.osaka-u.ac.jp

平成17年度 第1回 全体会議

日時　平成17年6月16日(木), 17日(金)
会場　国立情報学研究所12階会議室

プログラム

6月16日(木)

10:00 – 10:50　A01–A10班の研究紹介
11:00 – 12:00　ポスターセッション(A01–A10班)
12:00 – 14:00　幹事会
14:00 – 14:30　B01–B06班の研究紹介
14:40 – 15:40　ポスターセッション(B01–B06班)
16:00 – 16:55　C01–C11班の研究紹介
17:05 – 18:00　ポスターセッション(C01–C11班)

6月17日(金)

9:30 – 12:00　全体会議

時間枠つき配送計画問題に対するメタ戦略アルゴリズム

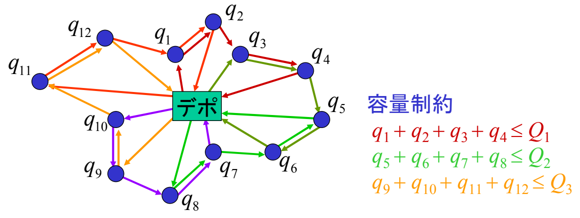
柳浦睦憲 （京都大学）
with
橋本英樹（京都大学） 茨木俊秀（関西学院大学）
今堀慎治（東京大学） 久保幹雄（東京海洋大学）
増田友泰（アクセンチュア） 野々部宏司（法政大学）
祖父江謙介（トヨタ） 宇野毅明（国立情報学研究所）

時間枠付配送計画問題

入力: 節点 $V = \{0, 1, \ldots, n\}$ (0: デポ, $i \geq 1$: 客), 車両 $M = \{1, \ldots, m\}$,
距離 d_{ij}， 移動時間 t_{ij}， 容量 Q_k， 重み q_i

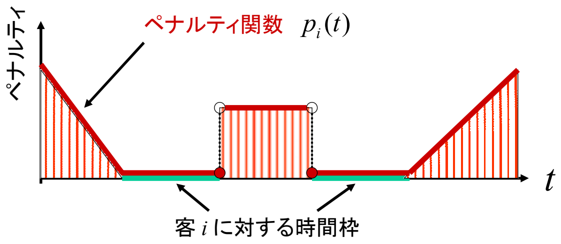
出力: 総距離を最小にする車両ルート

制約: 容量制約 および 時間枠制約



一般化時間枠

各客はサービスを受ける時刻に対する希望を提示する



$p_i(t)$: 非凸かつ不連続でもよい
（ただし区分線形関数のみ）

目的関数

$$\boxed{cost(\sigma) = D(\sigma) + T(\sigma) + Q(\sigma)}$$

σ	車両スケジュール
$D(\sigma)$	総距離
$T(\sigma)$	総時間ペナルティ
$Q(\sigma)$	総容量超過

時間枠制約および容量制約
⬇
ソフト制約

計算例

配送計画問題の基本モデルの問題例

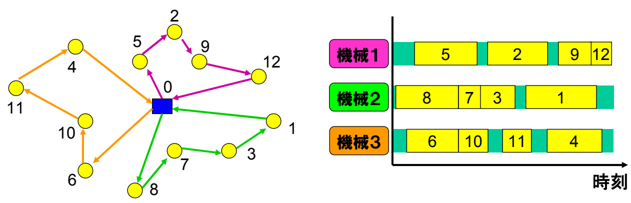
計算例1

時間枠つきの問題例

計算例2

スケジューリング

客	⇔	仕事
車両	⇔	機械
移動時間	⇔	処理時間とセットアップ時間
時間枠制約	⇔	準備時間や納期など
容量制約	⇔	資源制約



問題構造と探索空間

車両スケジュール:

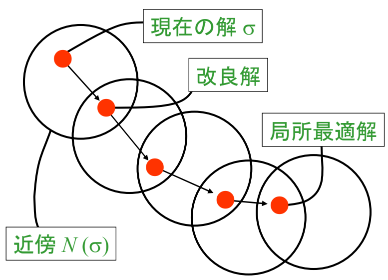
(a) 客の車両への割当
(b) 各車両における客の訪問順序
(c) 各客のサービス開始時刻

(a) と (b) ⟶ 局所探索法により探索

(c) ⟶ 動的計画法により最適に解く
((a)と(b)は固定という条件の下で)

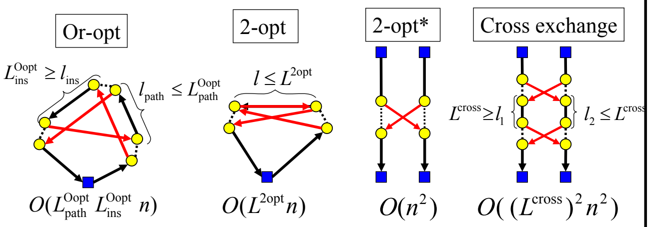
局所探索法

適当な初期解から始め、現在の解 σ の近傍 $N(\sigma)$ の中に σ よりもよい解 σ′があれば σ:= σ′と置き換える操作を反復



- 探索空間
- 解の評価法
- 近傍
- 初期解生成法

近傍操作



近傍内の解: 現在の解のパスを高々4つつなぎ合わせることで生成できる

最適サービス開始時刻決定問題

もともとの目的関数 $cost(\sigma) = D(\sigma) + \boxed{T(\sigma)} + Q(\sigma)$

問題（車両ルートは固定）
Optimal Start Time Problem (OSTP)

入力: 車両 k の客の訪問順序
$\sigma_k(0), \sigma_k(1), \ldots, \sigma_k(n_k), \sigma_k(n_k+1)$

出力: 車両 k の時間ペナルティを最小にするような、各客のサービス開始時刻

➡ 動的計画法により厳密に最適化

OSTPの関連研究

時間ペナルティ関数が一般の場合

Ibaraki, Imahori, Kubo, Masuda, Uno & Yagiura, 2005

時間ペナルティ関数が凸の場合

- 関連研究:
  Garey, Tarjan & Wilfong 1988
  Powell & Solomon 1992
  Davis & Kanet 1993
  Taillard, Badeau, Gendreau, Guertin & Potvin 1997
  Tamaki, Komori & Abe 1999
  Ahuja & Orlin 2001
  Hochbaum & Queyranne 2003
- OSTP は線形計画問題になる
- isotonic median regression 問題を含む
- $O(n_k \log n_k)$ 時間のアルゴリズムあり

動的計画法

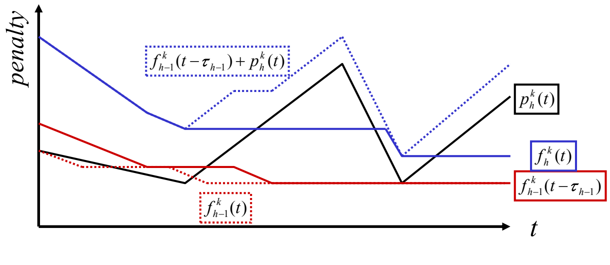
$f_h^k(t)$: 客 $\sigma_k(0), \sigma_k(1), \ldots, \sigma_k(h)$ が時刻 t 以前にサービスされるときのこれらの客の時間ペナルティの最小値

$$f_0^k(t) = \begin{cases} +\infty, & t \in (-\infty, e_0) \\ 0, & t \in [e_0, +\infty) \end{cases}$$

$$f_h^k(t) = \min_{t' \leq t}(f_{h-1}^k(t' - \tau_{h-1}) + p_h^k(t')), \quad 1 \leq h \leq n_k + 1$$

e_0: 車両のデポからの出発可能時刻
τ_h: h 番目の客から $h+1$ 番目の客への移動時間
$p_h^k(t)$: h 番目の客の時間ペナルティ関数

動的計画法の実行例

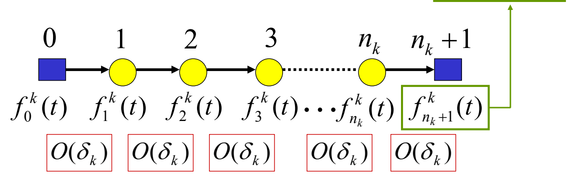


$$f_h^k(t)=\min_{t'\le t}(f_{h-1}^k(t'-\tau_{h-1})+p_h^k(t'))$$

動的計画法の計算時間

n_k: 車両 k のルート内の客数

δ_k: 車両 k のルート内の客の時間ペナルティ関数の線形区分数の合計 (通常 $\delta_k=O(n_k)$)



総計算時間: $O(n_k\delta_k)$

時間ペナルティの評価の高速化

アイデア
- 近傍内の解の評価に現在の解 σ の $f_h^k(t)$（計算済）を再利用
- 解が更新されたときには動的計画法(DP)を再計算

ペナルティ関数が一般の場合:
近傍内の解の評価: $O(\delta_k)$
解の移動の際のDPの再計算: $O(n_k\delta_k)$

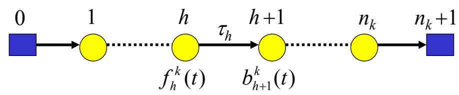
ペナルティ関数が凸の場合:
近傍内の解の評価: $O(\log\delta_k)$
解の移動の際のDPの再計算: $O(\delta_k\log\delta_k)$

平衡木を利用

時間ペナルティの高速計算

$b_h^k(t)$: 客 $\sigma_k(h), \sigma_k(h+1), \ldots, \sigma_k(n_k+1)$ が時刻 t 以降にサービスされるときのこれらの客の時間ペナルティの最小値

$f_h^k(t)$ と同様に計算できる



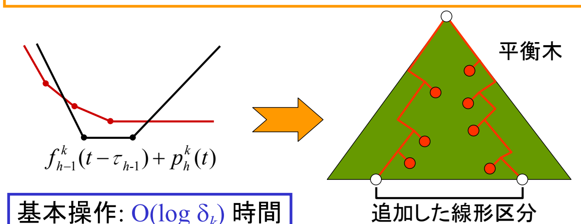
ルート k の最小ペナルティ: $\min_t\left(f_h^k(t)+b_{h+1}^k(t+\tau_h)\right)$

- ペナルティが一般の場合 $O(\delta_k)$, 凸の場合 $O(\log\delta_k)$ 時間
- 2-opt*には直接利用可. それ以外の場合は交換されるパス内の客の $f_h^k(t)$ を効率よく再計算できるよう探索順序に工夫の必要あり.

DP: 時間ペナルティが凸の場合

$p_i(t)$: 全ての i に対して凸 $\Rightarrow f_h^k(t)$: 凸

$f_h^k(t)$ を平衡木で表現
- 葉節点 ↔ $f_h^k(t)$の線形区分のひとつ
- 根から葉へのパス
  ↔ 葉に対応する線形区分の傾きや切片などの情報



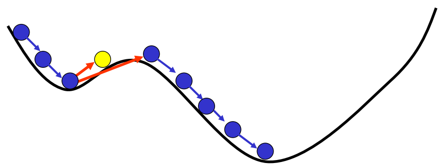
基本操作: $O(\log\delta_k)$ 時間
総計算時間: $O(\delta_k\log\delta_k)$

効率化のアイデア

- 近傍の制限
  - 近傍リスト: TSP や VRP に対し大きな効果
  - サービス開始時刻が近い客のペア
- 動的計画法の関数 $f_h^k(t)$ の区分数の削減
  - 関数 $f_h^k(t)$ の線形区分の中で対応するペナルティ値が非常に大きいものを削除

反復局所探索法 (ILS)

- 局所探索を多数反復
- 過去の探索で得られたよい解に少しのランダムな変形を加えたものを初期解



計算実験

問題例

- Gehring & Homberger のベンチマーク問題例（Solomonの問題例と同様の生成法．より大規模）
- 時間枠制約と容量制約: 絶対制約
- 客数: 200, 400, 600, 800, 1000
- 各サイズに対して60問（全部で300問）

計算環境と実験方法

- C 言語; PC（Pentium IV 2.8GHz）
- 絶対制約 → ペナルティ関数; 適応的重み調整
- 出力する解: 絶対制約を満たす解
- 車両数: 既知の最良値（± 1）

number of customers		best known	MB (2003)	GH (2002)	LC (2003)	our ILS
200	CNV	692	694	696	694	694
	CTD	169281	168537	179328	173061	170331
	time (min)		5.88	3.83	21.66	33.3
400	CNV	1386	1389	1392	1390	1384
	CTD	392444	390386	428489	408281	401285
	time (min)		12.49	12.95	43.32	66.6
600	CNV	2076	2082	2079	2088	2070
	CTD	799355	796172	890121	836261	827192
	time (min)		29.39	23.53	64.97	100
800	CNV	2754	2760	2765	2766	2750
	CTD	1429914	1535849	1361586	1475281	1426133
	time (min)		42.32	106.53	86.63	133.3
1000	CNV	3438	3446	3446	3451	3434
	CTD	2106125	2078110	2290367	2225366	2169452
	time (min)		440.82	361.2	108.29	166.6

CNV: cumulative number of vehicles
CTD: cumulative total distance

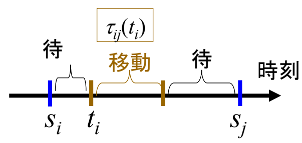
MB: Mester and Bräysy
GH: Gehring and Homberger
LC: Le Bouthillier and Crainic

より高い汎用性を目指して

- 時刻依存の移動時間および移動コスト
  - 朝夕のラッシュ
  - 生産ラインにおけるセットアップ
- 移動時間が可変，移動時間に応じたコスト
  - 有料道路の利用により移動時間を短縮
  - 特急仕事として高速処理

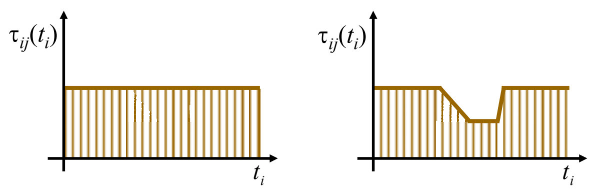
→ 一般の時間ペナルティをもつ問題に上のいずれかを追加した問題を考える

時刻依存の移動時間 $\tau_{ij}(t_i)$



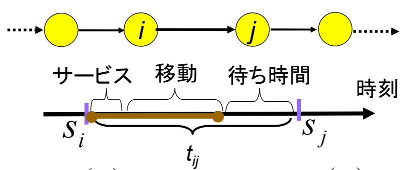
- 連続
- FIFO

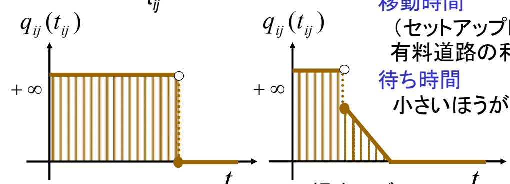
$t < t' \Rightarrow t + \tau_{ij}(t) < t' + \tau_{ij}(t')$



(a) 従来のモデル　(b) 提案モデル

移動時間コスト関数 $q_{ij}(t_{ij})$





(a) 従来のモデル　(b) 提案モデル

サービス時間
（処理時間）
人手やコストをかけて短縮可能
移動時間
（セットアップ時間）
有料道路の利用など
待ち時間
小さいほうがよい場合

計算の複雑さに関する結果

- 時刻依存の移動時間および移動コスト
  - 移動時間関数の総複雑度に対して線形時間のDP
  - 移動時間関数の複雑度が各客ペアでO(1)
    → 移動時間が定数の場合と同程度の計算時間
- 移動時間が可変，移動時間に応じたコスト
  - 移動時間コスト関数が一般
    → NP困難; 擬多項式時間のDP
  - 移動時間コスト関数が凸（時間ペナルティ関数は一般）→ 多項式時間のDP

まとめ

- 時間枠に対する要求を一般的なペナルティ関数として表現できる汎用的な問題の定式化
  → 時間ペナルティ最小化問題を解く必要性
  → 動的計画法の提案
- 高速化のさまざまな工夫
- Gehring & Homberger のベンチマーク問題例に対する良好な結果
- 移動時間に対する汎用的な定式化

今後の課題

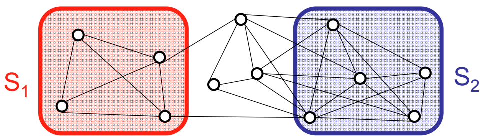
- 異なるタイプの汎用化
- 配送ルートの探索能力の向上
- 大規模問題例への適用
- 時間枠の取り扱い →
  より実装しやすい単純なアルゴリズムで
  高速かつ実用的なものの開発
  cf. 提案手法はいずれも非常に複雑で実装困難

クリークと列挙

- 最大クリーク問題
  - 近似度$\Omega(n^{1-\varepsilon})$ [Hastad 99]
- 極大クリークの列挙
  - O(nm)時間/個[Tsukiyama, et al. 77]
  - $O(\Delta^4)$時間/個[Uno03](Δは最大次数)
  - しかし指数($O(3^{n/3})$)個存在[Moon, et al. 65]

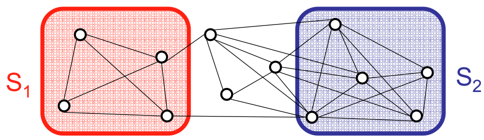
Why Cliques?

- Inside: Densely connected.
- Inside-Outside: Sparsely connected.



Isolated cliques

- Let c>0 be a constant. A clique $S \subseteq V$ with k vertices is an c-isolated clique if $|E(S)|<ck$. ($E(S)$={edges between S and V-S}.)
- 1-isolated cliques = isolated cliques.



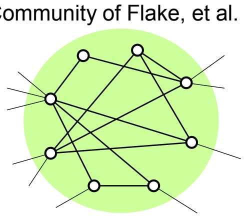
$|E(S_1)|=2<4 \Rightarrow$ isolated clique

$|E(S_2)|=9 \geqq 5 \Rightarrow$ non isolated clique

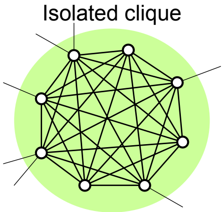
Related work

- Flake, Lawrence, Giles (2000)
  - Community $S \subset V$: $|E(v,S)|>|E(v,V-S)|\ \forall v \in S$.

Community of Flake, et al.



Isolated clique

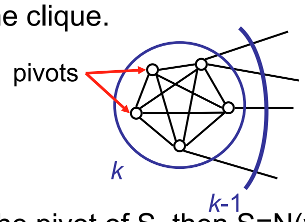


Preliminary Result

- Theorem 0.1. All isolated cliques can be enumerated in linear time.
- Corollary 1. The # of isolated cliques is $O(m)$ for any graph.

Observation for Proving Theorem 0.1

- Lemma 1. An isolated clique has a vertex (called a pivot) that has no outgoing edge from the clique.



- If v is the pivot of S, then $S=N(v)\cup\{v\}$.

Strategy of enumeration

- Check each vertex whether or not it can be a pivot.
- Sort and renumber all vertices as $d(v_1)\leqq d(v_2)\leqq\ldots\leqq d(v_n)$.
- If $\exists j>i$, $v_j\in N(v_i)$ (adjacent vertices of v_i), v_i can be ignored (we can consider that the vertex having the minimum index in a isolated clique is the pivot): Test (a).

Observation 2

- Lemma 2. If S is an isolated clique and v is the pivot, then $\sum_{w\in S} d(w) < (d(v)+1)^2$. (1)
- Proof. Let $k=|S|=d(v)+1$. $\sum_{w\in S} d(w)<k(k-1)+k=k^2 =(d(v)+1)^2$. □

Observation 2

- Lemma 3. If v has the minimum indices in $S=N(v)\cup\{v\}$ and S satisfies (1), then $d(w)<2d(v)+1$ $\forall w\in S$.
- Proof. Let $k=|S|=d(v)+1$. If $d(w)\geqq 2d(v)+1=2k-1$ for a $w\in S$, $\sum_{w\in S} d(w)\geqq (k-1)^2+2k-1=k^2 =(d(v)+1)^2$, contradiction. □

Strategy 2

- If v passes Test (a), we check whether $S=N(v)\cup\{v\}$ satisfies (1): Test (b). (This can be done in $O(d(v))$ time.)
- If N(v) passes this test, $d(w)<2d(v)+1=O(d(v))$ $\forall w\in N(v)$ from Lemma 3.

Observation 3

- Lemma 4. If $S=\{v=w_1, \ldots, w_k\}$ is an isolated clique and v is the pivot of S ($d(w_1)\leqq\ldots\leqq d(w_k)$), then $S_i=\{w_1, \ldots, w_i\}$ has at most i-1 outgoing edges from S.
- Proof. Assume that $|E(S_i,V-S)|\geqq i$. Then $d(w_i)\geqq d(v)+1$, and hence $d(vj)\geqq d(v)+1$ for all $j=i+1, \ldots, k$. Therefore $\sum_{w\in S} d(w)= \sum_{w\in S_i} d(w)+\sum_{v\in S-S_i} d(w) \geqq i+(k-i)=k$, contradiction.

Strategy 3

- Assume that v passed Tests (a) and (b) (primal tests).
- Let $S=N(v)\cup\{v\}=\{v=w_1, w_2, \ldots, w_k\}$ $(d(w_1)\leqq d(w_2)\leqq\ldots\leqq d(w_k))$.
- Clique test: From i=1 to k,
  - check whether (1) w_i is adjacent to $w_1, \ldots, w_{i-1}$ (i.e., $S_i=\{w_1, \ldots, w_i\}$ is a clique) and
  - (2) S_i has at most i-1 outgoing edges from S.
  - If not, *v* is not a pivot and then skipped (finish checking v).

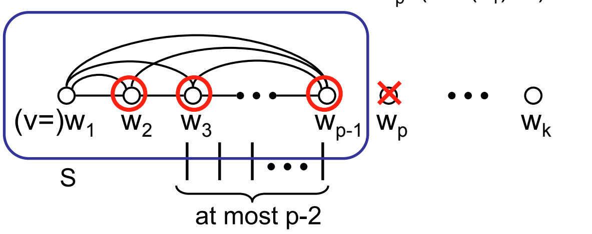
Running time

- Sorting vertices by their degrees: O(m).
- Primal tests: O(d(v)) for each $v\in V$, i.e., O(m).
- Clique test: Assume the test breaks at w_p. $d(w_1)+d(w_2)+\ldots+d(w_p)=O(w_p^2)\rightarrow O(m^2)$?

→ More precise estimation!

Running time (Cont.)

- Assume the test is done until w_p. $(k=d(v_1)+1)$



By v: $O(d(w_1)+d(w_2)+\ldots d(w_p))=O(pk)$
By other pivots: $O((p-1)d(w_{p-1}))=O(pk)$
$\Big\}$ O(pk)

→ Amortize as $O(k)=d(w_i)$ for each vertex in S.

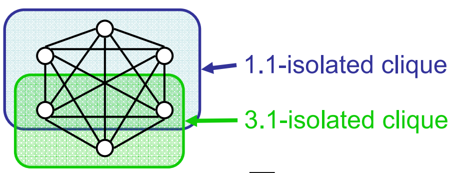
Running time (Cont.)

- Sorting vertices by their degrees: O(m).
- Primal tests: O(d(v)) for each $v\in V$, i.e., O(m).
- Clique test: O(d(v)) for each $v\in V$, i.e., O(m).

→ O(m). ··· Theorem 0.1 is proved.

Extension

- Theorem 0.1. All isolated cliques can be enumerated in linear time.
- For general c-isolated cliques?



- Maximal ones are important.

Results

- Theorem 1. All maximal *c*-isolated cliques of a graph with *n* vertices can be enumerated in $O(c^5 2^{2c} m)$ time.

- Corollary 1. For any constant *c*, all maximal *c*-isolated cliques can be enumerated in linear time.
- Corollary 2. For any $c=O(\log n)$, all maximal *c*-isolated cliques of a graph with n vertices can be enumerated in polynomial time.

Results (cont.)

- Theorem 2. Let c, x, and y are functions of n s.t. $c=xy$. There is a graph with m edges for which the # of maxmal c-isolated cliques is $\Omega((2x^y/c^2)m)$.

- Cor. 3. If $c=\omega(1)$, there is a graph with n vertices consisting of super-linear # of maximal c-isolated cliques.
- Cor. 4. If $c=\omega(\log n)$, there is a graph with n vertices consisting of super-polynomial # of maximal c-isolated cliques.

Results (cont.)

- Cor. 1. If $c=O(1)$, all maximal c-isolated cliques can be enumerated in linear time.
- Cor. 3. If $c=\omega(1)$, there is a graph consisting of super-linear # of maximal c-isolated cliques.

- $c=\Theta(1)$ is the tight bound for enumerating all maximal c-isolated cliques in linear time.

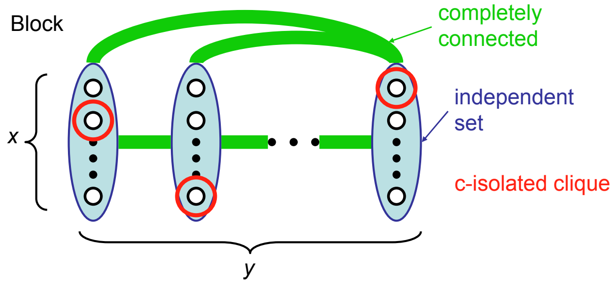
Results (cont.)

- Cor. 2. If $c=O(\log n)$, all maximal c-isolated cliques can be enumerated in polynomial time.
- Cor. 4. If $c=\omega(\log n)$, there is a graph consisting of super-polynomial # of maximal c-isolated cliques.

- $c=\Theta(\log n)$ is the tight bound for enumerating all maximal c-isolated cliques in polynomial time.

Proof of Theorem 2

- Theorem 2. Let c, x, and y are functions s.t. $c=xy$. There is a graph with m edges for which the # of maxmal c-isolated cliques is $\Omega((2x^y/c^2)m)$.



Proof of Corollaries 3 and 4

- If $c=\omega(1)$, then by letting x=2, y=c/2 $(2x^y/c^2)m$ becomes super-linear.
- If $c=\omega(\log n)$, then by letting x=c/log$n$, y=log$n$ $(2x^y/c^2)m$ is super-polynomial.

Other Results: Pseudo-Clieques

- Let $\alpha(k)$ and $\beta(k)$ are functions. Pseudo-Clique $PC(\alpha, \beta)$ is a vertex-proper-subset $S \subset V$ ($|S|=k$) s.t.
- $\mathrm{av}_{v\in S}\, d_{G(S)}(v) \geqq \alpha(k)$ and
- $\min_{v\in S} d_{G(S)}(v) \geqq \beta(k)$.

Results for PC

- Theorem 3. Suppose $f(k)=\Omega(1)$ and $0<\varepsilon<1$ is a constant.
  - There is a graph including super-poly. # of maximal isolated PC($k-f(k),k^{\varepsilon}$).
  - There is a graph including super-poly. # of maximal isolated PC($k-k^{\varepsilon},k/f(k)$).
- Proposition 1. All maximal isolated PC(α,c_1k) and PC($k-c_2,k^{\varepsilon}$) are enumerated in poly. time for constant $c_1<1$ and $c_2\geqq 1$.

Results for PC (Cont.)

- Theorem 4. There is a graph including super-poly. # of maximal isolated PC($k-(\log k)^{1+\varepsilon},k/(\log k)^{1+\varepsilon}$) for any $0<\varepsilon$.
- Theorem5. All maximal isolated PC($k-\log k,k/\log k$) can be enumerated in poly. time.

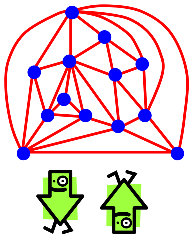
Summary

- Introduce f-isolated cliques with parameter function c.
- All c-isolated cliques can be enumerated in linear time for any constant c.
- $c=\Theta(1)$ is the tight bound of linear time enumeration.
- All c-isolated cliques can be enumerated in poly. time if $c=O(\log n)$.
- $c=\Theta(\log n)$ is the tight bound of poly. time enumeration.

新世代の計算限界　全体会議　2005/06/17

Compact Encoding of Plane Triangulations with Efficient Query Support

クエリを効率的にサポートする極大平面グラフのコンパクトな符号化



山中克久　○中野眞一

群馬大学

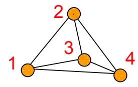
m 辺

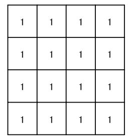
2m bit　＋少し

0000111001101110011001110010101101100110 01

超大規模なグラフを扱いたい

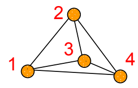
隣接行列　　n^2 bit

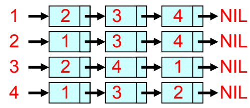




指定した2点間に辺ある？

隣接リスト　　2m log n　bit　+ (ポインタ)





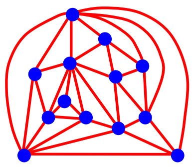
指定した1点に接続する辺のリストは？

超大規模なグラフを扱いたい

インターネットのグラフ
3Dモデル

- グラフスパナー
（本質的でない辺を削除）
(非可逆)

- 巧妙な符号化 (定数bit / 各辺)
（＋クエリの定数時間サポート）



点11の次数は？

4だよ！

点3と5の間に辺ある？

YES!

高速化？　ページング減少？
高速クエリ

011001110010101101100110 01

既知の結果その1

・もし、対象のグラフがt個あれば、
　平均の符号長は　log t bit 以上必要
・情報理論的な下限　= log t
　極大平面グラフの場合
　　　　　1.08 m bit　[Tutte 62]
　木の場合　　2m – o(n) bit
・任意の列挙アルゴリズムを利用して符号化できる
　k番目のグラフを　kの2進数表記に符号化する
　ただし、符号化・復号化に指数時間かかる

既知の結果その2

クエリサポートなし

・一般の平面グラフ	4m bit	[Turan 84]
・一般の平面グラフ	3.58m bit	[Keeler他 95]
・極大平面グラフ	1.53m bit	[同上]
・極大平面グラフ	1.33m bit	[He他 99]
・極大平面グラフ	1.33m bit	[Poulalhon 03] ICALP 2003
	1.08m bit 下限	

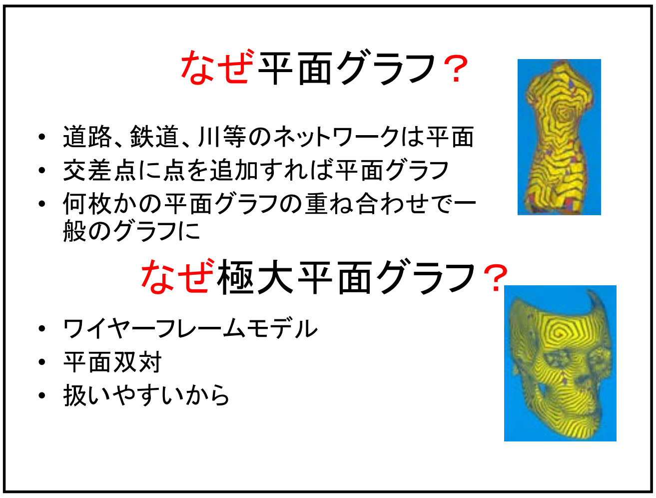
グラフを符号化して送り、複合化して使用

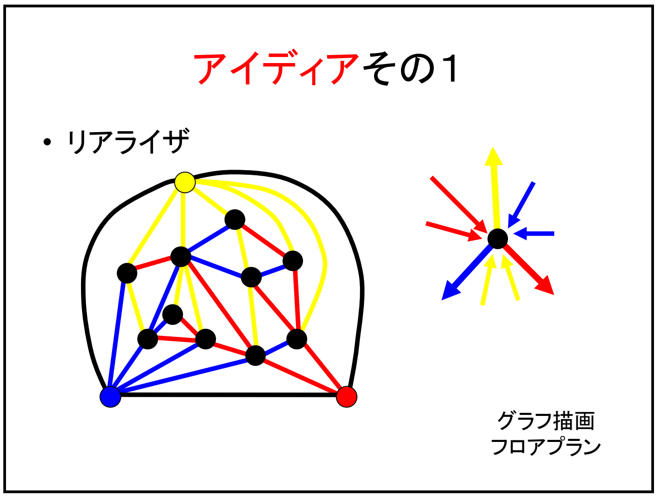
既存の結果その3

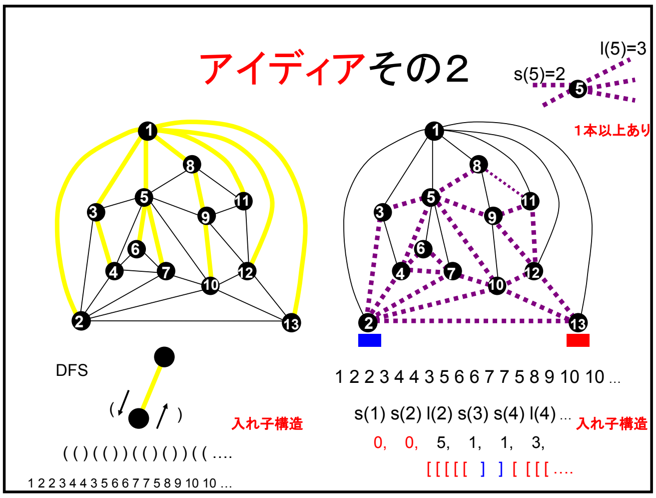
クエリサポートあり

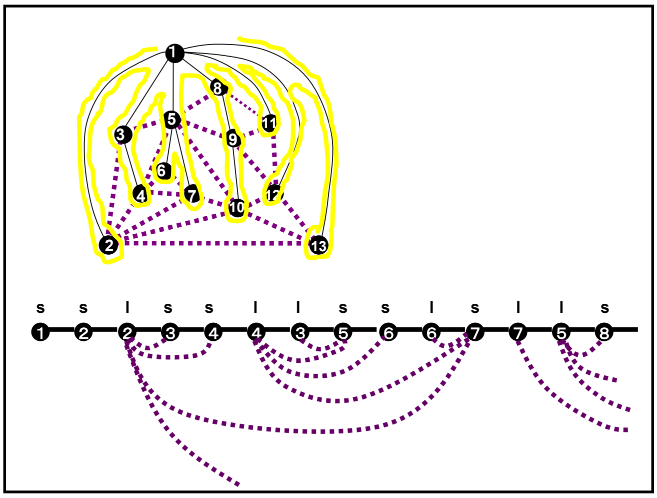
・木	2m +o(n)bit	[Jacobson 89] FOCS 89
・一般平面グラフ	2m+8n+o(n)bit	[Munro他97] FOCS 97
・一般平面グラフ	2m+2n+o(n)bit	[Chiang他01] SODA 01
・極大平面グラフ	2m+n+o(n)bit	[Chuang他]
一般平面グラフ	2m+5n+o(n)bit	ICALP 98
・極大平面グラフ	2m+o(n)bit	[中野 2005]

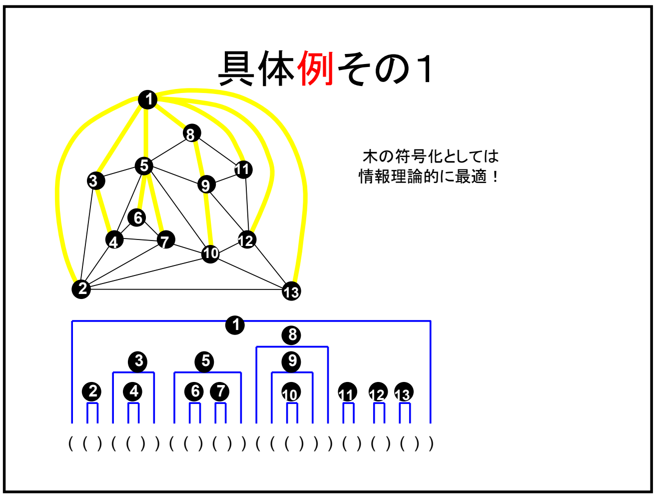
グラフを符号化して記憶し、複合化せずに使用

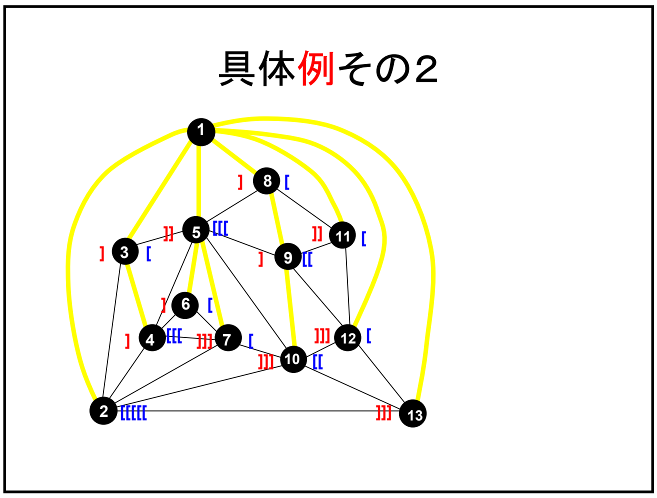

なぜ平面グラフ？
• 道路、鉄道、川等のネットワークは平面
• 交差点に点を追加すれば平面グラフ
• 何枚かの平面グラフの重ね合わせで一般のグラフに
なぜ極大平面グラフ？
• ワイヤーフレームモデル
• 平面双対
• 扱いやすいから


アイディアその1
• リアライザ
グラフ描画
フロアプラン

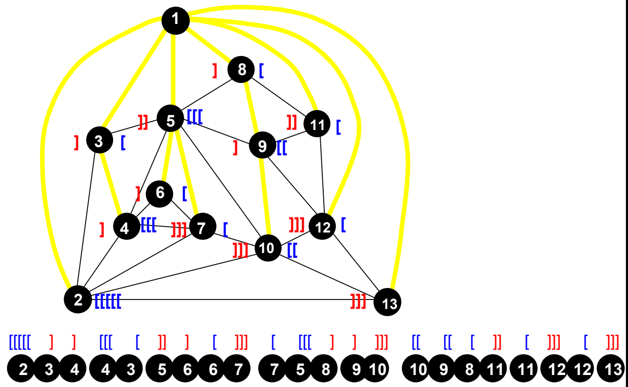

アイディアその2
l(5)=3
s(5)=2
1本以上あり
DFS
入れ子構造
(( ) (( )) (( ) ( )) ((....
1 2 2 3 4 4 3 5 6 6 7 7 5 8 9 10 10 ...
1 2 2 3 4 4 3 5 6 6 7 7 5 8 9 10 10 ...
s(1) s(2) l(2) s(3) s(4) l(4) ...
入れ子構造
0, 0, 5, 1, 1, 3,
[[[[[ ]] [[[[....


s s l s s l l s s l s l l s

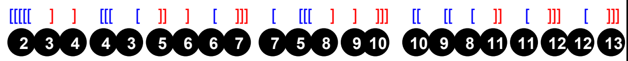

具体例その1
木の符号化としては
情報理論的に最適！
(( ) (( )) (( ) ( )) ((( ))) ( ) ( ) ( ))


具体例その2

具体例その3



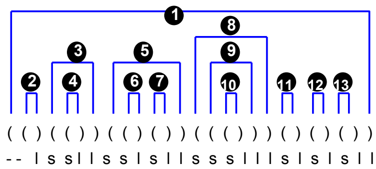
具体例その4



00001 1 1 001 1 01 1 1 001 1 001 .....

各ブロックは必ず1で終わるので長さがわかる

[か] の区別は木のコードをみればわかる。



bit 数の見積もり

- 各辺につき2bit 必要　　計2m bit

- 木の辺　　　----- 　(と)
- 木以外の辺　-----　 [と]

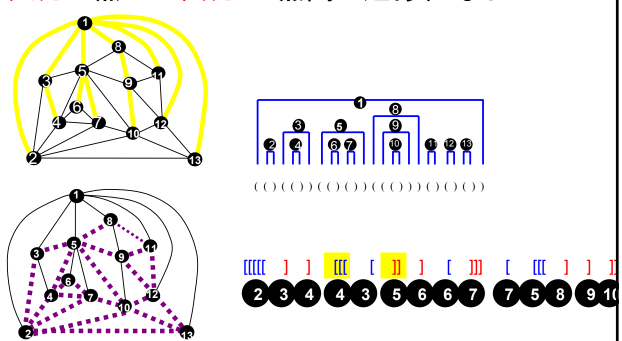
クエリのサポートその1

既知の結果（多段のテーブルを利用）
[Munro他 97 ＋ Chiang他 01]

- bit列が与えられたとき、o(n) bitの補助テーブルを利用して rank(p), select(i) を 定数時間で計算できる。
- バランスされた括弧の列が与えられたとき、o(n) bit の補助テーブルを利用して findclose(p), findopen(p), enclose(p), wrapped(p) を定数時間で計算できる。

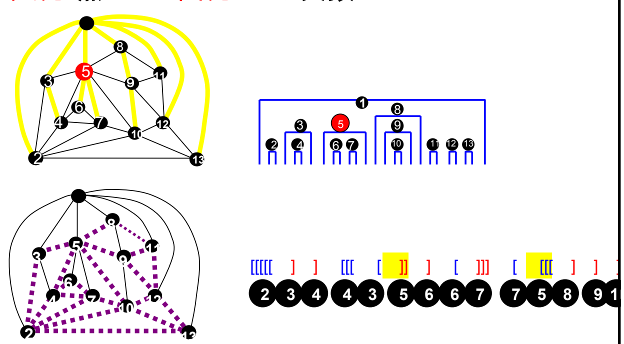
隣接クエリ

入力 2点　　出力 2点間に辺あり・なし



次数クエリ

入力 点v　　出力 vの次数



隣接点のリスト

- 隣接点のリストをO(d(v))時間で作成できる

略

まとめ

既知のベスト(クエリサポートつき)
極大平面グラフ　2m+n+o(n)bit　[Chuang他 ICALP98]

今回の結果
極大平面グラフ　2m+o(n)bit　[中野 05]

主なアイディア
リアライザの3本の木を
1本 ＋ 2本 とみなすこと

おしまいです

- 御静聴ありがとうございます。
- 質問・コメント・アドバイス等歓迎します。
- 応用？

中野眞一(群馬大学)

オンライン予測の理論と応用

東北大学大学院情報科学研究科
A01班　瀧本 英二

クイズ王に匹敵する問題

「タイガーウッズは寅年生まれである. ○か×か？」

正解は

×

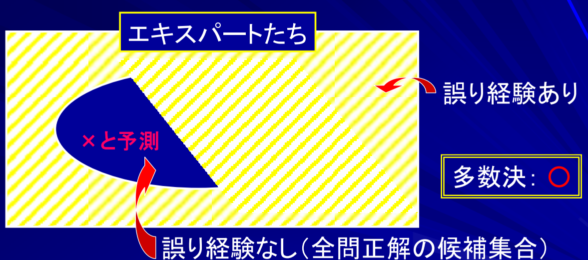
タイガーウッズはウサギ年生まれ

二分法(Halving Algorithm)

- クイズ王(エキスパート) N人のうち少なくとも1人は全問正解すると仮定
- 二分法:
「これまで一度も間違えていないエキスパートたちの予測の多数決に従う」

誤り回数は高々 $\lfloor \log_2 N \rfloor$

二分法の解析



- 間違えたとき，候補は半分以下
- 誤り回数が m ⇒ 候補数 $\leq N / 2^m$　$m \leq \lfloor \log_2 N \rfloor$
- 仮定より，　$1 \leq$ 候補数

二分法 ∈ 乗算型重み更新

- $v_i \in [0,1]$: エキスパート i の重み
- 初期重み $v_i = 1/N$
- $x_i \in \{-1,1\}$: エキスパート i の予測値
- 二分法の出力：重みつき多数決
　$\mathrm{sgn}(\sum_{i=1}^{N} v_i x_i \geq 0)$
- 重み更新
　$v_i \leftarrow v_i b_i$ / 規格化

ただし,
　$b_i = 1$ ・・・ エキスパート i が正しかったとき
　　0 ・・・ エキスパート i が間違えたとき

重みつき平均アルゴリズム
—— 全問正解者がいない場合 ——

- 重み更新

  $v_i \leftarrow v_i\ b_i$ / 規格化

  ただし,

  $b_i = 1$ ・・・ エキスパート i が正しかったとき

  0 $e^{-\eta}$ ・・・ エキスパート i が間違えたとき

誤り回数 $\leq m^* + O\left(\sqrt{m^* \ln N} + \ln N\right)$

m^* は最も成績の良かったエキスパートの誤り回数

パーセプトロン ∈ 加算型重み更新

- 学習目標 $v^* = (0, ..., 1, ..., 0)$ の線形分離関数（シングルトン）の学習
- パーセプトロンの出力：重みつき多数決

  $y' = \mathrm{sgn}(\Sigma_{i=1}^{N} v_i x_i \geq 0)$
- 重み更新

  $v_i \leftarrow v_i + \eta\,(y-y')\,x_i$

  ただし, y は真の値, η は学習定数
- 誤り回数 = $\Omega(N)$　cf. 二分法は $\lfloor \log N \rfloor$

パーセプトロン vs Winnow
加算型重み更新　乗算型重み更新

- 学習目標が $v^* = (1, 0, 1, 0, 0, ..., 1)$ のとき

  ただし, $k = \Sigma_i v^*_i$
- パーセプトロン

  $v_i \leftarrow v_i + \eta\,(y-y')\,x_i$

  誤り回数 = $\Omega(kN)$

  カーネル手法と整合する
- Winnow

  $v_i \leftarrow v_i \exp(\eta\,(y-y')\,x_i)$ / 規格化

  誤り回数 = $O(k \log N)$　・・・ attribute efficient

  一般に, カーネル手法と整合しない

この問題と解析の特徴

- オンライン性
  - 予測と結果の提示が交互に繰り返される
- ユニバーサル性
  - エキスパートの予測や結果の系列に対して何の仮定もおかない
- 最悪値評価
  - アルゴリズムの性能を最悪の場合で評価する
- 相対評価
  - アルゴリズムの性能を, 最適なエキスパートの成績を用いて相対的に評価する

オンライン予測

- モデル化
  - エキスパート統合モデル
- 統合アルゴリズム
  - Winnow (1988)
  - Aggregating Algorithm (1990)
  - Weighted Majority, Weighted Averaging (1993,1994,1999)
  - GD, EG family (1994)
  - Potential-Based (2001)
  - Following Perturbed Leader (2003)
- 応用
  - 学習, 符号化, 統計的推定, 投資, ゲーム理論, 最適化, …

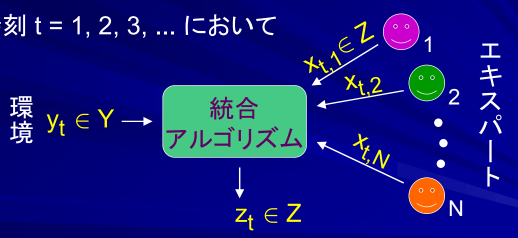
エキスパート統合モデル

Y ・・・真の結果値の集合

Z ・・・予測値の集合

L: Y × Z → [0,∞] ・・・損失関数

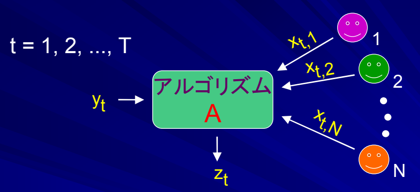
各時刻 t = 1, 2, 3, ... において



アルゴリズムの損失：　$L(y_t, z_t)$

エキスパート i の損失：$L(y_t, x_{t,i})$

相対損失(regret)



- アルゴリズム A の累積損失：$\mathrm{Loss}_A^T = \sum_{t=1}^T L(y_t, z_t)$
- エキスパート i の累積損失：$\mathrm{Loss}_i^T = \sum_{t=1}^T L(y_t, x_{t,i})$
- アルゴリズム A の相対損失：$R_A^T = \mathrm{Loss}_A^T - \min_i \mathrm{Loss}_i^T$

目標：$R^T{}_A \to$ small

クイズ王に匹敵する問題

- Y = {0,1}
- Z = {0,1}
- L(y, z) = |y − z|

に相当

$\mathrm{Loss}^T{}_A$ = アルゴリズム A の誤り回数
$\mathrm{Loss}^T{}_i$ = エキスパート i の誤り回数
m^* = $\min_i \mathrm{Loss}^T{}_i$

$$\exists A,\quad R_A^T \le 4\sqrt{m^* \ln N} + 2.5 \log_2 N$$

統合アルゴリズムの原理

Vovk の Aggregating Algorithm (AA)

- 各エキスパート i に対し，重み $v_{t,i}$ を保持
  ($v_{t,i} \ge 0$, $\sum_{1\le i\le N} v_{t,i} = 1$)
- 予測の統合

$$\forall y \in Y, L(y, z_t) \le -c(\eta) \ln \sum_{i=1}^N v_{t,i} e^{-\eta L(y, x_{t,i})}$$

を満たす $z_t \in$ Z を出力
c(η)：問題 (Y, Z, L) に固有の値

- 重み更新

$$v_{t+1,i} = \frac{v_{t,i} e^{-\eta L(y_t, x_{t,i})}}{規格化係数}$$

AAの性能

- **定理**
  任意のエキスパート，任意の環境の振る舞いに対し，
  最適なエキスパートを $i^* = \mathrm{argmin}_i \mathrm{Loss}^T{}_i$ とすると，

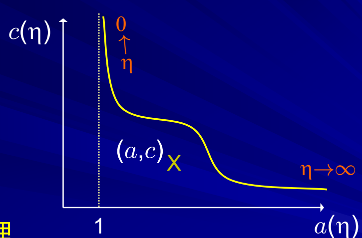
$$\mathrm{Loss}_{AA}^T \le c(\eta)\left(\eta \mathrm{Loss}_{i^*}^T + \ln \frac{1}{v_{1,i^*}}\right)$$

特に，初期重みが一様 ($v_{1,i}$ = 1/N) のとき，

$$\mathrm{Loss}_{AA}^T \le c(\eta)\left(\eta \mathrm{Loss}_{i^*}^T + \ln N\right)$$
$$\equiv a(\eta)\mathrm{Loss}_{i^*}^T + c(\eta) \ln N$$

AAの最適性

$$\mathrm{Loss}_{AA}^T \le a(\eta)\mathrm{Loss}_{i^*}^T + c(\eta) \ln N$$

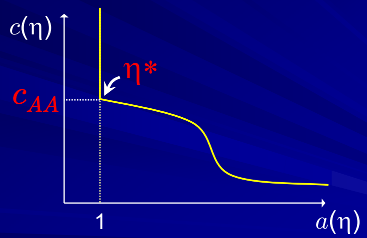


- **定理**
  問題 (Y, Z, L) が"自然"ならば，
  ∀A, ∃エキスパート, ∃環境

$$\mathrm{Loss}_A^T > a\mathrm{Loss}_{i^*}^T + c \ln N$$

凸損失関数に対する性質

L(y,z) が z に関して下に凸のとき
（2次損失，相対エントロピー損失，Hellinger損失，. . . ）



$$\mathrm{Loss}^T_{AA} \le \mathrm{Loss}^T_{i^*} + c_{AA} \ln N$$

Weighted Averaging Algorithm (WAA)

- 簡便なアルゴリズム（AAの近似）
- Y = Z = [0,1]，L ：凸損失関数と仮定
- 予測の統合

$$z_t = \sum_{i=1}^{N} v_{t,i} x_{t,i}$$

- 定理

$$\mathrm{Loss}^T_{WAA} \le \mathrm{Loss}^T_{i^*} + c_{WAA} \ln N$$

$$(c_{AA} \le c_{WAA})$$

さまざまな凸損失関数と c_{AA}，c_{WAA}

$\mathrm{Loss}^T_{AA} \le \mathrm{Loss}^T_{i^*} + c_{AA} \ln N$

$\mathrm{Loss}^T_{WAA} \le \mathrm{Loss}^T_{i^*} + c_{WAA} \ln N$

損失関数	$L(y, z)$	c_{AA}	c_{WAA}
2次損失	$(y - z)^2$	1/2	2
相対エントロピー損失	$(1-y)\ln\frac{1-y}{1-z} + y \ln \frac{y}{z}$	1	1
Hellinger損失	$\frac{1}{2}((\sqrt{1-y}-\sqrt{1-z})^2 + (\sqrt{y}-\sqrt{z})^2)$	$1/\sqrt{2}$	1
絶対値損失	$\lvert y - z \rvert$	凸関数ではない	

ダイバージェンスを用いた導出

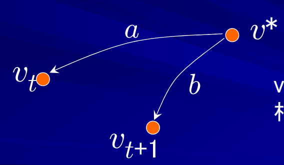
- KLダイバージェンス（相対エントロピー）
  確率ベクトル v，u に対し，
  $\mathrm{RE}(v \parallel u) = \sum_i v_i \ln (v_i / u_i)$
- 重み更新の導出
  $v_{t+1} = \mathrm{argmin}_v \left(\eta \sum_i v_i L(y_t, x_{t,i}) + \mathrm{RE}(v \parallel v_t)\right)$

$$= \frac{v_{t,i} e^{-\eta L(y_t, x_{t,i})}}{\text{規格化係数}}$$

上界の導出

- $v^* = (0, ..., 1, ..., 0) \in [0,1]^N$ （第 i^* 成分のみ1）

$$\boxed{\begin{aligned} &\mathrm{L}(y_t, z_t) - \mathrm{L}(y_t, x_{t,i^*}) \\ &\quad \le c\,(\mathrm{RE}(v^* \parallel v_t) - \mathrm{RE}(v^* \parallel v_{t+1})) \end{aligned}}$$



v_t が v^* に“時刻 t における”相対損失分以上近づく

- t=1, ..., T で和を取ると，
  相対損失 $\le$ c $\mathrm{RE}(v^* \parallel v_1)$ = c ln N

オンライン予測の応用

オンラインアルゴリズムの統合

- オンライン問題を解くアルゴリズム $A_1, \ldots, A_N$
- 入力系列 $S = y_1, \ldots, y_T$

$\max_S \min_i A_i(S)$ の損失 $\leq \min_i \max_S A_i(S)$の損失

AA の損失 $\leq \min_i A_i$の損失 + $c_{AA} O(\ln N)$

確率モデルの統合と情報圧縮

・・・ so she was considering in her __?__

⇓

言語モデル θ_i

⇓

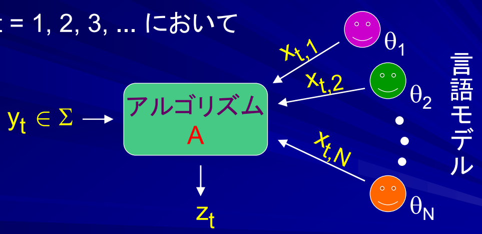
	Pr(· | context, θ_i)
mind	0.232
thought	0.183
own	0.129
office	0.073
:	:

確率モデルの統合モデル

$\Sigma = \{1, \ldots, K\}$: アルファベット

$y_1, y_2, \ldots, y_{t-1} \in \Sigma \longrightarrow \boxed{\theta_i} \longrightarrow x_{t,i}(\cdot) = \Pr(\cdot \mid y_1, \ldots, y_{t-1}, \theta_i)$

各時刻 t = 1, 2, 3, ... において



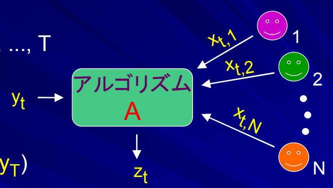
アルゴリズムの尤度: $z_t(y_t) = \Pr(y_t \mid y_1, \ldots, y_{t-1}, A)$

モデル θ_i の尤度: $x_{t,i}(y_t) = \Pr(y_t \mid y_1, \ldots, y_{t-1}, \theta_i)$

ユニバーサル符号と冗長度

t = 1, 2, ..., T

系列 S = ($y_1, \ldots, y_T$)



- Aの尤度: $P(S \mid A) = \prod_t P(y_t \mid y_{1..t-1}, A) = \prod_t z_t(y_t)$
- θ_i の尤度: $P(S \mid \theta_i) = \prod_t x_{t,i}(y_t)$
- A + ユニバーサル符号 ⇒ 符号長 = $-\log_2 P(S \mid A)$
  θ_i + ユニバーサル符号 ⇒ 符号長 = $-\log_2 P(S \mid \theta_i)$
- 冗長度 = $-\log_2 P(S \mid A) - \min_i (-\log_2 P(S \mid \theta_i))$

AAの適用とベイズ混合

以下の問題 (Y, Z, L) と等価

- $Y = \Sigma = \{1, \ldots, K\}$
- $Z = \{ z \in [0,1]^K : \Sigma$ 上の分布 $\}$
- $L(y, z) = -\ln z(y)$ ・・・ 対数損失
  ⇒ $\mathrm{Loss}^T_A = -\ln P(S \mid A)$ ・・・ 符号長
- AA の出力 = WAA の出力 = ベイズ混合

$$\begin{aligned} z_t(j) &= \textstyle\sum_{i=1}^N v_{t,i} x_{t,i}(j) \\ &= \textstyle\sum_{i=1}^N \Pr(\theta_i \mid y_{1..t-1}) \Pr(j \mid y_{1..t-1}, \theta_i) \end{aligned}$$

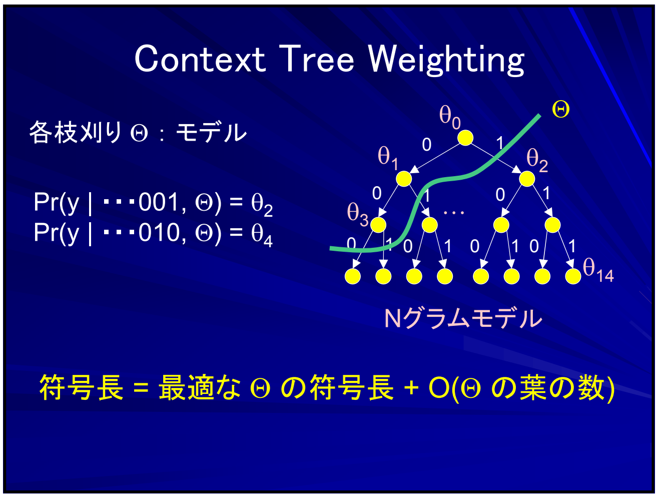
$\boxed{\text{冗長度 = 相対損失 = } \mathrm{Loss}^T_A - \mathrm{Loss}^T_{i^*} \leq \ln N}$

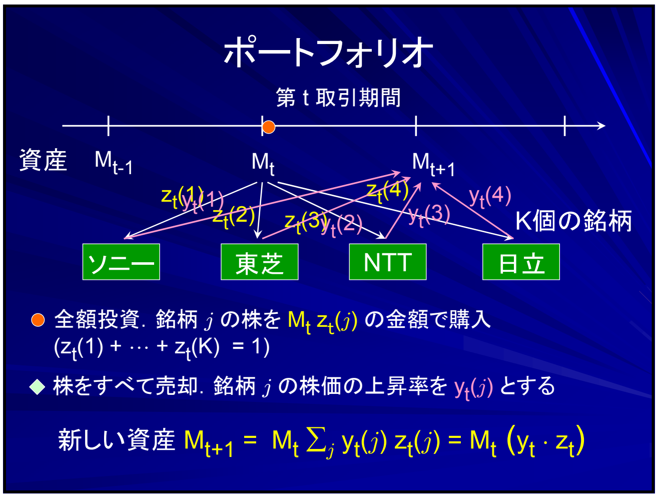
無限のエキスパートの統合

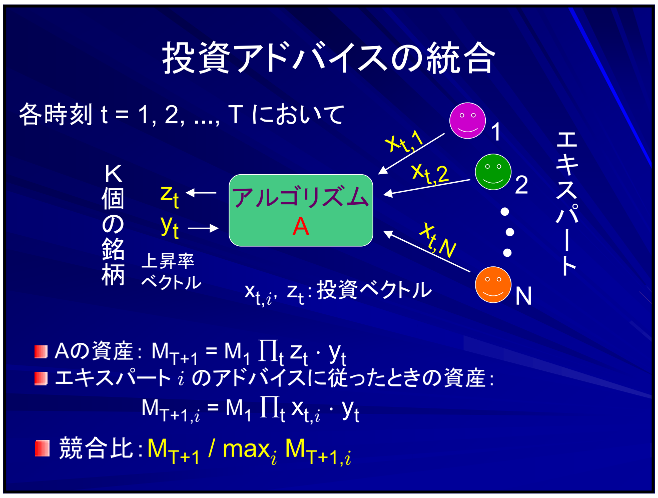
- Y = {0,1}
- エキスパート $\theta \in [0,1]$
  $x_{t,\theta}(1) = \Pr(1 \mid y_{1..t-1}, \theta) = \theta$
  $x_{t,\theta}(0) = \Pr(0 \mid y_{1..t-1}, \theta) = 1 - \theta$
- AAの出力 = KT-Estimator

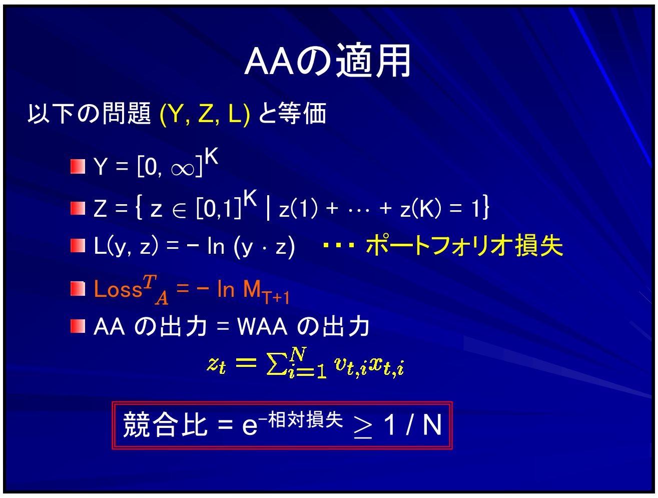
$$\begin{aligned} z_t(y) &= \int v_{t,\theta} x_{t,\theta}(y) d\theta \\ &= \frac{\sum_{s=1}^{t-1} y_s + 1/2}{t} \end{aligned}$$

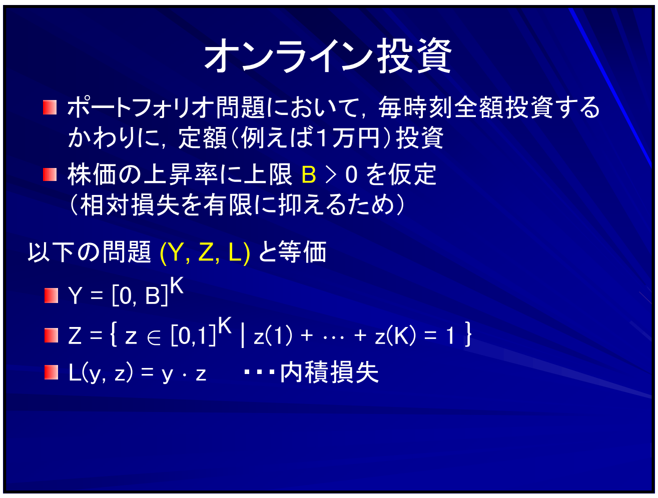
- 相対損失 = O(log T)

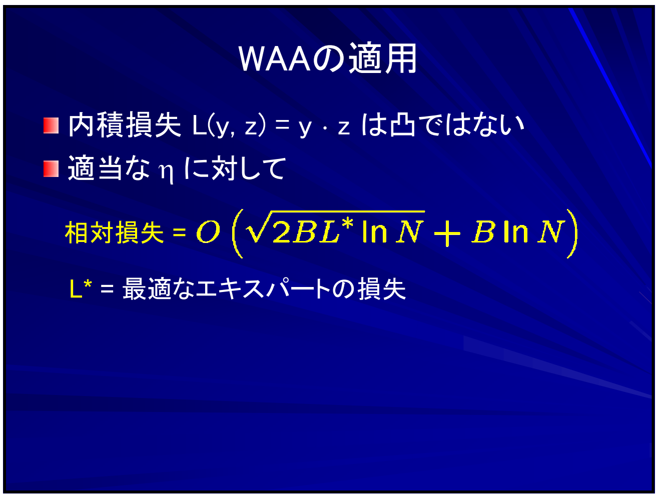

Context Tree Weighting
各枝刈り Θ : モデル
Pr(y | ･･･001, Θ) = θ_2
Pr(y | ･･･010, Θ) = θ_4
θ_0
θ_1
θ_2
θ_3
θ_{14}
Θ
Nグラムモデル
符号長 = 最適な Θ の符号長 + O(Θ の葉の数)


ポートフォリオ
第 t 取引期間
資産 M_{t-1} M_t M_{t+1}
$z_t(1)$ $z_t(2)$ $z_t(3)$ $z_t(4)$
$y_t(1)$ $y_t(2)$ $y_t(3)$ $y_t(4)$
K個の銘柄
ソニー 東芝 NTT 日立
全額投資. 銘柄 j の株を $M_t z_t(j)$ の金額で購入
$(z_t(1) + \cdots + z_t(K) = 1)$
株をすべて売却. 銘柄 j の株価の上昇率を $y_t(j)$ とする
新しい資産 $M_{t+1} = M_t \sum_j y_t(j)\, z_t(j) = M_t\, (y_t \cdot z_t)$


投資アドバイスの統合
各時刻 t = 1, 2, ..., T において
K個の銘柄
z_t
y_t
上昇率ベクトル
アルゴリズム A
$x_{t,1}$ $x_{t,2}$ $x_{t,N}$
1 2 N
エキスパート
$x_{t,i}$, z_t : 投資ベクトル
Aの資産 : $M_{T+1} = M_1 \prod_t z_t \cdot y_t$
エキスパート i のアドバイスに従ったときの資産 :
$M_{T+1,i} = M_1 \prod_t x_{t,i} \cdot y_t$
競合比 : M_{T+1} / $\max_i M_{T+1,i}$


AAの適用
以下の問題 (Y, Z, L) と等価
$Y = [0, \infty]^K$
$Z = \{ z \in [0,1]^K \mid z(1) + \cdots + z(K) = 1\}$
$L(y, z) = -\ln (y \cdot z)$ ･･･ ポートフォリオ損失
$\mathrm{Loss}^T_A = -\ln M_{T+1}$
AA の出力 = WAA の出力
$z_t = \sum_{i=1}^N v_{t,i} x_{t,i}$
競合比 = $e^{-相対損失} \geq 1 / N$

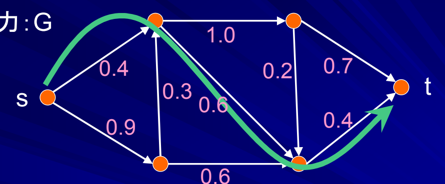

オンライン投資
ポートフォリオ問題において，毎時刻全額投資するかわりに，定額（例えば1万円）投資
株価の上昇率に上限 B > 0 を仮定
（相対損失を有限に抑えるため）
以下の問題 (Y, Z, L) と等価
$Y = [0, B]^K$
$Z = \{ z \in [0,1]^K \mid z(1) + \cdots + z(K) = 1 \}$
$L(y, z) = y \cdot z$ ･･･内積損失


WAAの適用
内積損失 $L(y, z) = y \cdot z$ は凸ではない
適当な η に対して
相対損失 = $O\left(\sqrt{2BL^* \ln N} + B \ln N\right)$
L^* = 最適なエキスパートの損失

Path Kernel 法

オンライン最短路問題

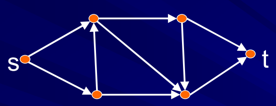
入力：G



1. あるs-t 道 P を（確率的に）選ぶ．この確率を $z_t(P)$ とする．
2. 各辺の遅延時間 $d_t(e) \in [0,1]$ が明らかとなる．
   道Pに沿って s から t に到達する時間は $y_t(P) = \sum_{e \in P} d_t(e)$
3. 到達時間の期待値 $\sum_P y_t(P)\, z_t(P) = y_t \cdot z_t$

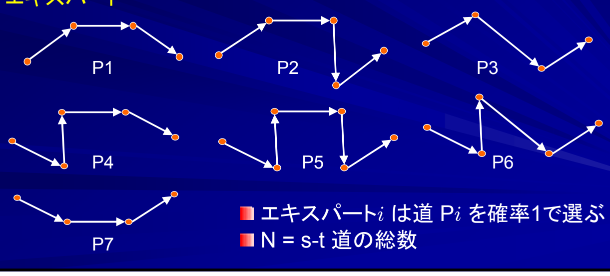
目標：$\sum_t y_t \cdot z_t - \min_P \sum_t y_t(P) \rightarrow$ 小

オンライン投資への還元



- $B = \max_P |P|$
- $Y = [0,B]^N$
- Z = s-t道上の分布の集合
- $L(y, z) = y \cdot z$

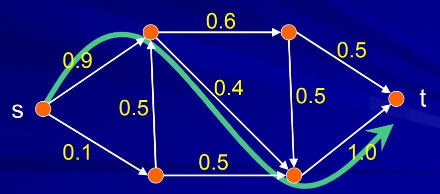
エキスパート



- エキスパートi は道 Pi を確率1で選ぶ
- N = s-t 道の総数

Path Kernel 法

- N（道の総数）は指数的になりうる
- すべての道 P に対し，重み $v_{t,P}$ を保持できない
- その代わり，各辺 e に対し，遷移確率 $a_t(e)$ を保持
- 関係式 $v_{t,P} = \prod_{e \in P} a_t(e)$ によって間接的に $v_{t,P}$ を表現



$v_{t,P} = 0.9 * 0.4 * 1.0 = 0.36$

重み更新の模倣

条件：

（更新前）　$v_{t,P} = \prod_{e \in P} a_t(e)$

$$v_{t+1,P} = \frac{v_{t,P}\, e^{-\eta y_t(P)}}{\text{normalize}}$$

（更新後）　$v_{t+1,P} = \prod_{e \in P} a_{t+1}(e)$

を満たすような辺上の重み更新

$$a_t \rightarrow a_{t+1}$$

を行う必要がある

重み更新の模倣

- 任意の P に対し，次の式を満たす a_{t+1} を求めればよい

$$v_{t+1,P} = \prod_{e \in P} a_{t+1}(e) = \frac{\prod_{e \in P} a_t(e) b_t(e)}{\sum_P \prod_{e \in P} a_t(e) b_t(e)}$$

ただし，$b_t(e) = \exp(-\eta\, d_t(e))$

道 P の損失 = 辺の損失の和
($L(y_t, x_{t,P}) = \sum_{e \in P} d_t(e)$)

Path Kernels

- 辺の重みベクトル: **a** = (a(e_1), ..., a(e_m))
- パスの重みベクトル: **v** = (v_{P1}, ..., v_{PN}),,

$$\forall P, v_P = \prod_{e \in P} a(e)$$

この関係を, **v** = Φ(**a**) と表す.

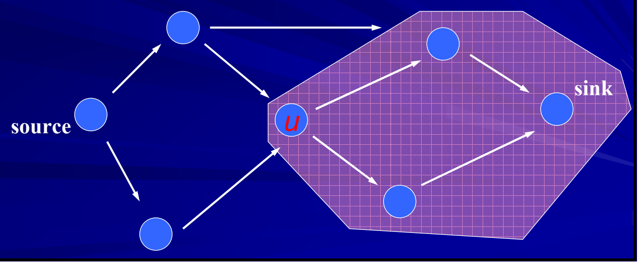
- 係数ベクトル : **b** = (b(e_1), ..., b(e_m))

Path Kernel

$$K(\boldsymbol{a}, \boldsymbol{b}) = \Phi(\boldsymbol{a}) \cdot \Phi(\boldsymbol{b}) = \underbrace{\sum_P \prod_{e \in P} a(e) b(e)}_{\text{normalization factor}}$$

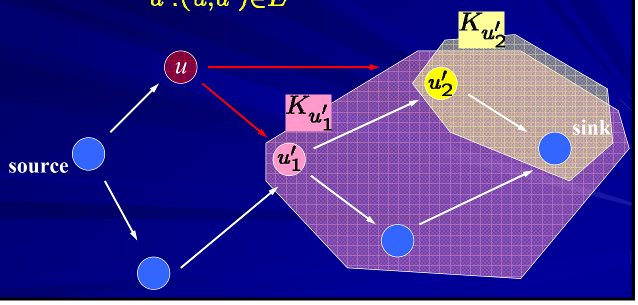
Path Kernel の計算

$$K_u = \sum_{P:\text{from } u} \prod_{e \in P} a(e) b(e)$$



Path Kernel の計算

- $K_{\text{sink}} = 1$
- $K_u = \sum_{u':(u,u') \in E} v(u,u') b(u,u') K_{u'}$

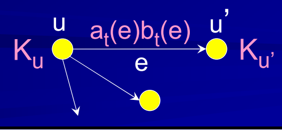


Weight Pushing Algorithm

- 目標 $\forall P, \prod_{e \in P} a_{t+1}(e) = \dfrac{\prod_{e \in P} a_t(e) b_t(e)}{K_{\text{source}}(\boldsymbol{a}_t, \boldsymbol{b}_t)}$
- 漸化式 $K_u = \sum_{u':(u,u') \in E} a_t(u,u') b_t(u,u') K_{u'}$

$$\boxed{a_{t+1}(e) = \frac{a_t(e) b_t(e) K_{u'}}{K_u}}$$

目標達成！



まとめ

- さまざまなオンライン最適化問題が, 共通の統合スキーム(AA, WAA)を用いて解ける
- 一般の最適化問題への適用
  - 各時刻 t における損失が, それまでの履歴に依存する
  - オフラインで最適な解を求めるのが困難
- 構造的エキスパートの統合の可能性
  - 決定リストの効率の良いオンライン学習
- モデルのさらなる拡張
  - リスク情報を用いた統合

平成17年度 第2回 全体会議

日時　平成17年11月21日(月), 22日(火)

会場　名古屋大学 野依記念学術交流館 / IB電子情報館 中棟 IB大講義室

プログラム

11月21日(月)

10:30 – 12:00　未解決問題セッション

Domain-Specific Image Segmentation and Shape Matching
浅野哲夫 (北陸先端科学技術大学院大学)

柔体を扱う計算幾何学
浅野哲夫 (北陸先端科学技術大学院大学)

線的施設配置問題
加藤直樹 (京都大学)

外面が四角形(以上)である格子凸描画を求める線形時間アルゴリズムの開発
三浦一之 (福島大学)

幾何的巡回セールスマン問題の厳密アルゴリズムについて
岡本吉央 (豊橋技術科学大学)

文字列検索における時間と領域のトレードオフ
定兼邦彦 (九州大学)

12:00 – 13:30　幹事会

13:30 – 15:50　未解決問題に対するグループ討論

16:00 – 17:30　全体会議

11月22日(火)

16:30 – 17:30　未解決問題グループ討論の結果報告

透過的データ圧縮
Transparent Data Compression

K. Sadakane and R. Grossi: Squeezing Succinct Data Structures into Entropy Bounds, Proc. ACM-SIAM SODA 2006, to appear.

九州大学システム情報科学研究院
定兼　邦彦
2005年11月22日

背景

- データ圧縮の目的 (過去)
  - ディスク容量の節約
  - 通信コスト(料金，時間)の削減

保存用

- データ圧縮の目的 (現在)
  - アクセスの高速化 (CPU速度 > ディスク速度)
  - 連続的なアクセスに限られる
- ランダムアクセスができたらどうなるか

2

透過的データ圧縮

もし圧縮データの任意部分を高速に復元できれば

- データを圧縮したまま保存できる
  - ディスク容量の節約
  - 高速なアクセスが可能
- 圧縮されていることを意識しなくていい

3

本研究の結果

- 長さ n の文字列 S を圧縮 (アルファベットサイズσ)
- サイズ: LZ78 [Ziv, Lempel 78] と漸近的に同じ

$$nH_k + \mathrm{O}\left(\frac{n(\log\sigma + \log\log_\sigma n + k)}{\log_\sigma n}\right) \text{ bits}$$

(H_k: S の k 次経験的エントロピー)

- S の i 文字目からの連続する $\log_\sigma n$ 文字 ($\log n$ ビット) を定数時間で復元可能 ($decode(S,i)$) (計算モデル: word RAM (語長 $\log n$ ビット))
- このアクセス時間は未圧縮の場合と同じ

4

研究の動機

- Succinctデータ構造を更に圧縮したい
  - bit vector
  - 検索木
  - グラフ
  - 圧縮接尾辞木

5

Succinctデータ構造

- あるデータ集合 D を格納するデータ構造
- データ構造のサイズ: 情報理論的下限に近い
  - 情報理論的下限 $L = \log$ (D の場合の数)
- 高速な問合せが可能
  - 補助的なデータ構造 (索引) を使用
  - サイズ: 漸近的に無視できる (o(L) bits)

6

集合のSuccinctデータ構造

- 集合 $D \subset \{1,2,...,n\}$ を格納
- 問合せ (定数時間)
  - *member*(D,i): D 中に i が存在するか
  - *rank*(D,i): D 中の i 以下の要素の数
  - *select*(D,j): D 中で j 番目に小さい要素
- サイズ: $n + o(n)$ bits [J89] [M96]

1　　　i　　　n

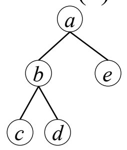
S: 0100011000100100001

rank$(D,i) = 3$

7

木のSuccinctデータ構造

- n 節点の順序木 T を格納
  - 通常のデータ構造は $O(n \log n)$ bits
- T は $\frac{1}{n+1}\binom{2n}{n} \approx 4^n$ 通り存在 (Catalan数)
  - 情報理論的下限 $= 2n - \Theta(\log n)$ bits
- 問合せ(定数時間)
  - 長男，兄弟，親，深さ，preorder，子孫の数など
- サイズ: $2n + o(n)$ bits [MR01] [GRR04]



a b c d e

S ((( ) ( )) ( ))

木を括弧列 S で表現

8

問: Succinctデータ構造は それ以上圧縮できないか?

- 答: 場合によっては可能
- 例: 集合 $D \subset \{1,2,...,n\}$ の要素数が少ない場合
- 要素数が m の集合は $\binom{n}{m}$ 通り

$B(m,n) = \log\binom{n}{m} \approx m \log\frac{n}{m} + (n-m)\log\frac{n}{n-m}$ bitsで表現できるはず

- $B(m,n) + O(n \log\log n / \log n)$ bitsのデータ構造で*member*, *rank*, *select*を定数時間で返すものが存在 [RRR02]

FID (Fully Indexable Dictionary) と呼ばれる

9

FIDを用いたポインタの圧縮

ある n ビットのデータ構造(ビット列)へのポインタ m個を格納する場合

- ポインタの値 i を $S[i] = 1$ で表現
- S をFIDで圧縮．ポインタは*select*で求まる
- $m = O(n/\log n)$ のとき，FIDのサイズは

$$\log\binom{n}{O(n/\log n)} = O\left(\frac{n}{\log n}\log\frac{n}{n/\log n}\right) + O\left(\frac{n\log\log n}{\log n}\right) = O\left(\frac{n\log\log n}{\log n}\right)$$

ビット列

S 0000000100000000000001000000000000010000000000000100000000

10

問: 木のSuccinctデータ構造は 圧縮できるか?

- FIDでは不可能
  - (と) が同数ある ⇒ $B(n,2n) = 2n$ bits 必要
  - $2n + O(n \log\log n/\log n)$ bits [GRR04]
- データ間の相関を考慮する必要がある
  - FIDでは 0 次のエントロピーまで圧縮

$$B(m,n) = m\log\frac{n}{m} + (n-m)\log\frac{n}{n-m} = nH_0$$

  - k 次のエントロピーまで圧縮したい

11

本研究の圧縮法の応用

- 集合 $D \subset \{1,2,...,n\}$ に対する*member*, *rank*, *select*を定数時間で返すデータ構造
- サイズ: $nH_k + O(n \log\log n/\log n)$ bits ($k = O(\log\log n)$)
  - H_kは D を表す0,1列 S の k 次経験的エントロピー
- EID (Entropy-Bound Indexable Dictionary) と呼ぶ
- FID よりもさらに小さい

$$nH_k \le nH_{k-1} \le \cdots \le nH_0 \approx B(m,n)$$

12

EIDのデータ構造とアルゴリズム

- データ構造
  - D を表す0,1列 S を圧縮したもの $nH_k + O(n \log\log n/\log n)$ bits (任意の連続する $\log n$ ビットを定数時間で復元可)
  - FIDの補助データ構造 $O(n \log\log n/\log n)$ bits
- 問合せアルゴリズム
  - FIDとほぼ同じ ($S[i,i+\log n-1]$ へのメモリ参照を $decode(S,i)$に置き換える)
- どんな問合せの計算量も漸近的には同じ

13

木のSuccinctデータ構造の圧縮

- FIDでは不可能
  - $2n + O(n \log\log n/\log n)$ bits [GRR04]
- EIDでは可能
  - 木を表現する括弧列 S の H_k まで圧縮可 (k=O(log log n))
  - $2nH_k + O(n \log\log n/\log n)$ bits
  - 問合せの計算量は圧縮前と同じ
  - 構造が同じ部分木があると H_k は小さくなる (接尾辞木で特に有効)

14

EIDの圧縮サイズの限界

- S = 010101...010101 を圧縮する場合
  - FID: $nH_0 = n$ bits (+ $O(n \log\log n/\log n)$)
  - EID: $nH_1 = O(\log n)$ bits (+ $O(n \log\log n/\log n)$)
- エントロピーが小さいと第2項が無視できない
- *rank* を定数時間で返すデータ構造のサイズは $\Omega(n \log\log n/\log n)$ bits [Miltersen 05] つまり第2項は最適

15

従来の圧縮法

16

従来の圧縮法

辞書式圧縮法

LZ77 [Ziv, Lempel 77]

LZ78 [Ziv, Lempel 78]

LZW[Welch 84]

compress

LZSS [Storer, Szymanski 82]

gzip

統計的圧縮法

PPM[Cleary, Witten 84]

PPMD [Howard 93]

PPM*[Cleary, Teahan, Witten 95]

block sorting

[Burrows, Wheeler 94]

高圧縮率、PPMより高速 (bzip2)

context tree weighting

[Willems, Shtarkov, Tjalkens 95]

PPMより高圧縮率

17

LZ77圧縮法

- 文字列を辞書へのポインタで置き換える
- 辞書 = すでに圧縮した文字列

....a compressed suffix tree consists of a compressed suffix array, a Pat tree and edge-length information.

....a compressed suffix tree consists of [l=19, d=36] array, a Pat [l=4, d=51] and edge-length information.

圧縮率は文字列のエントロピーに収束

18

PPM

- 文字を1文字ずつ符号化
- 各文字の符号は文脈から決まる
  – 文脈：符号化する文字の直前の k 文字
- 圧縮サイズ：$nH_k = \sum_{s\in A^k} n_s \sum_{c\in A} p_c \log\frac{1}{p_c}$

abcababc

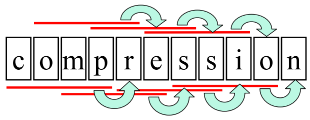
文脈　符号化する文字

19

高次圧縮の問題点

- k 次のエントロピーの圧縮率を達成するには隣接する文字列の情報を用いて圧縮する必要がある
- 一部分を復元する場合も全体を復元ことになる

$k=4$



20

従来の圧縮法との比較

	漸近的圧縮率	$\log n$ ビットの復号時間
LZ77 [ZL77]	nH_k	$O(n)$
LZ78 [ZL78]	nH_k	$O(n)$
PPM [CW84]	nH_k	$O(n)$
CTW [WST95]	nH_k	$O(n)$
Block Sorting [BW94]	nH_k	$O\left(\frac{\log^2 n}{\log\log n} + \log n \cdot \log\sigma\right)$ [GGV03]
本研究	nH_k	$O(1)$

21

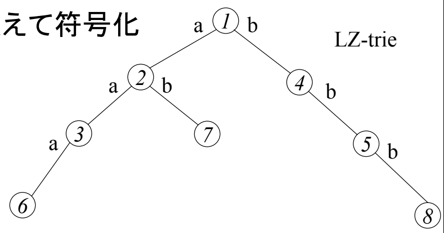
新圧縮法の概要

LZ78の改良+補助データ構造

22

LZ78 圧縮法

- 文字列を辞書中のフレーズに分割
- 数字に置き換えて符号化
- 辞書を更新



$S =$ aaabbbaaaabbbb
②③④⑤ ⑥ ⑦ ⑧

出力　①a②a①b④b③a②b⑤b

23

LZ78の圧縮率　[Ziv, Lempel 78]

長さ n の文字列 S を分解したフレーズの数 c は

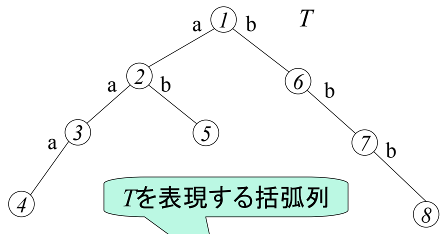
$$\sqrt{n} \le c \le \frac{n}{\log_\sigma n} \quad (\sigma：アルファベットサイズ)$$

- 圧縮後のサイズ：$c(\log c + \log\sigma)$ bits
- S が定常エルゴード情報源 (エントロピー H) から生成されるなら $\frac{c\log c}{n} \to H \quad (n\to\infty)$
- S の k 次経験的エントロピーH_kに対し

$$c\log c \le nH_k + c\log\frac{n}{c} + \Theta(kc+c) \quad \text{[Kosaraju, Manjini 99]}$$

24

本研究のデータ構造



*T*を表現する括弧列

S = aaabbbaaaabbbb

R ②③⑥⑦ ④ ⑤ ⑧

P 1 2 4 5 7 10 12

フレーズの添字を格納
添字が T のpreorderと一致するように付け替える

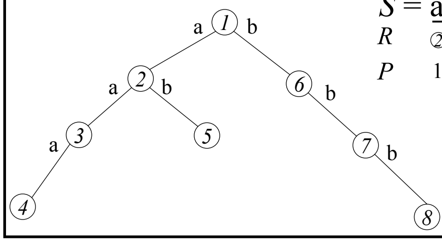
*T*の枝ラベル

フレーズの開始位置を格納

*decode*アルゴリズム

$S[i,i+\log n-1]$ を復号する場合

1. $S[i]$ を含むフレーズ p を求める
2. p を表すLZ-trieのノード v を求める
3. v から根に向かうパス上のラベルを復号



S = aaabbbaaaabbbb

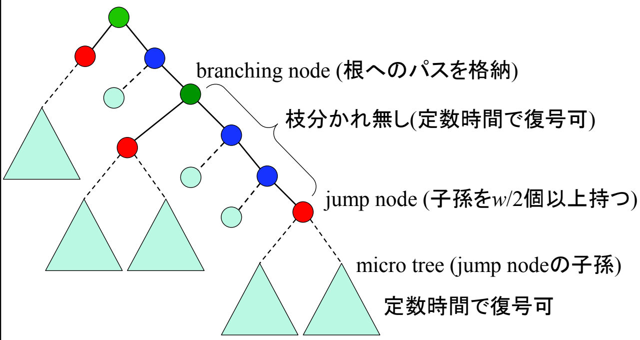
R ②③⑥⑦ ④ ⑤ ⑧

P 1 2 4 5 7 10 12

26

Long phraseの復号

- Long phrase: 長さ $w = ½ \log_\sigma n$ 以上のフレーズ



27

Short phraseの復号

- Short phrase: 長さ $w = ½ \log_\sigma n$ 未満のフレーズ
- Sの長さ ½ $\log n$ の部分列はshort phraseを $O(\log n)$ 個含む可能性あり ⇒ 定数時間で復号できない
- $r > 1$ 個の連続するshort phraseをそのまま格納
  - 対応する R は格納しない
- データ構造のサイズは増加しない
  - R を格納する場合: $r \log c$ bits
  - そのまま格納する場合: ½ $\log n$ bits
  - $½ \log n < r \log c \quad (\because \sqrt{n} \le c)$

28

まとめ

新圧縮法の提案

- 任意の文字列を高次エントロピー限界まで圧縮
- 部分列の定数時間復号 (未圧縮と同じ時間)

応用

- 既存の索引構造のサイズを改善
- 検索速度も同じ
- 個々の索引ごとに圧縮法を考える必要がない

課題

- 文字列の更新
- LZ77などの改良

29

Nov. 22, 2005

Statistical-Mechanical Approach to Probabilistic Information Processing

計算限界

確率伝播法とその可能性について[1]

渡辺 治（東京工業大学）

本日の話題

1. 確率伝播法 (Belief Propagation Method)
2. メッセージ伝播アルゴリズムの特徴と可能性について
3. アルゴリズムの平均的解析のモデル (Planted Solution Model)

例題として：グラフの二分割問題 Graph Bisection Problem

[1]このファイルは発表時に使用したプレゼンテーション用ファイルをＡ４版に変更したものです．スタイル等が悪くて見にくい部分がありますがご勘弁下さい．

1

1．確率伝播法 (Belief Propagation Method) について

一言でいうと

確率変数 $X_1,...,X_n$（各々の定義域を $A_1,...A_n$ とする）により決まる系 G において，次のような周辺確率を求める方法である．

$$p(a) \;=\; \Pr_G[X_1=\boxed{a}] \;=\; \sum_{a_2\in A_2,\;...,\;a_n\in A_n} \Pr[X_1=\boxed{a}\wedge X_2=a_2\wedge\cdots X_n=a_n].$$

ただし

確率変数間の依存関係が単純で，同時確率分布を分解して表わすことができる場合に限る $\Longrightarrow$ ベイジアン・ネットワーク

歴史

- AI の分野で Pearl が 1982 年に提唱したもの．
- 統計力学の研究者 MacKay らが LDPC 符号の復号法に使用して注目を集めた．それ以後，統計力学&情報理論を中心に研究が進んでいる．
- 最近では survey propagation なるものも登場している．

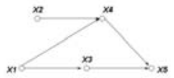
上記の周辺確率よりも，もっと一般的には次のような確率を求める（求めたい）ことが多い．

$$\Pr_G[X_1=a\,|\,X_n=b] = \frac{\displaystyle\sum_{a_2\in A_2,\;...,\;a_{n-1}\in A_{n-1}} \Pr[X_1=\boxed{a}\wedge X_2=a_2\wedge\cdots X_n=b]}{\displaystyle\sum_{a_1\in A_1,\;a_2\in A_2,\;...,\;a_{n-1}\in A_{n-1}} \Pr[X_1=a_1\wedge X_2=a_2\wedge\cdots X_n=b]}$$

2

ベイジアン・ネットワーク：

同時確率分布の分解法を示した確率変数間の依存グラフ



$$p(a_1,a_2,a_3,a_4,a_5) \;=\; p(a_1)p(a_2)p(a_3|a_1)p(a_4|a_1,a_2)p(a_5|a_3,a_4)$$

ただし，たとえば次のように略記する：

$$\Pr_G[X_2=a_2\,\&\,X_4=a_4] \;=\; p(a_2,a_4)$$

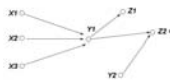
注意！（矢印がない関係はすべて独立）

$$\begin{aligned} p(a_1,a_2,a_3,a_4,a_5) &= \underline{p(a_1)}p(a_2)\underline{p(a_3|a_1)}p(a_4|a_1,a_2)p(a_5|a_3,a_4) \\ &= p(a_2)p(a_1,a_3)p(a_4|a_1,a_2)p(a_5|a_3,a_4) \\ &= p(a_1,a_2,a_3,a_4)p(a_5|a_3,a_4) \\ &= p(a_1,a_2,a_3,a_4,a_5) \end{aligned}$$

3

BP の計算方法：

確率変数間のメッセージ伝播により各確率変数の belief を計算する．



毎ステップ，各頂点において，次のような計算を行う：

以下，頂点 Y_1 と，Y_1 の取りうる値 a に注目．

（準備）

- $\pi_{Y_1}(a) = \Sigma_{\boldsymbol{b}}\ \Pr[Y_1=a|\boldsymbol{X}=\boldsymbol{b}]\cdot(\pi_{X_1\to Y_1}(b_1)\cdot\pi_{X_2\to Y_1}(b_2)\cdot\pi_{X_3\to Y_1}(b_3))$.
- $\lambda_{Y_1}(a) = \lambda_{Z_1\to Y_1}(a)\cdot\lambda_{Z_2\to Y_1}(a)$.

（メッセージの計算）

親へ（例：$Y_1 \longrightarrow Z_1$）： $\pi_{Y_1,Z_1}(a) = \pi_{Y_1}(a)\cdot\lambda_{Z_2\to Y_1}(a)$.

子へ（例：$Y_1 \longrightarrow X_1$）： $\lambda_{Y_1\to X_1}(b) \approx \Pr[X_1=b]$ の一部

（Y_1 への X_1 以外の情報により得られる値）

（現時点での $Y_1=a$ に対する belief の計算）

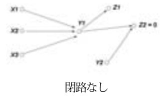
$\mathrm{Bel}_{Y_1}(a) = \alpha\cdot\lambda_{Y_1}(a)\pi_{Y_1}(a)$.

これまでにわかっていること：

定理.

1. 依存グラフに閉路がなければ正しく周辺確率を計算する．
2. その場合の計算時間は $\mathcal{O}(n)$（n は頂点数）である．

閉路あり



閉路なし

でも，経験的に… 閉路があっても大抵うまくいく場合がある！

1. LDPC 符号（Gallager 符号）の復号化
2. Turbo 符号の復号化
3. 画像修復，様々な制約充足問題

4

２．ＢＰ（メッセージ伝播法）の特徴と可能性

私の直感：

確率伝播法 $\Longleftrightarrow$ 乱択局所探索アルゴリズム

補足

- 乱択局所探索アルゴリズムにもいろいろあり，その性質も大きく異なる．上記の対応は「貪欲型」に対するもの．(もっと正確には，決定にランダム性を入れて多少ソフト化したもの．)

- 単純に乱択化しても効率が悪くなってしまう場合がある．

  比ゆ： 5 回サイコロを投げた和が 15 の元で，すべて 3 が出る確率 P_0

  $P_0 = \cdots$ 計算で求まる $\Longleftrightarrow$ $P_0 = \frac{\text{すべて 3 の回数}}{\text{和が 15 の回数}}$ 多量のサンプルが必要

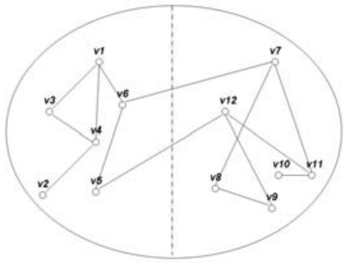
- 「貪欲型」が有効な場合に使える？(収束性がよい場合)

- もちろん（？）最悪時はダメ，平均的によい $\Longrightarrow$ 平均時解析

３．平均時解析の例

例題： グラフの二分割問題 Graph Bisection Problem

Graph Bisection Problem
Input: An undirected graph $G = (V, E)$, where $|V| = 2n$ for some n.
Task: Find an *equal size* partition V_0 and V_1 of V with min. # of cut edges.
Remark. Find one of them if there are several solutions.



知られていること：

- この問題はＮＰ-困難．
- 平均で多項式時間のアルゴリズムが知られている．
  その入力グラフのモデルとして planted solution model が導入された．

  > (Boppana, 1987): deterministic polynomial-time but not so simple
  > (Jerrum and Sorkin, 1998): randomized $\mathcal{O}(n^2)$ time
  > (Condon and Karp, 2001): randomized $\mathcal{O}(n+m)$ time
  > (F. McSherry, 1999, etc): simplified spectral method

ＢＰによるアルゴリズムの導出

確率変数
C_i $(1 \le i \le 2n)$: 頂点 v_i が V_0, V_1 のどちらに属するか？
$E_{i,j}$ $(1 \le i < j \le 2n)$: $E_{i,j} = 1 \Leftrightarrow v_i, v_j$ 間に辺が存在する．

計算の目標
与えられたグラフ G に対し，ＢＰを用いて次の周辺確率を求める．

$$p_i \;=\; \Pr[C_i = 1 \,|\, C_1 = 1 \;\wedge\; \bigwedge_{i,j} E_{i,j} = e_{i,j}]$$

$$\text{ただし，}\quad e_{i,j} \;=\; \begin{cases} 1, & G \text{ は辺 } (v_i, v_j) \text{ を持つ,} \\ 0, & G \text{ は辺 } (v_i, v_j) \text{ を持たない.} \end{cases}$$

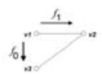
その $p_i > 0.5$ か否かで C_i を決める．

ベイジアン・ネットワーク

得られたアルゴリズム

非常に単純なメッセージ伝播アルゴリズム

program PseudoBP$(G = (V, E), p, r)$;
  set all x_i to 0;
  repeat MAXSTEP times **do** {
    $x_1 \leftarrow +\infty$;
    for each $i \in V$,
      $x_i \;\leftarrow\; \sum_{j \in N(i)} f_1(x_j) + \sum_{j \notin N(i)} f_0(x_j)$;
    if all beliefs are stabilized **then break**;
  }
  output$(x_1, ..., x_{2n})$; // Classification is (sign(x_1), ..., sign(x_{2n})).
end-program



補足：

1. 関数 f_0, f_1 の定義：

$$f_1(z) = \begin{cases} h_1 \cdot \theta_1, & \text{if } \theta_1 < z, \\ h_1 \cdot z, & \text{if } -\theta_1 \le z \le \theta_1, \\ h_1 \cdot (-\theta_1), & \text{if } z < -\theta_1; \end{cases} \quad f_0(z) = \begin{cases} -h_0 \cdot \theta_0, & \text{if } \theta_0 < z, \\ -h_0 \cdot z, & \text{if } -\theta_0 \le z \le \theta_0, \\ -h_0 \cdot (-\theta_0), & \text{if } z < -\theta_0, \end{cases}$$

   ただし，

$$a_0 = \frac{1-p}{1-r}, \;\; a_1 = \frac{p}{r}, \;\; h_0 = \left|\frac{a_0 - 1}{a_0 + 1}\right|, \;\; h_1 = \left|\frac{a_1 - 1}{a_1 + 1}\right|, \;\; \theta_0 = \left|\frac{\ln a_0}{h_0}\right|, \;\; \theta_1 = \left|\frac{\ln a_1}{h_1}\right|.$$

2. 本当のＢＰではない！
   - $\Pr[C_i = 1]$ と $\Pr[C_i = 0]$ の比の対数を求めることにした．
   - メッセージの伝播で大きな簡単化を１つ行った．

実際動かしていると非常にうまく働く． $\boxed{\text{何とか証明したい！}}$

そこで
- アルゴリズムの平均的解析 $\Longrightarrow$ Planted Solution Model
- MAXSTEP $= 2 \Leftarrow$ それでも結構よく働く

平均時計算量のモデル

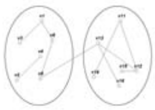
Planted Solution Model $=$ $\boxed{\text{入力グラフを生成する手法}}$

アルゴリズムの平均的解析のモデルとしてお勧め？

今回用いた Planted Solution Model

パラメータ n（頂点数 $2n$），$p, r, 0 < r < p < 1$ に対し，
次のようにグラフ $G = (V, E)$ を生成する．

1. $V_0 = \{1, ..., n\}$, $V_1 = \{n+1, ..., 2n\}$, and $V = V_0 \cup V_1$.
2. Put an edge (i, j) into E
   with probability p if $C_i = C_j$, and
   with probability r if $C_i \neq C_j$,
   where $C_i = 0 \iff i \in V_0$ and $C_i = 1 \iff i \in V_1$.



要するに
答えを作り，それを元に問題を（ランダムに）作成する方法．

$\boxed{\text{重要な性質}}$

定理. パラメータ n, p, r の間に

$$p - r = \Omega\left(\frac{1}{\sqrt{n}}\right)$$

という関係があるとき，埋め込んだ解 (planted solution) が，生成したグラフに対する Bisection Problem の唯一の解となる．

$\Downarrow$

Planted Solution Model のもとでは，アルゴリズムの正当性は，生成したグラフに対し，埋め込んだ解が正しく見出せるか否かを調べればよい．

得られた結果

定理. Planted Solution Model のもとでアルゴリズムを実行させたとき，

$$\Pr[\text{ the algorithm yields the planted solution }] \geq 1 - 2n \cdot e^{-\epsilon_1 n \cdot \frac{(p-r)^4}{p^2}}.$$

系.

For any $\delta > 0$ and any p and r, $cp < r < p < 1$,
$p - r \geq c_2 n^{-1/2} \log(1/(\delta(p-r))))$
$\Rightarrow$ the algorithm answers correctly with probability $> 1 - \delta$.

これまでのアルゴリズムとの比較：

(Boppana, 1987)	$p - r = \Omega(n^{-1/2})$ (almost!?)
(Dyer and Frieze, 1989)	$p - r = \Omega(n^{-1/2})$ when p is large, e.g., $\Omega(1)$
(Jerrum and Sorkin, 1998)	$p - r = \Omega(n^{-1/6})$
(Condon and Karp, 2001)	$p - r = \Omega(n^{-1/2+\varepsilon})$

その他にも…

- Most Likely Partition 問題（with Onsjoe）
- MAX-2SAT 問題に対する同様のアプローチ（with 山本）
- 固有値を求める手法にできないか？

未解決問題：本当のＢＰの妥当性？

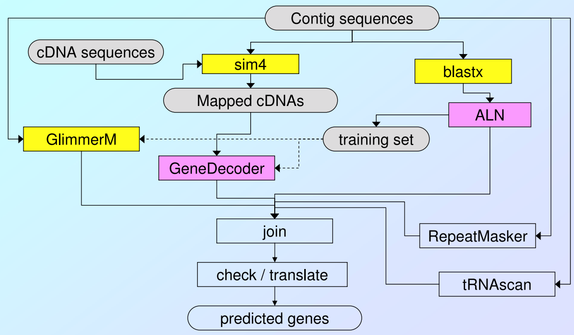
参考文献

R. McEliece, D. MacKay, and J.F. Cheng, Turbo decoding as an instance of Pearl's "Belief Propagation" algorithm, *IEEE J. on Selectead Areas in Communications*, 16(2):140–152, 1998.

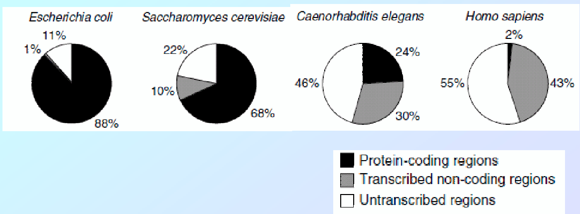
RNA配列の比較アルゴリズム

1. 東京大学大学院新領域創成科学研究科
2. 産業技術総合研究所生命情報科学研究センター
浅井　潔

麹菌
Aspergillus oryzae遺伝子発見パイプライン



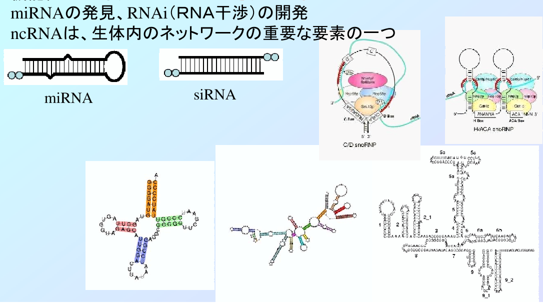
転写産物(RNA)の大部分は非コード領域 !!



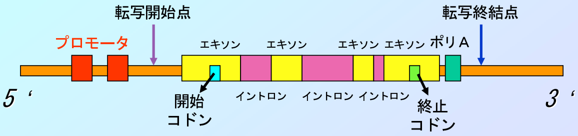
Genome Biology 2004, 5:105

様々なタイプの機能性非コードRNAがある

機能性RNAもゲノム上にコードされている
miRNAの発見、RNAi（RNA干渉）の開発
ncRNAは、生体内のネットワークの重要な要素の一つ



遺伝子の構造（真核生物）



タンパク質コード遺伝子の発見法

- 既知転写産物との類似性
  - DB上の既知遺伝子（DNA）で相同性検索
  - DB上の既知タンパク質で相同性検索
  - cDNAをゲノムDNA配列に「貼り付ける」
- 比較ゲノム
  - 保存領域（synteny）は「重要」
- “ab initio” 遺伝子発見
  - 統計的な情報を用いた遺伝子の確率モデル

タンパク質コード遺伝子の発見法

- 既知転写産物との類似性
  - アラインメントによる配列の比較
    - 動的計画法(DP)
    - BLAST
- 比較ゲノム
- "ab initio" 遺伝子発見
  - 統計的な情報を用いた遺伝子の確率モデル
    - モデルによるゲノムDNAの「局所検索」

非コードRNA 遺伝子の発見法

- 既知転写産物との類似性
  - アラインメントによる配列の比較
    - 動的計画法(DP)
    - BLAST？？？？
- 比較ゲノム
  - きわめて重要(QRNA, RNAz)

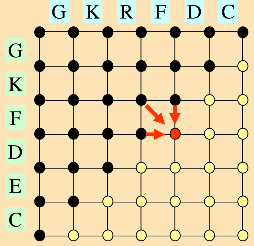
However, is it OK to define the "conserved region" by sequence similarities ?

- "ab initio" 遺伝子発見
  - 統計的な情報を用いた遺伝子の確率モデル
    - モデルによるゲノムDNAの「局所検索」

RNA配列情報解析の課題

- ゲノム配列からの既知ncRNA発見
  - tRNAscan-SE, Snoscan, Infernal
- cDNA配列からのncRNA発見
  - クラスタリング、二次構造予測(xxxfold)
- 比較ゲノム保存領域からのncRNA発見
  - 保存領域抽出、ncRNA判定(QRNA, RNAz)
- 特定モチーフを持つncRNA発見
  - 二次構造モチーフと特定配列。標的予測

配列の比較：2次元DP



$$F(i, j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_j) \\ F(i, j-1) - d \\ F(i-1, j) - d \end{cases}$$

メモリ　O(L^2)

計算時間　O(L^2)

通常のアライメント法

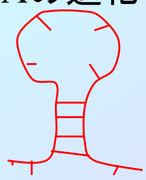
```
| | | | | | | | | | | | | | |
G T T A A C T G A G T A A C G
| x x | x | | | | | | x | | |
G C A A G C T G A G T T A C G
```

	A	C	G	T
A	1.1	-2.7	-1.2	-2.8
C	-2.0	1.4	-2.1	-1.1
G	-1.1	-2.1	1.4	-2.1
T	-2.7	-1.2	-2.6	1.1

- 塩基配列の各位置が、進化の過程で、それぞれ独立に他の塩基への置換を起こすというモデルに基づいている。
- ClustalWなど

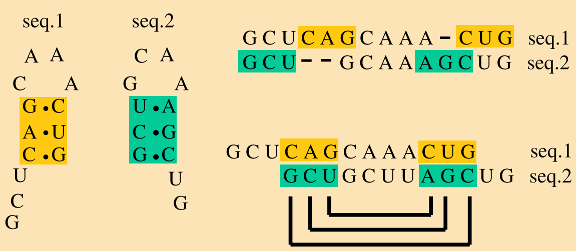
二次構造をもつRNAの進化

```
T T G T T C G A A A G A A C G
| | | x x | | | | | | x x | |
T T G A C C G A A A G G T C G
```

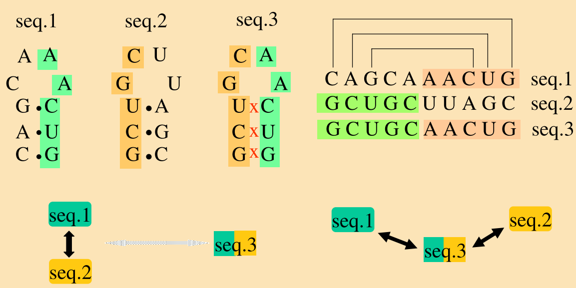


- 塩基対をつくる塩基ペアは、塩基対を保つような塩基置換の仕方(共置換)をする。
- →離れた2点での塩基の進化は独立でない。
- 配列の相同性が低くなってくると従来のアライメント法はうまくいかなくなってくる。

二次構造を考慮したアラインメント



二次構造を考慮したスコア



あらゆるRNA配列情報解析の基礎

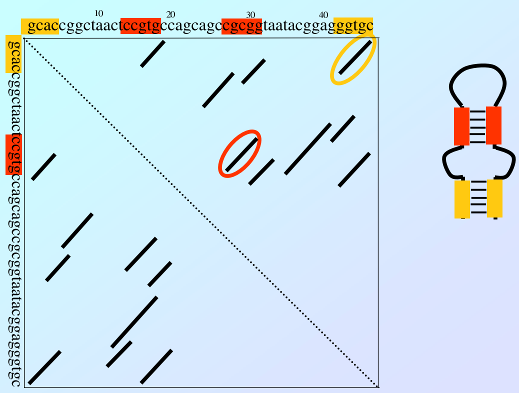
RNA配列の比較

- 配列の文字列としての類似性
  - ステム部分の塩基対類似性
  - 非塩基対部分の配列類似性
- RNAの二次構造の類似性
  - 二次構造が異なる　　→ どう比較するのか？
  - 二次構造が分からない→ どう比較するのか？

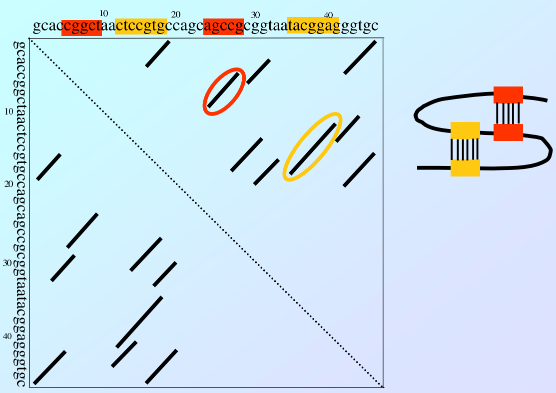
RNAの２次構造予測

- 自由エネルギー最小化（含：準最適構造）
  - Nussinov & Jacobson (“Nussinov Algorithm, 1980)
  - Mathews, Turner & Zuker (MFOLD, 2000)
- 配列比較解析
  - 相同RNA配列群が必要、結果が初期整列に依存
  - Chiu and Kolodziejczak (1991)
  - Eddy and Durbin (CM=プロファイルSCFG, 1994)

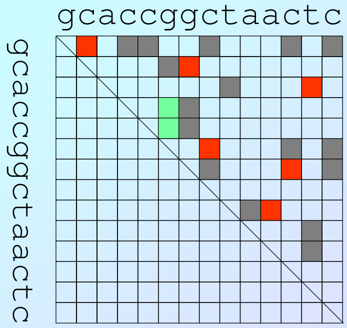
Secondary Structure Prediction of RNA



Secondary Structure Prediction of RNA

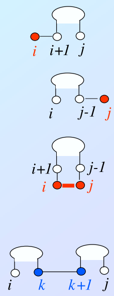


Secondary Structure Prediction of RNA



Nussinov アルゴリズム

- $M(i,j)$: maximum # of base pairs in $(x_i, \ldots x_j)$
  - $M(i,j) = M(i+1, j)$
  - $M(i,j) = M(i, j-1)$
  - $M(i,j) = M(i+1, j-1) + 1$
  - $M(i,j) = \max_k (M(i,k)+M(k+1,j))$



確率文脈自由文法（SCFG）による RNA二次構造予測

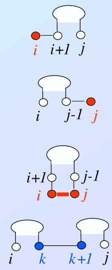
Σ = {a, c, g, u}, (x,y = a|c|g|u)

S = xS　$M(i,j) = M(i+1, j)$

S = Sx　$M(i,j) = M(i, j-1)$

S = xSy　$M(i,j) = M(i+1, j-1) + 1$

S = SS　$M(i,j) = \max_k (M(i,k)+M(k+1,j))$



SCFGのCYK アルゴリズム

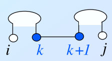
$\gamma(i,j)$: maxmum probability of (xi, … xj) is parsed by S

S = xS
S = Sx
S = xSy
S = SS

$$\gamma(i,j) = \max \begin{cases} \gamma(i+1,j) + \log p(x_i S); \\ \gamma(i,j-1) + \log p(S x_j); \\ \gamma(i+1,j-1) + \log p(x_i S x_j); \\ \max_{i<k<j}\{\gamma(i,k) + \gamma(k+1,j) + \log p(SS)\} \end{cases}$$

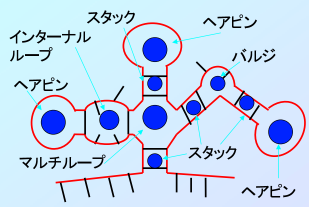
RNA二次構造予測の計算複雑度

- $M(i,j) = \max_k (M(i,k)+M(k+1,j))$



  - メモリ: $M(i,j)$ for all position i,j = L^2
  - 計算時間: Iterate for all i,j,k = L^3

Turnerのエネルギーパラメーター



RNA配列比較の課題

- 2本配列の比較
  - 共通二次構造の推定とアラインメント
  - 局所アラインメント(検索)
- 配列群のマルチプルアラインメント
- 配列群の共通二次構造予測
- 二次構造付配列への構造アラインメント
  - 局所アラインメント(検索)
- 二次構造への構造アラインメント
  - 局所アラインメント(二次構造モチーフ検索)
- 配列クラスタリング
- 特定RNAのモデル
  - 局所アラインメント(特定RNA発見)
- 比較ゲノム保存領域からの機能性RNA発見

SCARNA

Marlet

PHMMTS

PSTAG

共通2次構造予測

- Sankoff (1985)
  - 2(N)本のRNAの折畳み。$O(L^{3N})$ time, $O(L^{2N})$space
- Gorodkin, Heyer & Stormo (FOLDALIGN, 1997)
  - 2本局所整列、分岐構造なし、塩基対の最大化、$O(L^4)$ time
  - 多重整列:　　トーナメント法＋　Greedy アルゴリズム
- Mathews & Turner (Dynalign, 2002)
  - 自由エネルギー最小化＋配列比較解析
  - ステム間距離をM以下 $O(M^3 L^3)$ time
- Perriquet, Touzet & Dauchet (CARNAC, 2003)
  - 局所配列類似性　＋　エネルギー　＋　塩基対共変

2本のRNA配列からの共通二次構造の計算

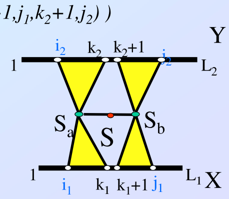
$\gamma(i_1,j_1,i_2,j_2)$: 4 次元のDP行列

(# of non-terminal is constant)

$$\gamma(i_1,j_1,i_2,j_2) = \max_{k_1,k_2} \left(\gamma(i_1,k_1,i_2,k_2) + \gamma(k_1+1,j_1,k_2+1,j_2) \right)$$

メモリ : $O(L^4)$

計算時間 : $O(L^6)$



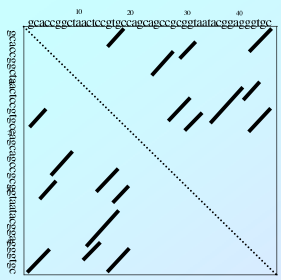
長さの6乗の計算時間とは？

- 配列長が2倍になると64倍の計算時間
- 配列長が10倍になると100万倍！！
- 1分の100万倍は約2年
- Sankoffアルゴリズムでは、150塩基の2配列の共通二次構造の計算に10分以上かかります、、、

SCARNA
Stem Candidate Aligner for RNA

- For each nucleotide sequences,
  - Extract stem candidates of a fixed length
  - Decompose each stem candidates into two parts:
    5' and 3' stem component (left SC and right SC)
  - Sort all SCs as a sequence by their positions
- Pairwise alignment DP of SCSs
- Remove inconsistent stem matches
- Construct backbone of common secondary structure
- Alignment of remaining nucleotide

Stem Candidates

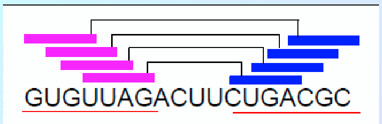


Base-pairing probabilities are calculated by McCaskill's algorithm.

Base pairs of low probabilities are not used

Stem fragments of a fixed length

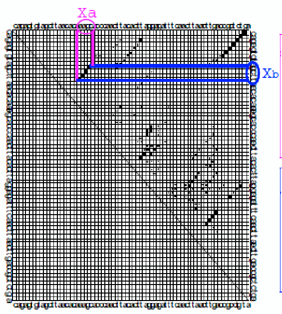
Stem candidates longer than the fixed length is treated as continuous stem fragments



SCARNA
Stem Candidate Aligner for RNA

- For each nucleotide sequences,
  - Extract stem candidates of a fixed length
  - Decompose each stem candidates into two parts:
    5' and 3' stem component (left SC and right SC)
  - Sort all SCs as a sequence by their positions
- Pairwise alignment DP of SCSs
- Remove inconsistent stem matches
- Construct backbone of common secondary structure
- Alignment of remaining nucleotide

Decomposition of stem fragments into stem components



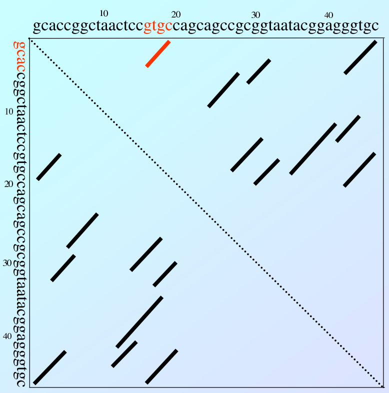
5'Stem Component (Xa)
(a) 20 position
(b) aagc sequence
(c) -6 distance
(d) gctt complementary
(e) 3.84 confidence score
(f) -0.9 stacking energy

3'Stem Component (Xb)
(a) 10
(b) gctt
(c) 6
(d) aagc
(e) 3.84
(f) -0.9

SCARNA
Stem Candidate Aligner for RNA

- For each nucleotide sequences,
  - Extract stem candidates of a fixed length
  - Decompose each stem candidates into two parts:
    5' and 3' stem component (left SC and right SC)
  - Sort all SCs as a sequence by their positions
- Pairwise alignment DP of SCSs
- Remove inconsistent stem matches
- Construct backbone of common secondary structure
- Alignment of remaining nucleotide

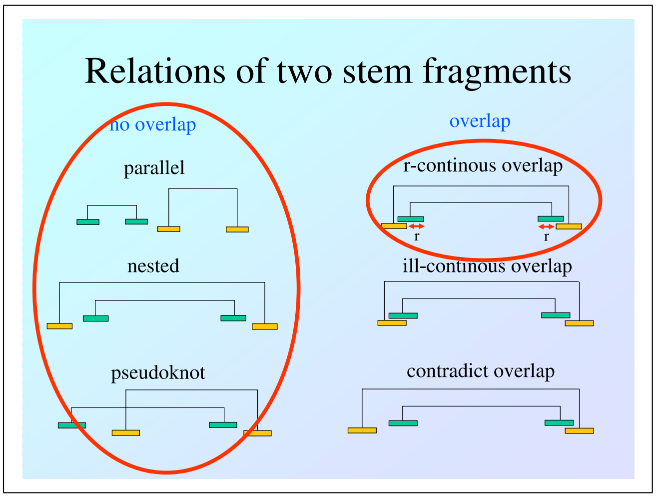
Stem Component Sequence (SCS)

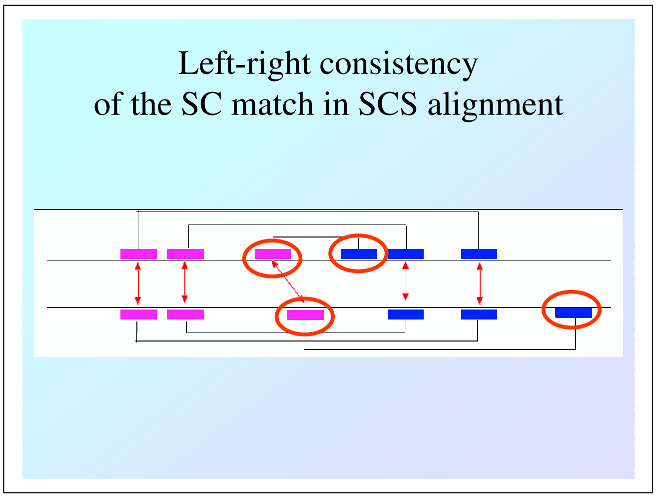


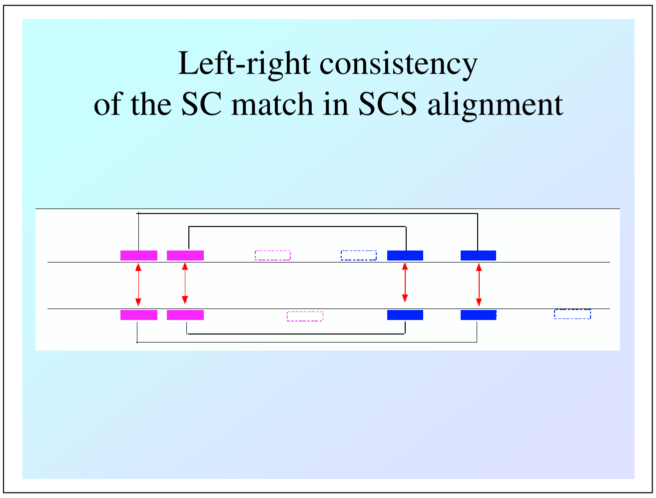
1 gcac	11	gtgc 16
1 gcac	38	gtgc 43
2 cacc	36	ggtg 42
3 accg	22	cggt 29
5 cggc	16	gccg 25
6 ggct	14	agcc 24
11 actc	26	gggt 41
12 ctcc	22	ggag 38
13 tccg	20	cgga 37
14 ccgt	10	gcgg 28
14 ccgt	18	acgg 36
15 cgtg	8	cgcg 27
15 cgtg	16	tacg 35
16 gtgc	−11	gcac 1
16 gtgc	23	gtgc 43
17 tgcc	9	ggta 30
17 tgcc	−21	ggtg 42
24 agcc	−14	ggct 6
25 gccg	−16	cggc 5
27 cgcg	−8	cgtg 15
28 gcgg	−10	ccgt 14
29 cggt	−22	accg 3
30 ggta	−9	tgcc 17
35 tacg	−16	cgtg 15
36 acgg	−18	ccgt 14
37 cgga	−20	tccg 13
38 ggag	−22	ctcc 12
41 gggt	−26	actc 11
42 ggtg	−36	cacc 2
42 ggtg	−21	tgcc 17
43 gtgc	−38	gcac 1
43 gtgc	−23	gtgc 16

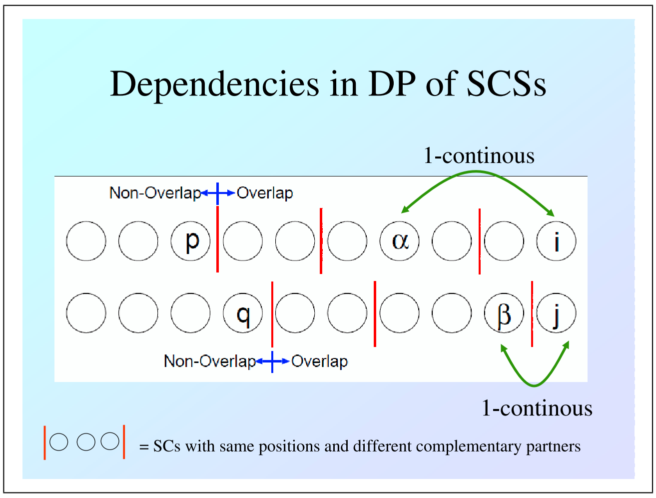
SCARNA
Stem Candidate Aligner for RNA

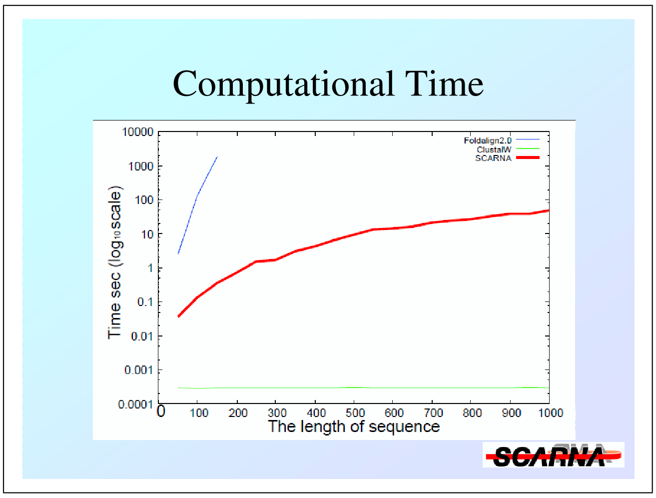
- For each nucleotide sequences,
  - Extract stem candidates of a fixed length
  - Decompose each stem candidates into two parts:
    5' and 3' stem component (left SC and right SC)
  - Sort all SCs as a sequence by their positions
- Pairwise alignment DP of SCSs
- Remove inconsistent stem matches
- Construct backbone of common secondary structure
- Alignment of remaining nucleotide

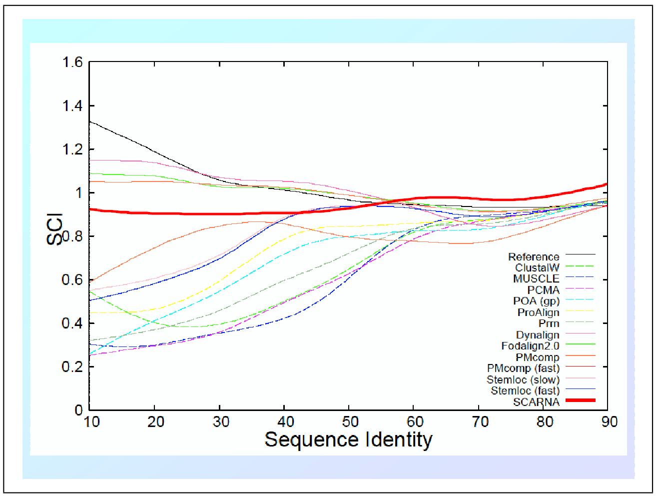

Relations of two stem fragments
no overlap
overlap
parallel
r-continous overlap
r
r
nested
ill-continous overlap
pseudoknot
contradict overlap


Left-right consistency
of the SC match in SCS alignment


Left-right consistency
of the SC match in SCS alignment


Dependencies in DP of SCSs
1-continous
Non-Overlap
Overlap
p
α
i
q
β
j
Non-Overlap
Overlap
1-continous
= SCs with same positions and different complementary partners


Computational Time
Foldalign2.0
ClustalW
SCARNA
Time sec (log₁₀scale)
10000
1000
100
10
1
0.1
0.01
0.001
0.0001
0
100
200
300
400
500
600
700
800
900
1000
The length of sequence
SCARNA

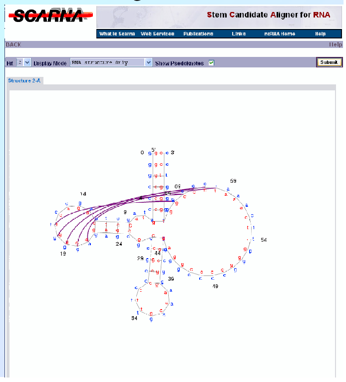

SCI
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
10
20
30
40
50
60
70
80
90
Sequence Identity
Reference
ClustalW
MUSCLE
PCMA
POA (gp)
ProAlign
Prrn
Dynalign
Fodalign2.0
PMcomp
PMcomp (fast)
Stemloc (slow)
Stemloc (fast)
SCARNA

Scarna web server

http://www.scarna.org

Scarna web server

http://www.scarna.org



Acknowledgments

- “Functional RNA Project” of METI
- Grant-in-Aid for Scientific Research on Priority Area “Comparative Genomics”
- AIST, CBRC, BIRC
- JBIC
- The University of Tokyo

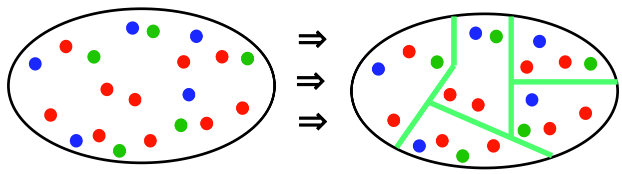
3つの資源節点集合を持つ4点連結グラフを均等分割する問題について

石井 利昌 (豊橋技術科学大学)
岩田 健吾 (マツダ)
永持 仁　(京都大学)

Background

Districting Problem
- political constituencies
- school board boundaries
- sales or delivery regions

Criteria
- equity
- contiguousness

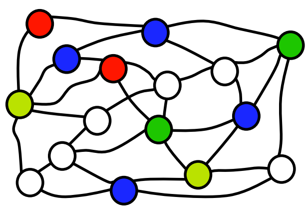


---

k-bipartition Problem

Input:

i) graph $G=(V, E)$.
ii) disjoint subsets $T_1, T_2, T_3, T_4, \ldots, T_k \subseteq V$
(Resource sets)
($|T_i|$: even)

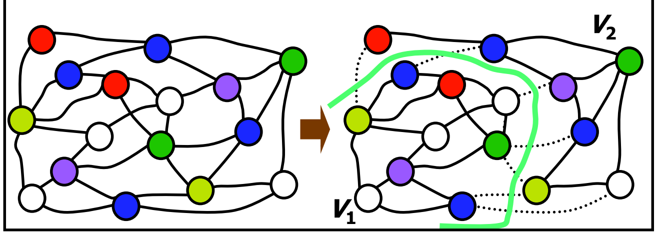


---

k-bipartition Problem

Output:

a partition $\{V_1, V_2\}$ of V → ***k*-bipartition**
s.t.
(1) $|T_i \cap V_1| = |T_i \cap V_2| = |T_i|/2$ for each i,
(2) Both of V_1 and V_2 induce connected graphs.



Related Results

k : # of resource sets, $n=|V|$, $m=|E|$

- Testing whether a k-bipartition exists or not is NP-hard even if k=1 [Dyer, Frieze 85][Chleikova 99]

Related Results

k : # of resource sets, $n=|V|$, $m=|E|$

- Testing whether a k-bipartition exists or not is NP-hard even if k=1 [Dyer, Frieze 85][Chleikova 99]

- Sufficient condition for which a k-bipartition exists:
1-bipartition ・・・ 2-connectivity suffices.
$O(m)$ time [Suzuki et al.90][Wada, Kawaguchi94]
2-bipartition ・・・ 3-connectivity suffices.
$O(n^2 \log n)$ time [Nagamochi et al. 02]

Conjecture
Every (k+1)-connected graph admits a k-bipartition.

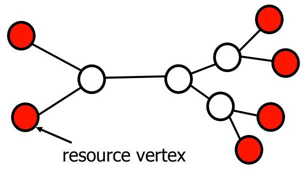
Our Recent Results

3-bipartition

(1) 5-vertex-connectivity does not suffice!
(2) 4-vertex-connectivity suffices if K4 is contained.
(3) For the edge version of *k*-bipartition (*k*=1,2,3), (*k*+1)-edge-connectivity suffices.

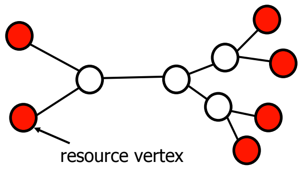
A 1-vertex-connected graph which has no 1-bipartition of *V*

k=1



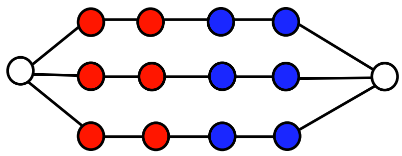
A 1-vertex-connected graph which has no 1-bipartition of *V*

k=1



A 2-vertex-connected graph which has no 2-bipartition of *V*

k=2



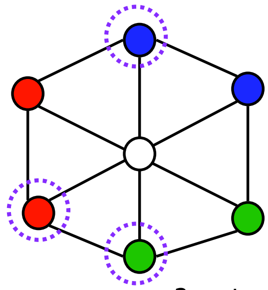
Our Results

3-bipartition

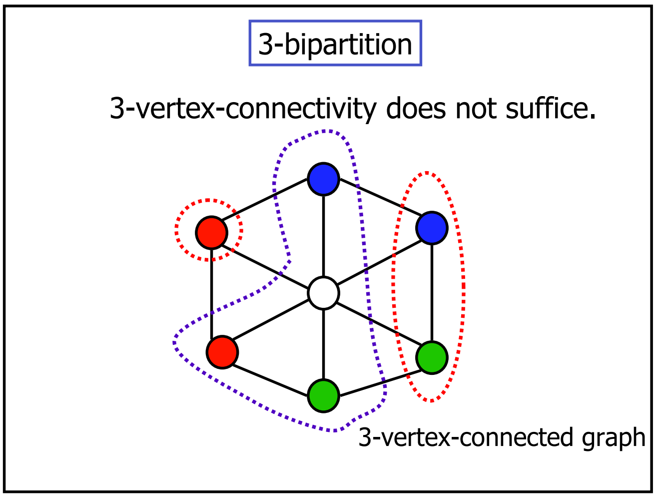
(1) 5-vertex-connectivity does not suffice!
(2) 4-vertex-connectivity suffices if K4 is contained.
(3) For the edge version of *k*-bipartition (*k*=1,2,3), (*k*+1)-edge-connectivity suffices.

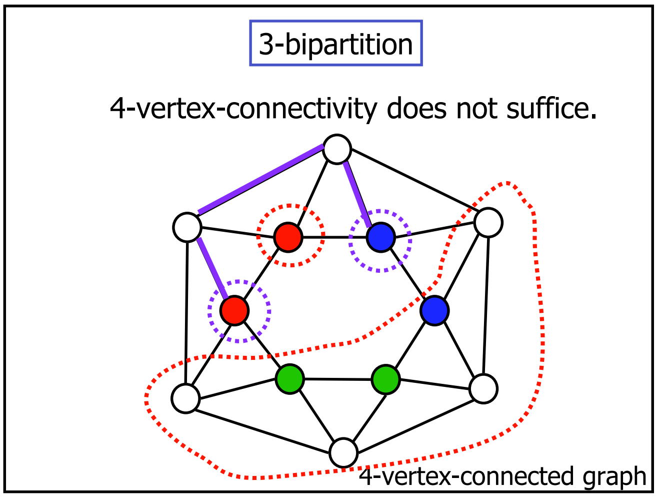
3-bipartition

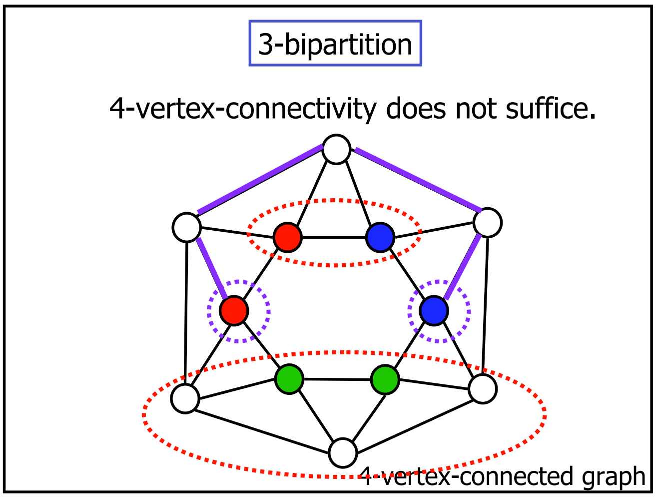
3-vertex-connectivity does not suffice.

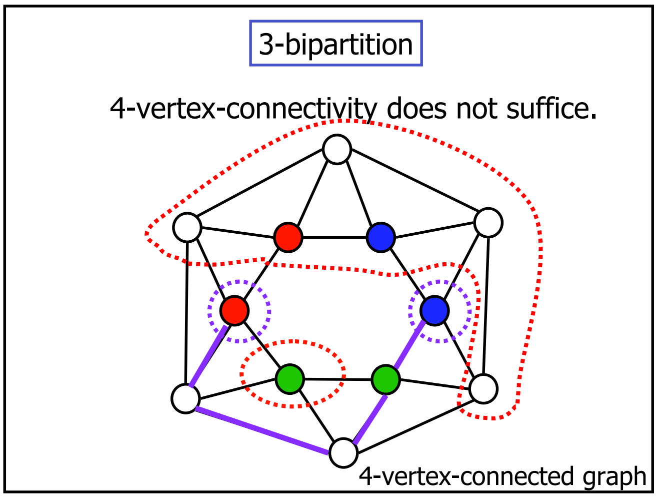


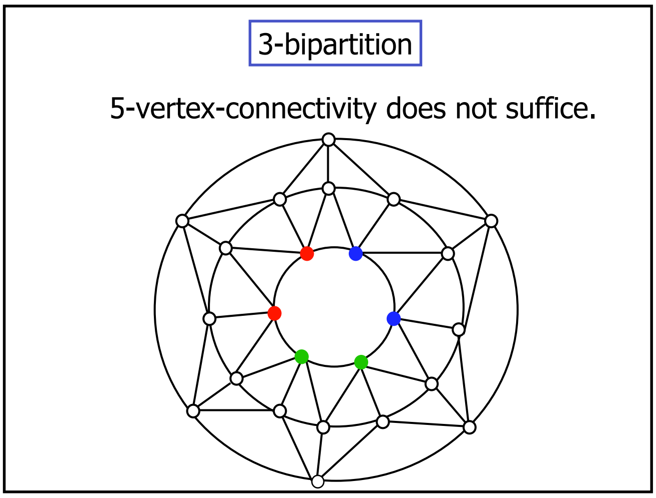
3-vertex-connected graph

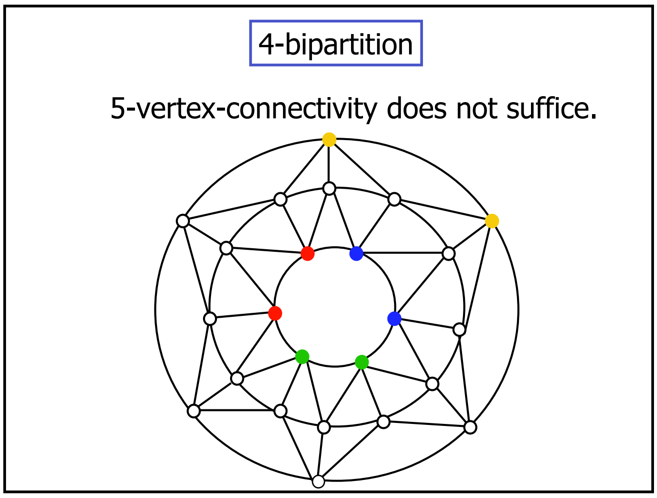

3-bipartition
3-vertex-connectivity does not suffice.
3-vertex-connected graph


3-bipartition
4-vertex-connectivity does not suffice.
4-vertex-connected graph


3-bipartition
4-vertex-connectivity does not suffice.
4-vertex-connected graph


3-bipartition
4-vertex-connectivity does not suffice.
4-vertex-connected graph


3-bipartition
5-vertex-connectivity does not suffice.


4-bipartition
5-vertex-connectivity does not suffice.

Our Results

3-bipartition

(1) 5-vertex-connectivity does not suffice!

(2) 4-vertex-connectivity suffices if K4 is contained.

(3) For the edge version of k-bipartition (k=1,2,3), (k+1)-edge-connectivity suffices.

Theorem

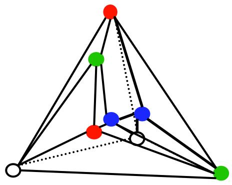
If G is a 4-vertex-connected graph and contains K4, then there exists a 3-bipartition, and moreover, a 3-bipartiton can be found in $O(n^3 \log n)$ time.

Algorithm for finding a 3-bipartition

Reduction to a geometrical problem [Nagamochi et al. 02]

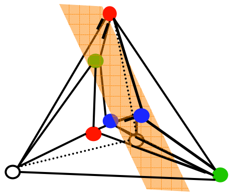
Phase 1

Find an embedding of G into the 3-dimensional space R^3 called **"convex-embedding"**.



Phase 2

Bisect V in R^3 into $\{V_1, V_2\}$ by a plane called **"ham-sandwich cut"**.



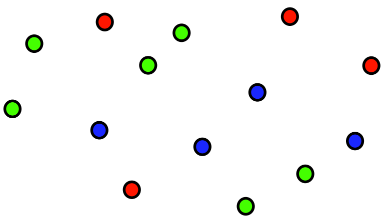
Ham-sandwich cut

$P_1, P_2, \ldots, P_k$: k subsets of points

Ham-sandwich cut with respect to $P_1, P_2, \ldots, P_k$

··· hyperplane bisecting each P_i

P_1, P_2, P_3



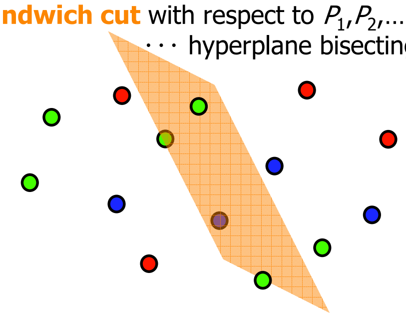
Ham-sandwich cut

$P_1, P_2, \ldots, P_k$: k subsets of points

Ham-sandwich cut with respect to $P_1, P_2, \ldots, P_k$

··· hyperplane bisecting each P_i

P_1, P_2, P_3



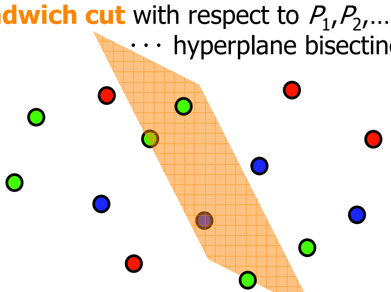
Ham-sandwich cut

$P_1, P_2, \ldots, P_k$: k subsets of points

Ham-sandwich cut with respect to $P_1, P_2, \ldots, P_k$

··· hyperplane bisecting each P_i

P_1, P_2, P_3



Theorem [Edelsbrunner87]

In R^k, a ham-sandwich cut w.r.t. $P_1, \ldots, P_k$ always exists.

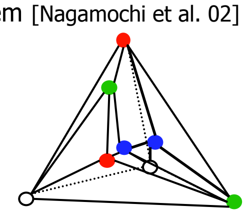
$O(n^{3/2})$ if k=3 (n : #points) [Chi-Yuan et al.94]

Algorithm for finding a 3-bipartition

Reduction to a geometrical problem [Nagamochi et al. 02]

Phase 1

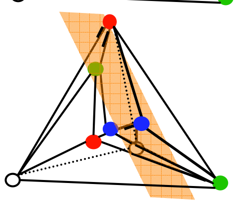
Find an embedding of G into the 3-dimensional space R^3 called **"convex-embedding"**.



Phase 2

Bisect V in R^3 into $\{V_1, V_2\}$ by a plane called **"ham-sandwich cut"**.

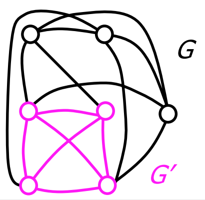
(1) $|T_i \cap V_1| = |T_i \cap V_2| = |T_i|/2$



Convex Embedding
[Nagamochi et al.02]

$N_G(v)$: the set of neighbors of v.

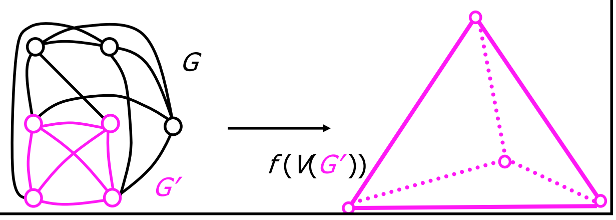
f: V R^k is a **convex embedding** of G with boundary G' into R^k,
if (i)
(ii)
(iii)



Convex Embedding
[Nagamochi et al.02]

$N_G(v)$: the set of neighbors of v.

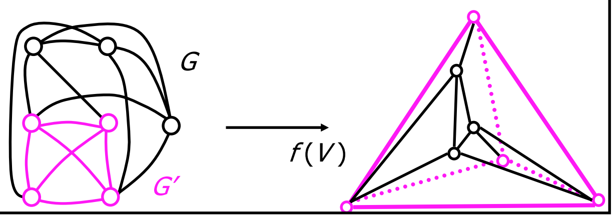
f: V R^k is a **convex embedding** of G with boundary G' into R^k,
if (i) the convex hull of $f(V(G'))$ is isomorphic to G',
(ii)
(iii)



Convex Embedding
[Nagamochi et al.02]

$N_G(v)$: the set of neighbors of v.

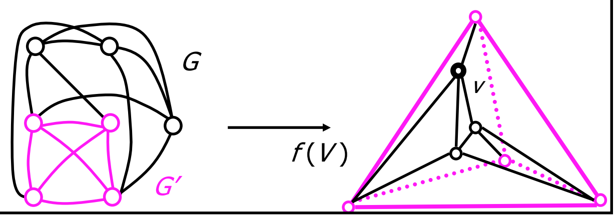
f: V R^k is a **convex embedding** of G with boundary G' into R^k,
if (i) the convex hull of $f(V(G'))$ is isomorphic to G',
(ii) For $\forall v \in V - V(G')$,
$f(v)$ is strictly included in the convex hull of $f(N_G(v))$.
(iii)



Convex Embedding
[Nagamochi et al.02]

$N_G(v)$: the set of neighbors of v.

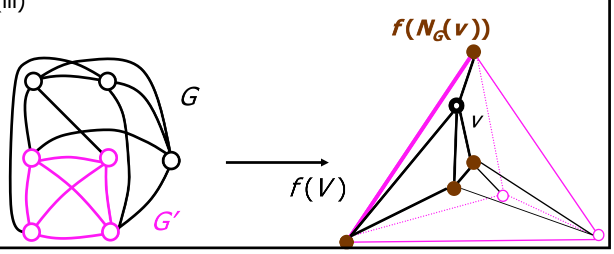
f: V R^k is a **convex embedding** of G with boundary G' into R^k,
if (i) the convex hull of $f(V(G'))$ is isomorphic to G',
(ii) For $\forall v \in V - V(G')$,
$f(v)$ is strictly included in the convex hull of $f(N_G(v))$.
(iii)



Convex Embedding
[Nagamochi et al.02]

$N_G(v)$: the set of neighbors of v.

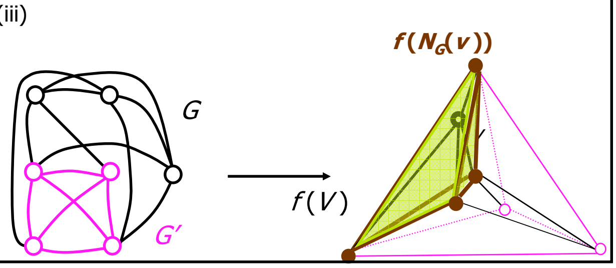
f: V R^k is a **convex embedding** of G with boundary G' into R^k,
if (i) the convex hull of $f(V(G'))$ is isomorphic to G',
(ii) For $\forall v \in V - V(G')$,
$f(v)$ is strictly included in the convex hull of $f(N_G(v))$.
(iii)



Convex Embedding
[Nagamochi et al.02]

$N_G(v)$: the set of neighbors of v.

f: V R^k is a **convex embedding** of G with boundary G' into R^k,
if (i) the convex hull of $f(V(G'))$ is isomorphic to G',
(ii) For $\forall v \in V - V(G')$,
$f(v)$ is strictly included in the convex hull of $f(N_G(v))$.
(iii)

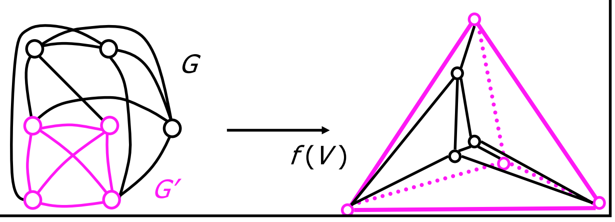


Convex Embedding

[Nagamochi et al.02]

$N_G(v)$: the set of neighbors of v.

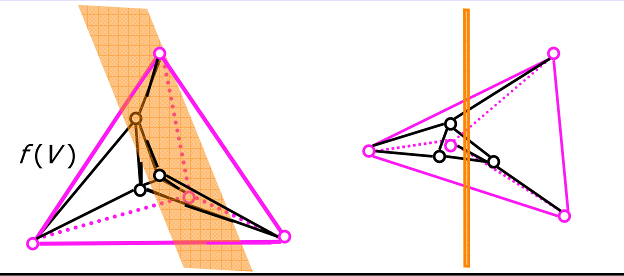
$f: V \quad R^k$ is a **convex embedding** of G with boundary G' into R^k,
if (i) the convex hull of $f(V(G'))$ is isomorphic to G'.
(ii) For $\forall v \in V - V(G')$,
$f(v)$ is strictly included in the convex hull of $f(N_G(v))$.
(iii) Points of $\{f(v) \mid v \in V\}$ are in general position.



Convex Embedding

Lemma [Nagamochi et al.02]

$f: V \quad R^k$: a convex embedding of G with boundary G' into R^k.
$\{V_1, V_2\}$: a parition of V obtained by separating $f(V)$ with **an arbitrary hyperplane**.
⇒ Both of V_1 and V_2 induce connected graphs.



Algorithm for finding a 3-bipartition

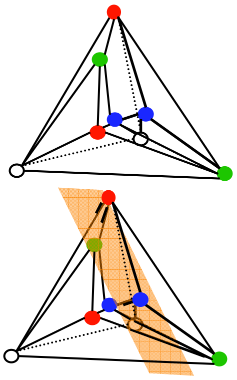
Reduction to a geometrical problem [Nagamochi et al. 02]

Phase 1
Find an embedding of G into the 3-dimensional space R^3 called **"convex-embedding"**.

Phase 2
Bisect V in R^3 into $\{V_1, V_2\}$ by a plane called **"ham-sandwich cut"**.

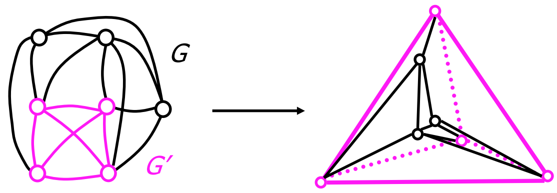
(1) $|T_i \cap V_1| = |T_i \cap V_2| = |T_i|/2$

(2) V_1 and V_2 induce connected graphs



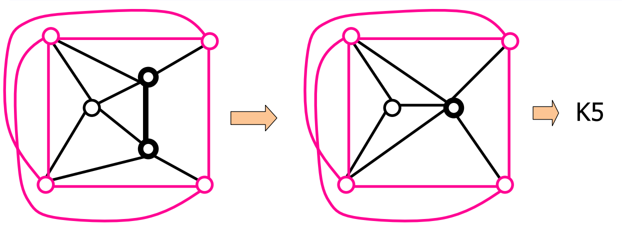
3-bipartition

Theorem
G : a 4-connected graph which includes K4 (denoted by G').
⇒ G has a convex embedding into R^3 with boundary G'



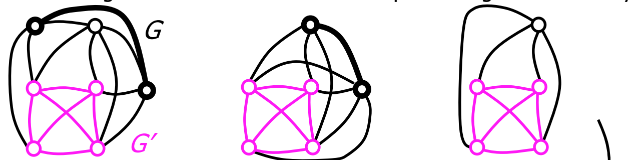
Key Lemma

Lemma Let G be a 4-vertex-connected graph $G(\neq K5)$, and H be a subgraph of G with H=K4.
Then G has a contractible edge in $E(G)$-$E(H)$ in such a sense that its contraction preserves 4-vertex-connectivity.

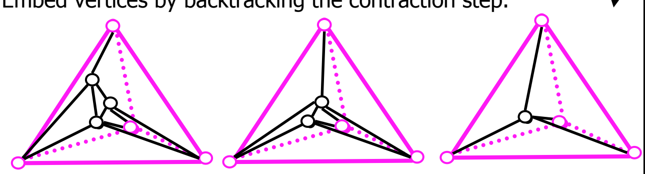


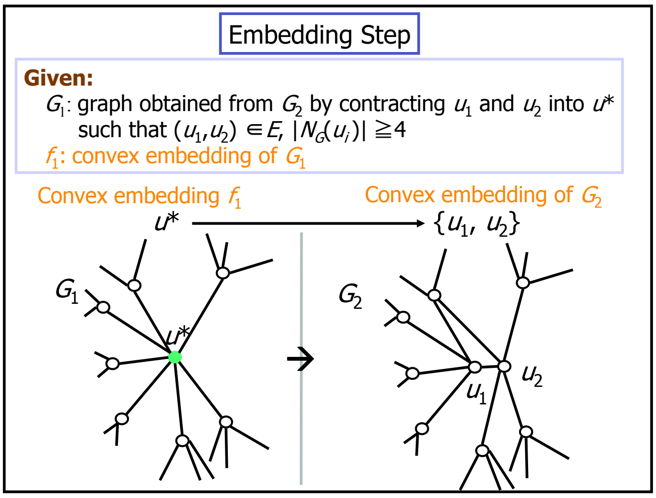
Algorithm for finding a convex embedding into R^3

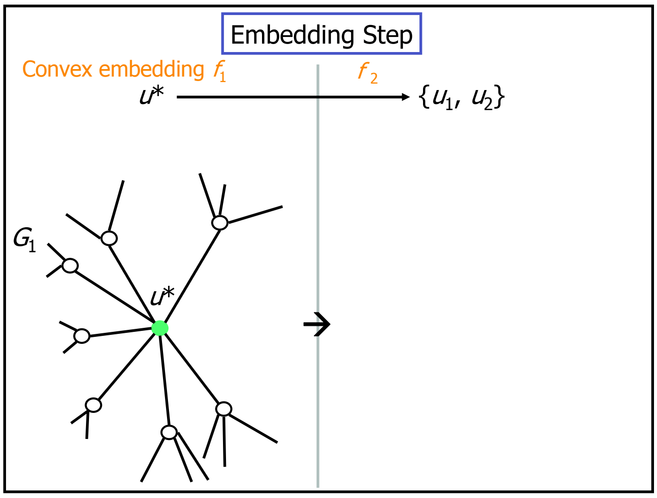
Step 1 (contraction step)
Contract edges not contained in G' while preserving 4-connectivity

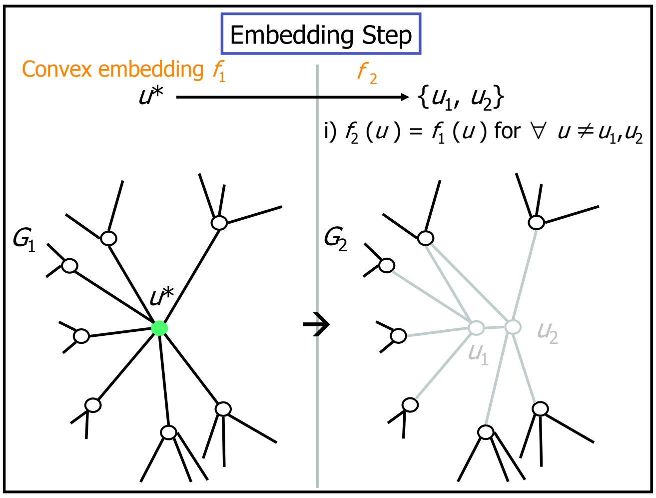


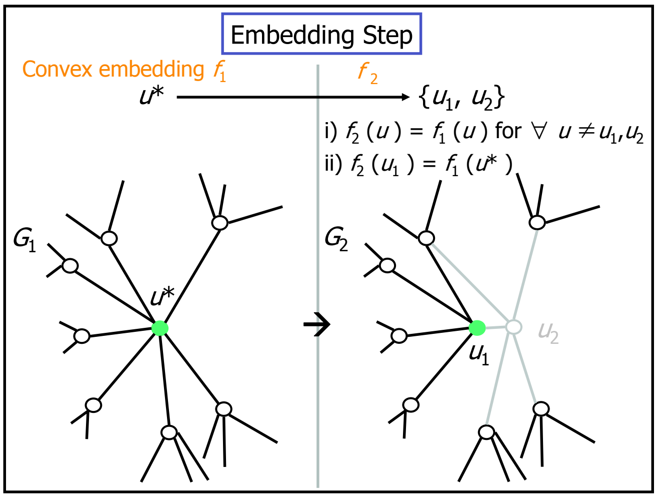
Step 2 (embedding step)
Embed vertices by backtracking the contraction step.

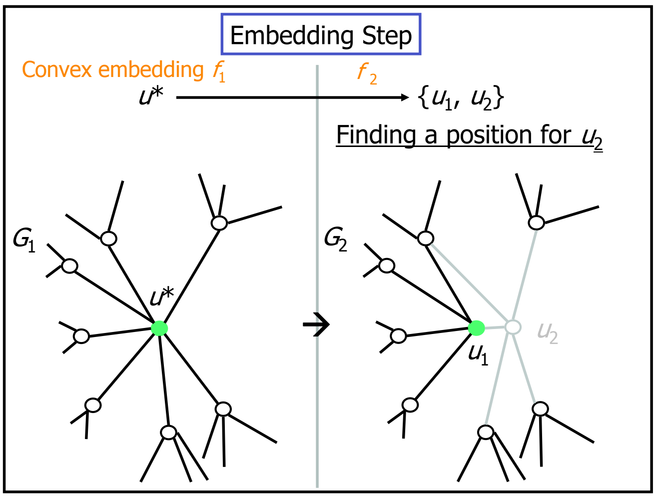


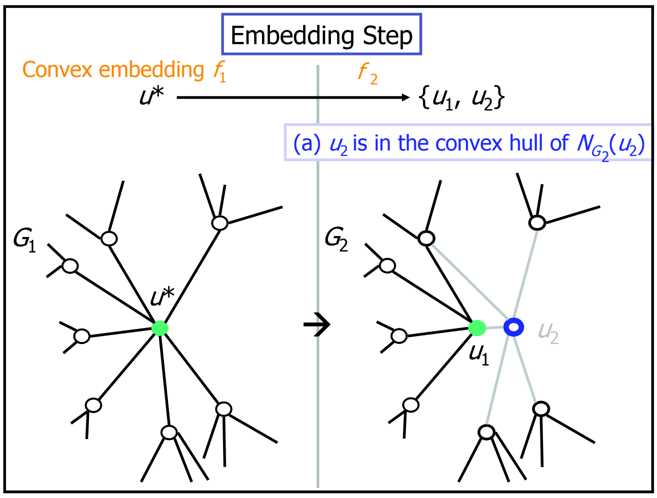

Embedding Step
Given:
G_1: graph obtained from G_2 by contracting u_1 and u_2 into u^* such that $(u_1,u_2) \in E$, $|N_G(u_i)| \geqq 4$
f_1: convex embedding of G_1
Convex embedding f_1
Convex embedding of G_2
u^*
$\{u_1, u_2\}$
G_1
u^*
G_2
u_1
u_2

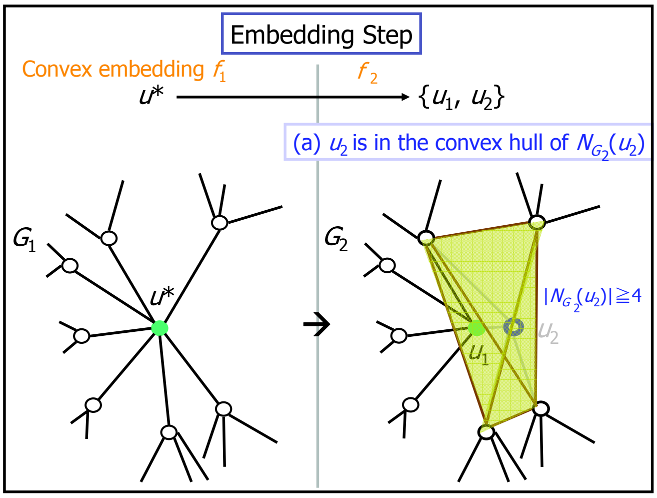

Embedding Step
Convex embedding f_1
f_2
u^*
$\{u_1, u_2\}$
G_1
u^*

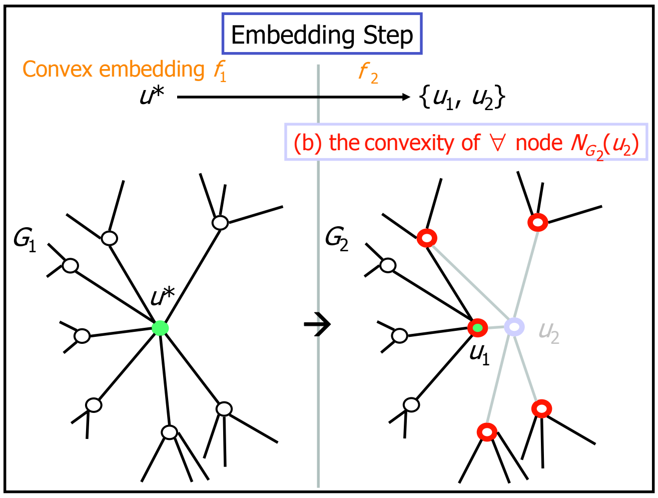

Embedding Step
Convex embedding f_1
f_2
u^*
$\{u_1, u_2\}$
i) $f_2(u) = f_1(u)$ for $\forall\ u \neq u_1, u_2$
G_1
u^*
G_2
u_1
u_2

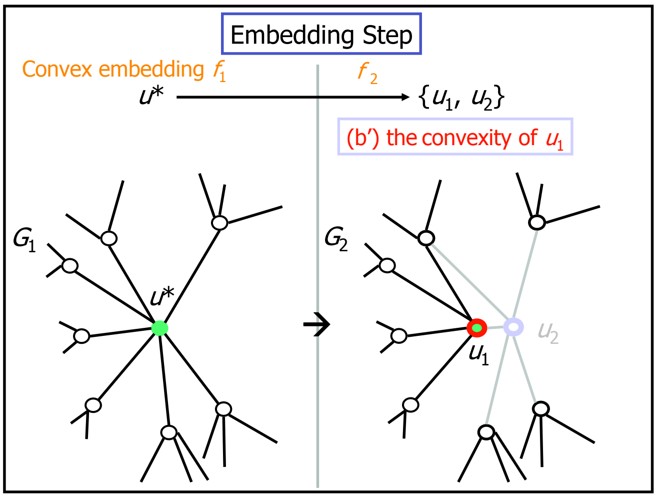

Embedding Step
Convex embedding f_1
f_2
u^*
$\{u_1, u_2\}$
i) $f_2(u) = f_1(u)$ for $\forall\ u \neq u_1, u_2$
ii) $f_2(u_1) = f_1(u^*)$
G_1
u^*
G_2
u_1
u_2

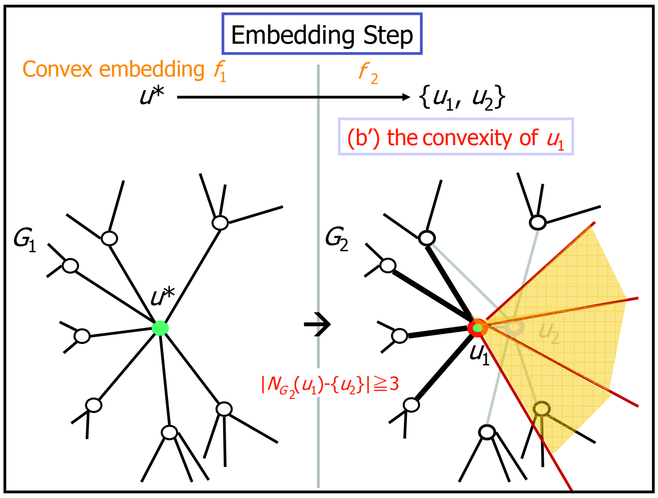

Embedding Step
Convex embedding f_1
f_2
u^*
$\{u_1, u_2\}$
Finding a position for u_2
G_1
u^*
G_2
u_1
u_2

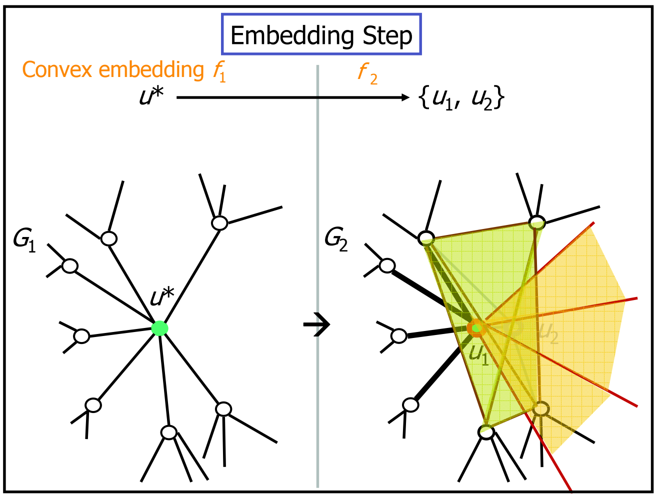

Embedding Step
Convex embedding f_1
f_2
u^*
$\{u_1, u_2\}$
(a) u_2 is in the convex hull of $N_{G_2}(u_2)$
G_1
u^*
G_2
u_1
u_2

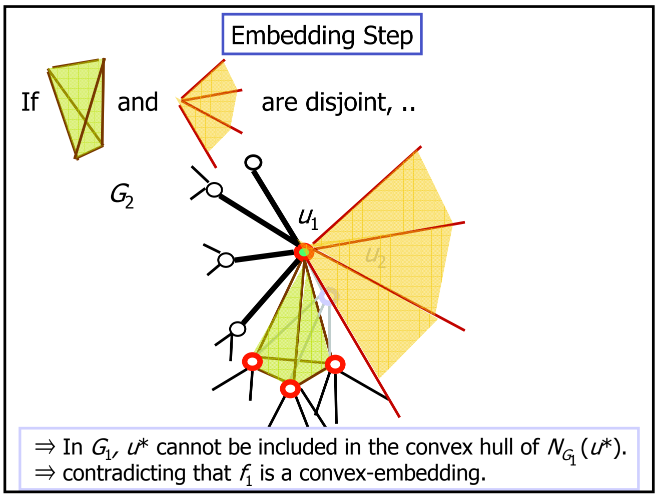

Embedding Step
Convex embedding f_1
u^*
f_2
$\{u_1, u_2\}$
(a) u_2 is in the convex hull of $N_{G_2}(u_2)$
G_1
u^*
G_2
$|N_{G_2}(u_2)| \geqq 4$
u_1
u_2


Embedding Step
Convex embedding f_1
u^*
f_2
$\{u_1, u_2\}$
(b) the convexity of $\forall$ node $N_{G_2}(u_2)$
G_1
u^*
G_2
u_1
u_2


Embedding Step
Convex embedding f_1
u^*
f_2
$\{u_1, u_2\}$
(b′) the convexity of u_1
G_1
u^*
G_2
u_1
u_2


Embedding Step
Convex embedding f_1
u^*
f_2
$\{u_1, u_2\}$
(b′) the convexity of u_1
G_1
u^*
G_2
u_1
u_2
$|N_{G_2}(u_1)\text{-}\{u_2\}| \geqq 3$


Embedding Step
Convex embedding f_1
u^*
f_2
$\{u_1, u_2\}$
G_1
u^*
G_2
u_1
u_2


Embedding Step
If
and
are disjoint, ..
G_2
u_1
u_2
⇒ In G_1, u^* cannot be included in the convex hull of $N_{G_1}(u^*)$.
⇒ contradicting that f_1 is a convex-embedding.

Our Results

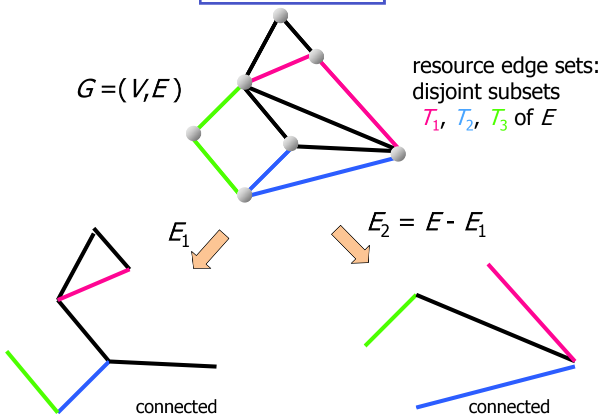
3-bipartition

(1) 5-vertex-connectivity does not suffice!

(2) 4-vertex-connectivity suffices if K4 is contained.

(3) For the edge version of k-bipartition (k=1,2,3), (k+1)-edge-connectivity suffices.

Edge-Version



Edge-Version

Input: a graph and subsets T_i of resource edge sets

Output: a bipartition $\{E_1, E_2\}$ of E

s.t. $|E_1 \cap T_i| = |E_2 \cap T_i|$

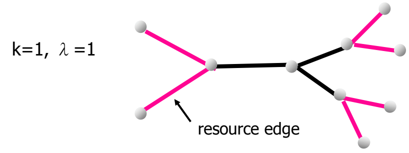
E_1 and E_2 induce connected graphs.

For the edge version of k-bipartition (k=1,2,3), (k+1)-edge-connectivity suffices.

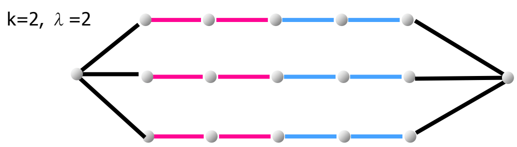
G --> Line graph $L(G)$

(k+1)-edge-connected --> (k+1)-vertex-connected & Kk+1

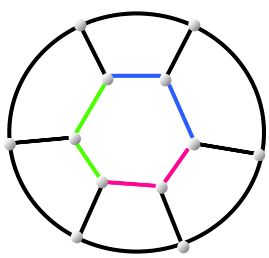
A 1-edge-connected graph which has no 1-bipartition of E



A 2-edge-connected graph which has no 2-bipartition of E



A 3-edge-connected graph which has no 3-bipartition of E



24 vertices
deg=5
κ=5

What we have done is ...

Every 4-vertex-connected graph G admits a 3-bipartition if G has a K_4

5-vertex-connecitivity does not suffice for 3-bipartition
5-vertex-connecitivity does not suffice for 4-bipartition
5-vertex-connecitivity does not suffice for 5-bipartition

The vertex version implies the edge version.

Every (k+1)-edge-connected graph G admits a k-bipartition of E (k=1,2,3).

Open Problems

·Sufficient condition for which a k-bipartition exists

Conjecture
Every (k+1)-vertex-connected graph with K_{k+1} admits a k-bipartition.

the edge version

Conjecture
Every (k+1)-edge-connected graph admits a k-bipartition.

Open Problem

Define $f(k)$ be the smallest p such that every p-vertex-connected graph admits a k-bipartition.
$f(1)=2$, $f(2)=3$

For k>5, prove $f(k) \geqq k+1$.

$f(3) \geqq 6$, $f(4) \geqq 6$, $f(5) \geqq 6$

For k>3, bound $f(k)$ from above by k+constant.

$f(k) = O(\Sigma |T_i|)$

The same questions for the edge version.

Approximating graph coloring of minor-closed graphs

Joint Work with Erik Demaine, Mohammad Hajiaghayi, Bojan Mohar, Robin Thomas
Partially joint Work with Neil Robertson and Paul Seymour

Ken-ichi Kawarabayashi
Tohoku University
E-mail: k_keniti@dais.is.tohoku.ac.jp
http://www.dais.is.tohoku.ac.jp/~k_keniti

Contents (Mostly, FOCS paper)

- Motivation (FOCS paper)
- 2-approx. of the chromatic number of minor-closed graphs (FOCS paper)
- Tree-width, Grid-minor, RS-structure.
- Overview of Algorithm (Robertson-Seymour)
- Approx. the list-chromatic number of minor-closed graphs.
- Toward Structural Theorem

Why is it accepted in FOCS?

- It is building on math deep theory. (although NOT AT ALL practical.)
- Minor-closed graphs are natural. (a generalization of planar graphs.)
- It tells how to use RS' main structural theorem.
- It is a bit easier to access (than RS' papers)
- Nice approx. for graph coloring of minor-closed graphs.
- Lucky.

Motivation

Mathematical Motivation

1. Hadwiger's Conjecture. (A far generalization of 4CT)
2. Graph Minor Theory (Robertson-Seymour)

Algorithmic Motivation

1. Chromatic number is hard to compute.
   NP-complete even for deciding 3-colorability of Planar graphs.
2. Even hard to approx.
   NP-hard to approx. within constant factor.
3. NP-complete to decide the chromatic number of minor closed graphs. (Even for planar graphs)
   Can you approx. ?

Algorithmic Results

- Theorem (Demaine, Hajiaghayi, KK, FOCS2005)
  There exists a 2-approx. algorithm for the chromatic number in minor-closed graphs. (graphs with no K_k-minor)
- The best known result was $O(k\sqrt{\log k})$ approx.
- Proof uses the whole graph minor papers….
- Robertson-Seymour theory consists of 23 papers. Most of them are published in JCTB.

Why is it 2-approx ?

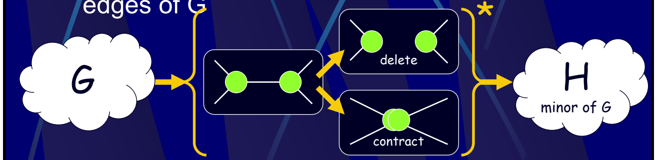
- The main theorem says that if G is K_k-minor-free graphs, then it can be decomposed into two graphs G_1,G_2 such that both G_1 and G_2 have tree-width at most f(k).
- If tree-width is bounded, one can compute the chromatic number in the linear time.
- It remains to give an algorithm for the main theorem…

Proof depends on

- Robertson-Seymour theorem.
- It gives a structural theorem for minor-closed graphs.
- Once we have this structure, the rest of proof is not so hard (but not trivial.)
- The main challenge is how to obtain RS-structure.
- It depends on the whole graph minor papers.

Minors

- A graph G ***has a minor H*** if H can be formed by removing and contracting edges of G



- Otherwise, G is ***H-minor-free***
- For example, planar graphs are both $K_{3,3}$-minor-free and K_5-minor-free

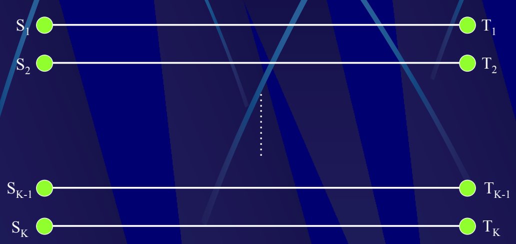
Graph Minor Theory [Robertson & Seymour 1984-2004]

- Seminal series of ≥ 20 papers
- Powerful results on excluded minors:
  - Every **minor-closed** graph property (preserved when taking minors) has a finite set of **excluded minors** [Wagner's Conjecture]
  - Every minor-closed graph property can be decided in polynomial time
  - For fixed graph H, graphs minor-excluding H have a special structure: drawings on bounded-genus surfaces + "extra features"

Highlights of Graph Minor Theory

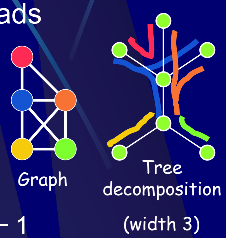
- Theorem(The disjoint paths problem) For fixed k, there is a polynomial time algorithm for deciding the disjoint paths problem.
- Minor testing can be done.
- Tree-width and grid-minors are discovered.
- Many mathematical and algorithmic applications.

The disjoint paths problems



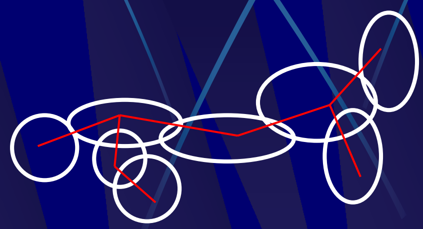
Treewidth [GM2—Robertson & Seymour 1986]

- ***Treewidth*** of a graph is the smallest possible width of a **tree decomposition**
- ***Tree decomposition*** spreads out each vertex as a connected subtree of a common tree, such that adjacent vertices have overlapping subtrees
  - ***Width*** = maximum overlap − 1



- Treewidth 1 ≡ tree; 2 ≡ series-parallel; ...

Tree-Decomposition



Tree-Decomposition of Graph

A tree-decomposition of a graph G is (T,W), where T is a tree and W=(W_t : t ⊂V(T)) satisfies

- $\bigcup_{t \subset V(T)} W_t = V(G)$
- If t' ⊂ T[t,t''], then $W_t \cap W_{t''} \subset W_{t'}$
- uv ⊂ E(G) for some t ⊂ V(t) s.t. u, v ⊂ W_t.

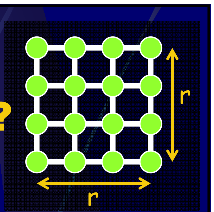
The width is max($|W_t|$-1 : t ⊂ V(T)).
The tree-width of G is a minimum width.

- Tree-width at most 1 < = > G is a forest.
- Tree-width at most 2 < = > G is series parallel.
- Tree-width at most 3 < = > G has no minor isomorphic to K_5, Octahedron, 5-prism, V_8.
- Tree-width of the complete graph of order n is n-1.
- Tree-width is minor-monotone.
- The (k × k)-grid minor has the tree-width k.

Tree-Width

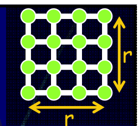
- Discovered by Robertson-Seymour.
- NP-hard to determine tree-width.
- A linear time to decide whether tree-width is k or not for fixed k.
- Many NP-hard problems can be solved in polynomial time if a given graph has small tree-width. (even linear)
- It is useful for structural results.
- It is a key for the proof of RS.
- It is closely related to grid.

Grid Minors
Why important ?



- **r × r grid**:
  - r^2 vertices, 2 r (r − 1) edges
  - Treewidth ~ r
- r × r grid is the **canonical** planar graph of treewidth Θ(r): every planar graph of treewidth w has an Ω(w) × Ω(w) grid minor [Robertson, Seymour, Thomas 1994]
  - So any planar graph of large treewidth has an r × r grid minor **certifying** large treewidth
  - What about nonplanar graphs?

Grid Minors
Why important ?



- For any fixed graph H, every H-minor-free graph of treewidth ≥ w(r) has an r × r grid minor [GM5—Robertson & Seymour 1986]
  - Re-proved & strengthened [Robertson, Seymour, Thomas 1994; Reed 1997; Diestel, Jensen, Gorbunov, Thomassen 1999]
  - Best bound of these: $w(r) = 20^{5|V(H)|^3 r}$ [Robertson, Seymour, Thomas 1994]
  - New optimal bound: w(r) = Θ(r) [Demaine,Hajiaghayi KK 2005]
    - Grids certify large treewidth in H-minor-free graph

Huge Grid is important

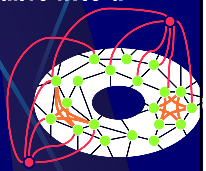
- Routing problem
- The disjoint paths problem and its generalization.
- Actually, Robertson-Seymour use this idea.

Structure of H-minor-free Graphs [GM16—Robertson & Seymour 2003] Main result of RS

- Every H-minor-free graph can be written as O(1)-**clique sums** of graphs
- Each summand is a graph that can be O(1)-**almost-embedded** into a bounded-genus surface
- O(1) constants depend only on |V(H)|

Almost-Embeddable Graphs

- A graph is ***O(1)-almost-embeddable into a bounded-genus surface*** if it is
  - A bounded-genus graph
  - + a bounded number of **vortices**:
    - ***Vortex*** = Replace a face in the bounded-genus graph by a graph of bounded **pathwidth**
    - The interiors of the replaced faces are disjoint
  - + a bounded number of **apices**:
    - ***Apex*** = extra vertex with any incident edges



What do we need ?

- Crosscaps
- Handles
- Genus
- Vortex
- Apex (easy)

But

- There cannot be so many crossings that are far apart.
- The genus addition process stops quite soon.

Otherwise, we would get a desired minor, a contradiction.

We know that

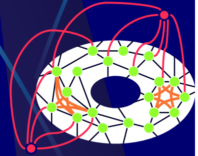
- Any long jump must be contained in the handle. This tells how to detect a handle.
- Any crossings and crosscaps are contained in small area. This tells how we can find a crosscap and a vortex.
- If there is no crosscap in the small area, then it is either vortex or planar graph.
- There cannot be many non-planar small areas that are far apart. This tells us that there are bounded number of vortices.

In summary

1. Stating with huge grid H.
2. As long as there is a long jump, we shall detect handles.
3. Otherwise the graph is embedded into a surface such that all the non-planar graphs are in small areas.
4. We shall look at each small area, and detect either vortex or crosscap.
5. There are only finitely many vortices and crosscaps. So the process stops.

Almost-Embeddable Graphs

- A graph is ***O(1)-almost-embeddable into a bounded-genus surface*** if it is
  - A bounded-genus graph
  - + a bounded number of **vortices**:
    - ***Vortex*** = Replace a face in the bounded-genus graph by a graph of bounded **pathwidth**
    - The interiors of the replaced faces are disjoint
  - + a bounded number of **apices**:
    - ***Apex*** = extra vertex with any incident edges



Approx. list coloring

- Theorem[Mohar and KK]

There is an O(k)-approx. for graphs without Kk-minor, I.e., minor-closed graphs.

Actually, it is "almost" O(√logk)-approx.

It is approximating within O(√logk)c + O(k), where c is optimal.

The best know appox. was O(k √ logk) approx.

Open: O(1) ? (Maybe NP-hard.)

Algorithm for List-coloring

- Theorem[KK & BM]

There is an O(n^3) algorithm for the following:

Input : A graph G, vertex set Z with |Z| <= 4k, precoloring of Z and each vertex in G has 16k-colors available in each list.

Output : Determine either

- G has a K_k-minor, or
- Precoloring of Z can be extended to the whole graph G, or
- G has a subgraph H such that H has no K_k-minor and has a vertex set Z' with |Z'| <=4k such that some precoloring of Z' cannot be extended to H.

Algorithm for List Coloring

- Corollary:

There is an O(n^3) algorithm for deciding the following:

(1) G has a K_k-minor

(2) G has a 16k-list-coloring

(3) G has a subgraph H such that H has no K_k-minor and no 12k-list-coloring.

It is easy to list-color by O(k √ logk) colors

NHC Spring School and Workshop on Discrete Algorithms

Feb. 27th – Mar. 3rd, 2006

University of Electro-Communications / Chofu Creston Hotel

<u>Spring School</u>

<u>Workshop</u>

Mar. 3rd (Fri.)

Could Your iPod Be Holding the Greatest Mystery in Modern Science?

by Bernard Chazelle

Tuturologists are an amiable bunch, so it is a puzzle why the future has been so cruel to them. From flying cars and self-cleaning houses to that bugaboo of workaholics—the leisure society—the soothsayer's trail is littered with the carcasses of pet predictions turned roadkill.

Gordon Moore need not worry. The co-founder of Intel tried his hand at crystal gazing once—and struck gold. His celebrated "law" makes the outlandish prediction that every 18 months, like clockwork, white-clad technicians will huddle in a silicon wafer clean room and cram twice as many transistors onto a microchip.

Moore's Law has ruled the roost for the last 40 years. All the oohs and aahs you hear about the digital revolution are nothing but the squeals humans emit when tickled pink by Moore's Law. From the nice (medical imaging, e-commerce, whole-genome sequencing) to the vital (Xbox, IM, iPod), its rule has been a veritable ticklefest. Moore's Law has been the sizzling cauldron in which savvy cooks have whipped up a dazzling variety of tasty dishes. Without it, the Information Superhighway would be a back alley to Snoozeville; the coolest thing about a computer would still be the blinking lights.

Moore's law has had a good run but, alas, its days are numbered. By mid-century, a repeal is all but certain. With the heady days of the Incredible Shrinking Chip receding in the past, expect the revolution to grind to a halt; expect pioneers to give way to tinkerers. Bye-bye ticklefest, hello slumber party.

No tears please. Perched atop their towering achievements, computer scientists (the cooks, remember?) will bask in the soothing certainty that their glorious science died at its peak. With a tinge of sadness but not a little pride, they'll chime in unison "There is nothing new to be discovered in computer science now."

If you think you've seen this movie before, you have. A few short years before Einstein turned our world upside down, the great Lord Kelvin bloviated this gem for the ages: "There is nothing new to be discovered in physics now." Not his lordship's finest hour.

Moore's Law has fueled computer science's sizzle and sparkle, but it may have obscured its uncanny resemblance to pre-Einstein physics: healthy and plump—and ripe for a revolution. Computing promises to be the most disruptive scientific paradigm since quantum mechanics. Unfortunately, it is the proverbial riddle wrapped in a mystery inside an enigma. The stakes are high, for our inability to "get" what computing is all about may well play iceberg to the Titanic of modern science.

Brilliant foresight or latest tripe from the Kelvin school of prophecy?

Computing is the meeting point of three Big Ideas: universality; duality; self-reference. To this triad, the modern view adds the concept of tractability and the revolutionary algorithmic paradigm. Here's how it works:

Universality Few would mistake your iPod for an IBM Blue Gene/L—the world's fastest computer. Yet, fundamentally, the two are the same. Why is that? At the heart of your iPod is a written document made of two parts: "*program, data*". The data section stores the songs as long sequences of 0s and 1s. The program section explains in words (again, 0s and 1s) how to read the data and turn it into sound. Add to this mix a smattering of hardware, the *control,* to read the program and follow its instructions, and voilà: you've got yourself an iPod. The beauty of the scheme is that the control need not know a thing about music. In fact, simply by downloading the appropriate program/data document, you can turn your iPod into an earthquake simulator, a word processor, a web browser, or a paperweight. Your dainty little MP3 player is a *universal* computer.

Separating *control* (the hardware) from *program* (the software) was the major insight of Alan Turing—well, besides this little codebreaking thing he did in Bletchley Park that helped win World War II. The separation was the key to universality. No one had seen anything quite like it before. At least not since the Chinese philosopher opined: "Give a man a fish and you feed him for a day. Teach a man to fish and you feed him for a lifetime." In Confucius's hands, the specialized view of fishing = river + fisherman finds itself replaced by a universal one: fishing = river + fishing manual + you. There you have it, computing = data + program + control. The control part of your iPod is a marvel of electronics, but the shocker is that it need not be so: universal computers can be built with control boxes vastly simpler than a cuckoo clock. For all purposes, computing = data + program.

Duality Consider the iPod document "*Print this, Let 'em eat cake*". Push the start button and watch the words "*Let 'em eat cake*" flash across the screen. Note how the program part of the document, *Print this*, is interpreted as a command—printing is what it wants and printing is what it gets. Contrast this with the data part, *Let 'em eat cake*, which is treated as plain text: no one's eating anything (to Marie-Antoinette's later chagrin). Strings of 0s and 1s are interpreted in one of two ways: as form (data) or as content (program). Tapping into the comic, artistic, and academic potential of this duality, great minds went to work: Abbott and Costello ("Who's on First?"), Magritte ("Ceci n'est pas une pipe"), and Saussure ("signified vs. signifier"). Staring at the sublime will, of course, send the deeper thinkers among us rushing for the classics—such as Homer Simpson's immortal quip: "Oh Marge, cartoons don't have any deep meaning; they're just stupid drawings that give you a cheap laugh."

Self-Reference Write the iPod document "*Print this twice, Print this twice*" and press the start button. The screen lights up with the words: "*Print this twice, Print this twice*". Lo and behold, the thing prints itself! Just like a computer virus (remember, I did *not* teach you this). The magic word is "twice." For example, the iPod document "*Print this, Print this*" prints this: "*Print this*"—more Dr. Seuss than self-replication.

The "Big Ideas" were the air that the Gang of Four, Princeton branch, breathed all day—that would be Alonzo Church, Alan Turing, Kurt Gödel, and John von Neumann. Mother Nature, of course, figured it all out a few billion years earlier. Reformat your genome by lining up the two strands of DNA one after the other, so it looks like a regular program-data iPod document (billions of letters long though):

"*ACAAGAT...GCCATTG,   TGTTCTA...CGGTAAC*".

The base pairings (A,T) and (C,G) ensure that the two strands spell the same word with different letters. So, we lose no genomic information if we translate the "data" part and rewrite the whole document as the duplicated text

"*ACAAGAT...GCCATTG,   ACAAGAT...GCCATTG*".

This is the biological analog of "*Print this twice, Print this twice*". Life's but a walking shadow, Macbeth warned us. Not quite. Life's but a self-printing iPod! Offended souls will bang on preachily about there being more to human life than the blind pursuit of self-replication—though Hollywood's typical fare would seem to refute that. Existential angst aside, *duality* is the option we have to interpret the word *ACAAGAT...GCCATTG* either as genes (the "form" encoding our genome) or as proteins (the "content" mediating the DNA replication). *Self-reference* is the duplication embodied in the base pairings. Viewed through the computing lens, life = duality + self-reference.

In the 1953 *Nature* article that unveiled to the world the structure of DNA, Watson and Crick signed off with this lovely understatement: *"It has not escaped our notice that the specific pairing we have postulated immediately suggests a possible copying mechanism for the genetic material."* Duality and self-reference embedded in molecules: what sweet music to Turing's ears this must have been! Alas, our war hero was a little distracted at the time, busy as he was enjoying the rewards that the British authorities had lavished upon him for saving millions of lives during World War II—generous rewards like a court conviction for homosexuality with a sentence of forced estrogen injections. Almost one year to the day of Watson and Crick's triumph, Alan Turing went home, injected cyanide into an apple, ate it, and died. His mother preferred to believe it was an accident.

Tractability The genesis of this fourth Big Idea was the ho-hum observation that checking the validity of a math proof tends to be much easier than finding the proof in the first place. But is it really? Amazingly, no one knows.

Welcome to the most important open question in all of computer science!

Ever wondered if your iPod's 5000-tune library is rich enough to let you compile a playlist of a thousand songs, no two which have ever been played back-to-back on MTV? Let's hope not, because not even an IBM Blue Gene/L could do the job in less time than has passed since dinosaurs were last seen roaming the earth. To find such a playlist (proof-finding) seems hopelessly hard, even on a computer, but to test whether a tentative playlist fits the bill (proof-checking) is a cinch: simply match all possible pairs against MTV's complete playlist, which is readily accessible on the web.

The twin reality of hard proof-finding and easy proof-checking is hardly an MTV-induced aberration. Computer scientists have catalogued over 1000 problems just like it. Of course, courtesy of Murphy's Law, these "Jurassic-1K" include all of the questions humanity is desperate to answer—in artificial intelligence, computational biology, resource allocation, rational drug design, etc.

OK, so life is tough. But since when has that observation qualified as a Big Idea?

Since 1970, roughly. Just as Einstein rebuilt Newtonian mechanics around the constancy of the speed of light, Cook, Edmonds, Karp, and Levin set out to rebuild computing around the notion of *tractability*. A problem is tractable if it can be solved in time growing polynomially in the input size, which is a fancy way of saying 'reasonably fast.' None of the Jurassic-1K appear to be tractable. At least those in the know believe they are not—of course, not so long ago, those in the know believed the earth was flat. Sadly, the great promise of computing seems to lie with problems afflicted with exponentialitis: the dreaded ailment that places even small-size problems beyond the reach of any computer.

This much we know: it's genetic. If a single one of the Jurassic-1K is tractable then, wonder of wonders, all of them are. These tough puzzles are nothing but different translations of the same Shakespeare play. Heady stuff! The day your playlist question can be answered in a few hours will be the day public-key cryptography dies, bringing down with it all of e-commerce. That day will see biology conquer its highest peak, protein folding, and mathematicians contemplate early retirement. Indeed, the day the Jurassic-1K are shown to be tractable (P=NP in computer parlance), proof-finding will be revealed to be no more difficult than proof-checking. Andrew Wiles, the conqueror of Fermat's Last Theorem, will be found to deserve no more credit than his referees. (Note that this says nothing about *understanding* the proof.) To be P or not to be P, that is NP's question. It is likely that P=NP would do for science what the discovery of the wheel did for land transportation. Little wonder no one believes it.

To discover the wheel is always nice, but to roll logs in the mud has its charms, too. Likewise, the intractability of proof-finding would have its benefits. When you purchase a book from Amazon, the assurance that your transaction is secure is predicated on more than your endearing naiveté. For one thing, it relies critically on the intractability of factoring a number into primes.

Just as modern physics shattered the platonic view of a reality amenable to noninvasive observation, tractability clobbers classical notions of knowledge, trust, persuasion, and belief. No less. For a taste of it, consider the great *zero-knowledge* (ZK) paradox: two mutually distrusting parties can convince each other that each one holds a particular piece of information without revealing a thing about it. Picture two filthy-rich businessmen stuck in an elevator. Their immediate goal is (what else?) finding out who's the wealthier. ZK dialogues provide them with the means to do so while revealing zero information about their own worth (material worth, that is—the other kind is already in full view).

Here is a ZK question for the State Department: can a signatory to the Nuclear Non-Proliferation Treaty demonstrate compliance without revealing any information whatsoever about its nuclear facilities? Just as game theory influenced the thinking of cold war strategists, don't be surprised to see ZK theory become the rage in international relations circles.

Tractability reaches far beyond the racetrack where computing competes for speed. It literally forces us to think differently. The agent of change is the ubiquitous *algorithm*.

The Algorithmic Revolution An algorithm is an iPod program with a human face. If a computer could wash your hair, its program would look like "0110001100100110..." but the algorithm behind it might read: "*Rinse, lather, repeat.*" (Don't try this at home if you're a computer scientist.) An algorithm is a list of instructions that tells the computer what to do. It may loop around and entertain alternatives, as in "*Rinse, lather, repeat if unhappy, dry, go to office, answer question: why didn't you rinse the shampoo off your hair?*" An algorithm is, in essence, a work of literature.

The library's bottom shelves might stack the one-line zingers—algorithmic miniatures that loop through a trivial algebraic calculation to produce *fractals* (pictures of dazzling beauty and infinite intricacy) or print the transcendentally mysterious digits of π. Algorithmic zingers can do everything. For the rest, we have the sonnets on the middle shelves. With names like FFT, RSA, LLL, AKS, they are short and crisp, and tend to pack more ingenuity per square inch than anything else in the computing world. The top shelves hold the lush, richly textured, multilayered novels.

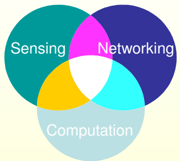
Give it to them, algorithmic zingers know how to make a scientist swoon. No one who's ever tried to compute the digits of π by hand can remain unmoved by the sight of its decimal expansion flooding a computer screen like lava flowing down a volcano. And that's not even the awesome part. For that, one must turn to the infamous Brazilian butterfly whose evil wing flaps cause typhoons in China. Zingers embody the potential of a local action to unleash colossal forces on a global scale: complexity emerging out of triviality. Cellular automata, chaos theory, dynamical systems, and all that.

For a glimpse of the fiction genre on the top shelves, check out *PCP*. Suppose that, after popping the genius pill, you wake up one bright morning with a complete proof of the Riemann hypothesis in your head (that's the "Notorious B.I.G." of math rap: the biggest open problem in the field). Few number theorists are likely to listen to your story. That is, until you offer them the *PCP deal*. You'll write down your proof in an agreed-upon format, and then let a "verifier" pick 10 lines at random. On the basis of these 10 lines alone, the verifier will decide whether your proof is correct. The shocker: beyond any reasonable doubt, she will be right! (Randomness plays a key role, but the chance of erring is less than that of the proverbial monkey typing all of Hamlet flawlessly.)

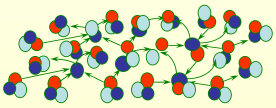
The mind reels. If your proof is fine, then it will pass any test the verifier can throw at it. But, based on only 10 lines, how can she know that you've proven the Riemann hypothesis and not a baby cousin like 2+2=4? If your proof is bogus, the intuition does not help much either. Presumably, the agreed-upon format is designed to smear any bug all across the proof. But how will the verifier be sure that you didn't play fast and loose with the formatting rules? So many ways to cheat; so little evidence to check. The PCP algorithm upends basic notions of evidence and persuasion, and accomplishes what is usually philosophy's prerogative: to turn the comprehended into the incomprehensible. Somewhere, Wittgenstein must be smiling.

Moore's Law has put computing on the map. Algorithms will now unleash its true potential. Physics, astronomy, and chemistry are all sciences of *formulae*. Chaos theory moved the algorithmic zinger to centerstage. The quantitative sciences of the 21st century (eg, genomics, neurobiology) will complete the dethronement of the *formula* by placing the *algorithm* at the core of their modus operandi. Algorithmic thinking is likely to cause the most disruptive paradigm shift in the sciences since quantum mechanics. And yes, you may trust the future to be kind to this prediction.

Sensor Networks: A Digital Bridge to the Physical World



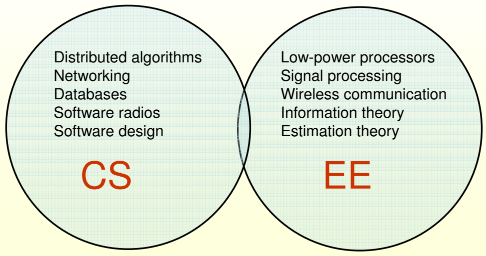
Leonidas Guibas
Computer Science Dept.
Stanford University



NHC Spring School and Workshop on Discrete Algorithms

1

Introduction



Many good algorithmic and theoretical questions!

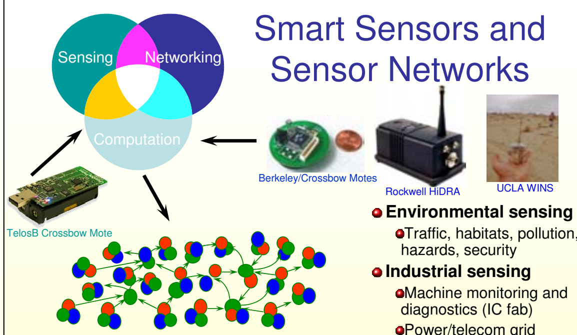
2

Lecture Outline

- **Part 1:** Introduction to Wireless Sensor Networks
  - Distributed monitoring applications; Sensor network hardware; Research issues in sensor networks; Naming and routing; Sensor tasking and control.
- **Part 2:** Structure Discovery and Information Brokerage
  - Morphological analysis (boundaries, holes, bridges); Landmarks and local coordinates; Information diffusion; Hierarchical hashing.
- **Part 3:** Lightweight Spatio-Temporal Reasoning
  - Configuration spaces; Collaboration groups; Identity management; Occupancy tracking; Conclusion

3

Smart Sensors and Sensor Networks



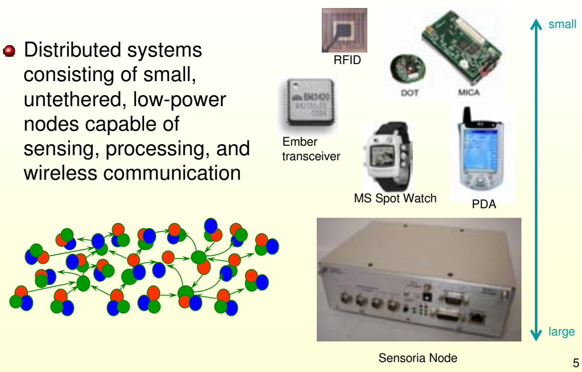
Untethered micro sensors will go anywhere and measure anything -- traffic flow, water level, number of people walking by, temperature. This is developing into something like a nervous system for the earth. -- **Horst Stormer in *Business Week, 8/23-30, 1999.***

- **Environmental sensing**
  - Traffic, habitats, pollution, hazards, security
- **Industrial sensing**
  - Machine monitoring and diagnostics (IC fab)
  - Power/telecom grid monitoring
- **Human-centered computing**
  - Smart, human-aware spaces and environments

4

Wireless Sensor Networks

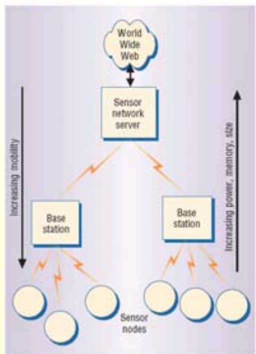
- Distributed systems consisting of small, untethered, low-power nodes capable of sensing, processing, and wireless communication



5

Monitoring the World

- Monitoring the environment and other spaces
- Monitoring objects
- Monitoring interactions between objects, or between objects and their environment



6

Petrel Nesting Behavior at Great Duck Island

7

Wireless Sensor Network Deployment

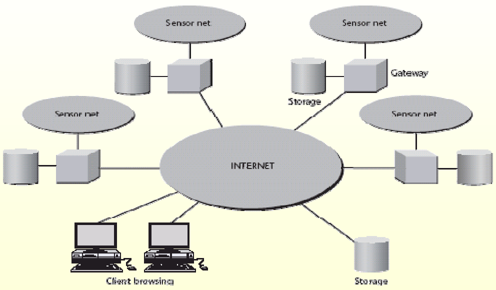
- Advantages:
  - sensors can be close to signal sources, yielding high SNR
  - phenomena can be monitored that are widely distributed across space and time
    - a `macroscope' [Culler]
  - a distributed architecture provides for scalable, robust and self-repairing systems
  - significant installation savings on cabling, etc. are possible

British Columbia winery with networked temperature sensors

Other data collection and monitoring: temperature in data centers (HP), oil tanker vibrations (BP/Intel), soil contaminants, etc.

8

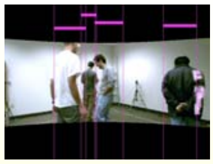
Integration with Current Networks



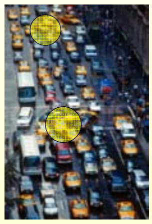
Access to unfiltered information, highly localized in time and space.
Plans for next generation Internet all include edge sensor networks.

9

More Demanding Sensor Network Applications



- Beyond simple data collection and aggregation
  - dynamic, mobile foci of activity (tracking mobile objects)
  - Amidst clutter of irrelevant data
  - distributed attention: focus and context
  - acting on the world (closing the loop)
- Network must adapt to highly dynamic foci of activity
- Sensing is driven by user queries
- Sensing and communication tasks must be planned and allocated
- Resources must be apportioned between detection, tracking, etc.



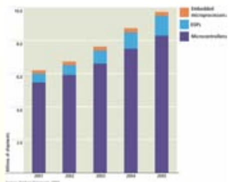
10

Sensor Network Hardware

11

Wireless Sensor Trends

- Of 9.6 billion µP's shipped in 2005, 98% were embedded processors!



- Riding on Moore's law, smart sensors get:

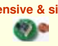
More powerful

Sensoria WINSNG 2.0
CPU: 300 MIPS
1.1 GFLOP FPU
32MB Flash
32MB RAM
Sensors: external

Easy to use

HP iPAQ w/802.11
CPU: 240 MIPS
32MB Flash
64MB RAM
Both integrated and off-board sensors

Inexpensive & simple



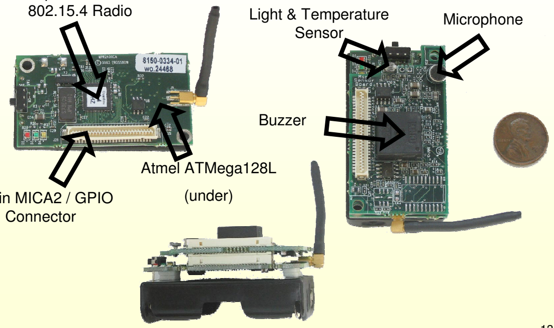
Crossbow Mica2dot mote
4 MIPS CPU (integer only)
8KB Flash
512B RAM
Sensors: on board stack (accel, light , microphone)

Supercheap & tiny

Smart dust (in progress)
CPU, Memory: TBD (LESS!)
Sensors: integrated

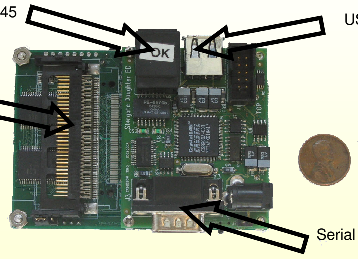
12

Currently Popular: Crossbow Motes



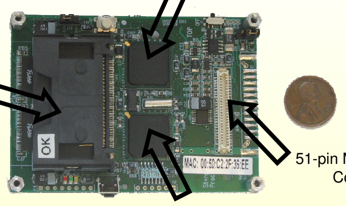
13

Crossbow Stargate - Top View



14

Crossbow Stargate - Bottom View



15

Specifications

MicaZ Mote

- TinyOS
- 16 Mhz Atmel ATMega128L
- 128 kB Program FLASH
- 512 kB Serial FLASH
- Current Draw
  - 8 mA – Active Mode
  - <15 uA – Sleep Mode
- Chipcon CC2420 802.15.4 Radio
  - 250 kbps
  - 26 Channels – 2.4 Ghz
  - Current Draw – 15 mA

Stargate

- Embedded Linux OS
- 400 Mhz Intel Xscale
- 64 MB SDRAM
- 32 MB FLASH
- Many different interfaces
  - RS-232, Ethernet, USB,…

www.xbow.com
http://computer.howstuffworks.com/mote4.htm

16

Power Breakdown …

	Active	Idle	Sleep
CPU	5 mA	2 mA	5 μA
Radio	7 mA (TX)	4.5 mA (RX)	5 μA
EE-Prom	3 mA	0	0
LED's	4 mA	0	0
Photo Diode	200 μA	0	0
Temperature	200 μA	0	0

Rene motes data, Jason Hill

Computation/communication ratio per byte:

- Rene motes:
  - Comm: (7mA*3V/10e3)*8=16.8μJ per 8 bit
  - Comp: 5mA*3V/4e6=3.8 nJ per instruction
  - Ratio: 4,400 instructions/hop
- Sensoria nodes:
  - Comm: (100mW/56e3)*32=58μJ per 32 bit
  - Comp: 750mW/1.1e9=0.7nJ per instruction
  - Ratio: 82,000 instructions/hop

Panasonic CR2354
560 mAh

This means

– Lithium Battery runs for 35 hours at peak load and years at minimum load, a three orders of magnitude difference!

17

Architectural Challenges

18

Sensor Network Challenges

- Power management
  - communication 1000s of times more expensive than computation
  - load balancing across nodes
  - coordinated sleeping/awake schedules
  - correlated sensor data
- In-network processing
  - data aggregation
  - overcounting of evidence
- Difficult calibration
  - localization
  - time-synchronization
- Constant variability
  - networking
  - sensing

[Picture from CACM June 2004]

19

Dense Sampling: Multi-Hop RF Advantage

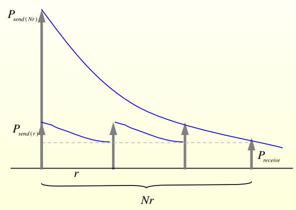
RF power attenuation near ground:

$$P_{receive} \propto \frac{P_{send}}{r^{\alpha}}, \qquad \alpha : 3-5$$

Or equivalently, $P_{send} \propto r^{\alpha} P_{receive}$

Power advantage:

$$\frac{P_{send(Nr)}}{N \cdot P_{send(r)}} = \frac{(Nr)^{\alpha} P_{receive}}{N \cdot r^{\alpha} P_{receive}} = N^{\alpha-1}$$



Oversimplified: fixed overheads, delay, etc.

20

Dense Sampling: Detection and SNR Advantage

Sensors have a finite sensing range. A denser sensor field improves the odds of detecting a target within the range. Once inside the range, further increasing sensor density by *N* improves the SNR by 10log*N* db (in 2D). Consider the acoustic sensing case:

Acoustic power received at distance *r*: $P_{receive} \propto \frac{P_{source}}{r^2}$

Signal-noise ratio (SNR):

$$SNR_r = 10\log P_{receive} - 10\log P_{noise} = 10\log P_{source} - 10\log P_{noise} - 20\log r$$

Increasing the sensor density by a factor of *N* gives a SNR advantage of:

$$SNR_{\frac{r}{\sqrt{N}}} - SNR_r = 20\log\frac{r}{\frac{r}{\sqrt{N}}} = 10\log N$$

21

Collaborative Estimation

- Structuring communication is very important:

  In a setting where each node wishes to communicate some data to another node at random, interference hinders scaling:

  the per node throughput scales as $\frac{1}{\sqrt{N}}$ (Gupta & Kumar '99)

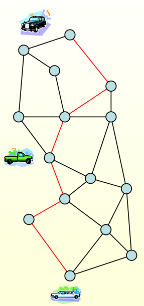
  Effectively each node is using all of its energy to route messages for other nodes.

  In a sensor network, however, because data from nearby sensors are highly correlated, more intelligent information dissemination strategies are possible.

22

Networking Sensor Networks

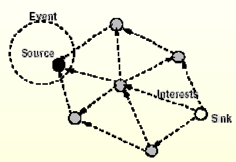
- Network support for a small number of collaborative tasks.
- Data-centric, (as opposed to a node-centric) view of the world.
- Monitoring processes may migrate from node to node, as the phenomena of interest move or evolve.
- Communication flow and structure is dictated by the geography of signal landscapes and the overall network task.



23

`Semantic' Routing and Networking

- We want to address spatial locations or information, not individual nodes
- Content and address in a message get intermixed – unlike classical networks
- In a distributed setting, how do we help information providers and information consumers find each other?

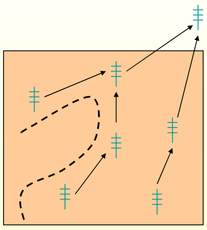


Directed diffusion
Geo-routing

24

In-Network Processing

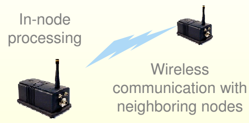
- Information aggregation can happen on the way to a destination and provide substantial energy savings
- Need to balance quality of paths with quality of information collected
- But aggregation makes data lineage harder to ascertain
- Can we have "application-independent" paradigms of information aggregation?



Temperature aggregation

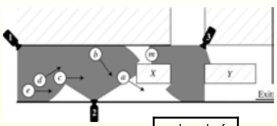
Power-Aware Sensing, Computing, and Communication

- Variable power systems
- Let most sensors sleep most of the time; use paging channels
- Exploit correlation in readings between nearby sensors
- Load-balance, to avoid depleting critical nodes



Sensor Tasking and Control

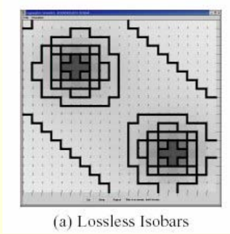
- Decide which sensors should sense and communicate, according to the high-level task – a non-trivial algorithmic problem
- Direct sensing of relations relevant to the task – do not estimate full world state



a ahead-of *b*
b ahead-of *c*
c ahead-of *d*
d ahead-of *e*

Enable Data-Base Like Operations

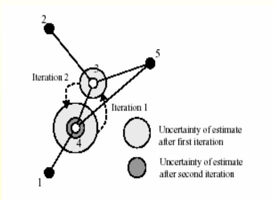
- Data only available right after sensing operation
- Dense data streams must be sampled, or otherwise summarized
- Must deal with distributed information storage – "where is the data?"
- Large flash memory availability can make in-network storage possible



Field isolines

Self-Configuration for Ad-Hoc Deployment

- Network size makes it impossible to configure each node individually
- Environmental changes may require frequent re-calibration
- Network must recover after node failures



Iterative localization

Structure Discovery

- A sensor network is a novel type of computing device -- a sensor computer
- One of its first tasks is to discover its own structure and establish
  - information highways
  - sensor collaboration groups
- as well as adapt to its signal landscape

New System Architectures

- Resource constraints require close coupling between the networking application layers
- Can we define *application-independent* programming abstractions for sensor networks?

A sensor net stack?

User queries, external databases
In-network: application processing, data aggregation, query processing
Data dissemination, storage, caching
Adaptive topology, geo-routing
MAC, time and location services
Phy: comm, sensing, actuation, SP

31

Various Issues

- Integration of sensors with widely different modalities
  - High data-rate sensors (cameras, laser scanners)
- Sensor mobility
- Actuation

Distributed robotics

32

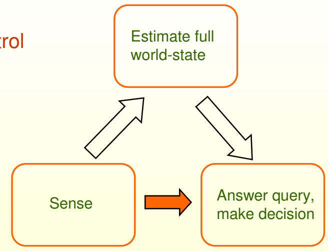
What Defines Sensor Networks?

- Multi-hop communication
  - Many nodes act as routers
  - Multiple paths exist and must be considered
- Bandwidth limitations
  - Volume of data sensed exceeds to capacity of the network to transport
- Power limitations
  - (At least some) nodes operate untethered and energy conservation must be considered in all of sensing, processing, and communication
- A cooperative system
  - All nodes serve one, or a small number of tasks

33

Sensor Network Research

- power awareness
- sensor tasking and control
- formation of sensor collaboration groups
- in-network, distributed processing
- node management, service establishment, software layers
- coping with noise and uncertainty in the environment



A key algorithmic problem is how to sense and aggregate only the portions of the world-state relevant to the task at hand, in a lightweight, energy-efficient manner.

34

Naming and Routing in Sensor Networks

35

Routing in Sensor Networks

- Point-to-point routing protocols in communication networks obtain route information between pairs of nodes wishing to communicate. Such protocols can be
  - proactive: the protocol maintains routing tables at each node that are updated as changes in the network topology are detected
  - reactive: the protocol constructs paths on demand only
- Because of the high rate of topology changes, reactive protocols are much more appropriate for sensor networks
- Several such protocols have already been developed for ad hoc mobile communication networks. Examples are:
  - Ad hoc on demand distance vector routing (AODV)
  - Dynamic source routing (DSR)
    - both, however, may flood the network to discover paths

36

Geographic Routing

- In sensor networks, naming and routing is frequently based on a node's attributes and sensed data, rather that on some pre-assigned network address.
- Geographic routing uses a node's location to name the node and discover paths to that node
- We assume that
  - nodes know their geographic location
  - nodes know their 1-hop neighbors
  - routing destinations are specified geographically (a point, a region)
  - each packet can hold a small amount ($O(1)$) of additional routing info to record where it has been on the network
  - most of the time we will model the connectivity graph of the nodes as a unit distance graph

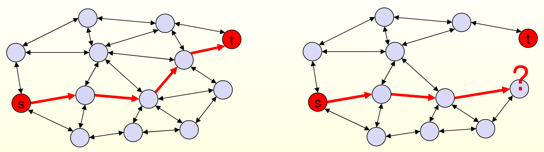
37

Routing Desiderata

- Guaranteed delivery
- Path quality
- Energy awareness
- Robustness to low-level link volatility

38

Greedy Methods



In a greedy method, each node forwards a packet to its best neighbor

Greedy methods can get stuck at "dead-ends"

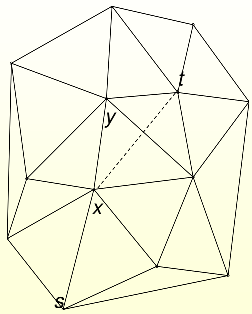
Note that no flooding is involved for route discovery

39

Greedy Unicast Geographic Routing

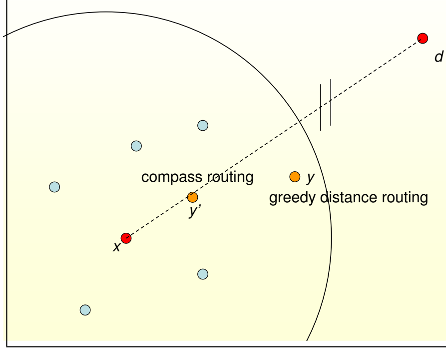
To go from source *s* to destination *t*, at each intermediate node *x* advance to the neighbor *y* that makes most progress towards *t*.

- greedy distance routing (GPSR)
- compass routing



40

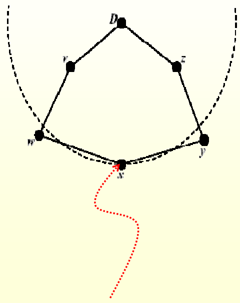
Neighbor Choice



41

Greedy Protocols Can Get Stuck

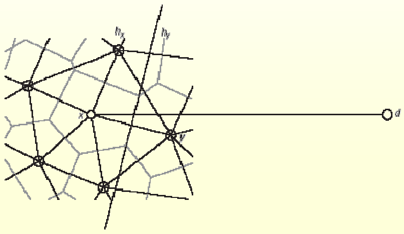
- The intermediate node *x* can be a local optimum towards the destination
- In general, local optima will arise if the node graph contains "holes" – areas with no sensor nodes
- To prove that such situations cannot happen we need to assume special properties about the connectivity graph *G*



42

Delaunay Triangulations (DT)

- In a Delaunay triangulation (dual to the Voronoi diagram of the nodes), packets cannot get stuck
- However, unless the nodes are spaced very closely, it is unlikely that the UDG will contain all DT edges



43

Measures of Path Quality

- First and foremost, a protocol should guarantee packed delivery, whenever such delivery is possible
- Second, the quality of the path produced should be good when compared to the optimal path available. Different path costs can be used:

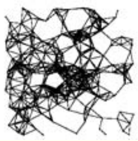
$$c(\pi) = \sum_{e \in \pi} l^d(e), \quad d = 0,1,2,3,4,\ldots$$

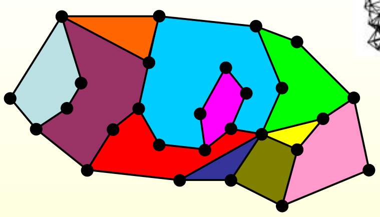
d = 0, hop length
d = 1, normal path length
d = 2, 3, 4 ..., energy costs

- These can be made roughly equivalent by assuming a constant node density or a minimum node spacing
  - This can be attained by a node clustering process

44

Planarizations

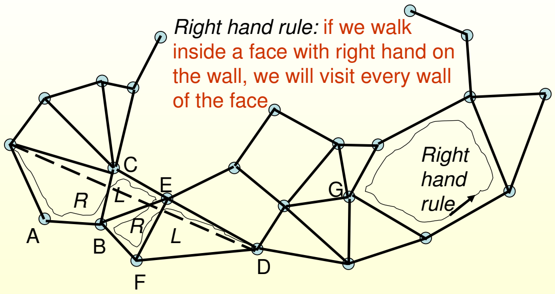




A planar straight-line graph has no crossing edges. It subdivides the plane into regions called faces.

45

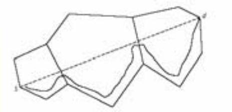
Traversing Planar Graphs: Perimeter Routing

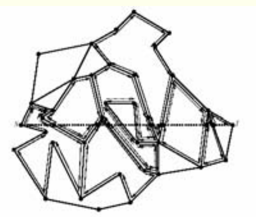


46

Routing in Planar Graphs

- To guarantee packet delivery, *it may be advantageous to disable some connections*, so as to make the routing graph planar
- On a planar graph, perimeter routing guarantees delivery
- Another variant is other face routing
- The quality of paths can be bad, however

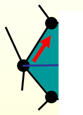


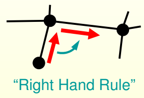


47

Perimeter/Face Routing Properties

- All necessary information is stored in the message
  - Source and destination positions are given
  - Point of transition to next face needs to be chosen

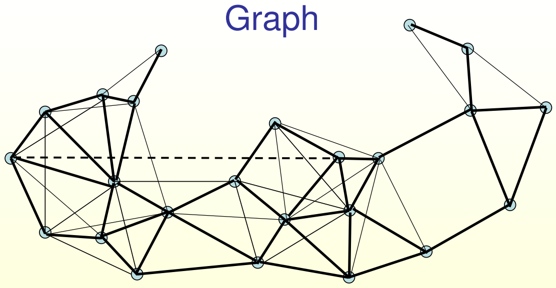




- Completely local:
  - Knowledge about direct neighbors' positions sufficient
  - Faces are implicit, only local neighbor ordering around each node is needed

48

Planarizing the Communication Graph



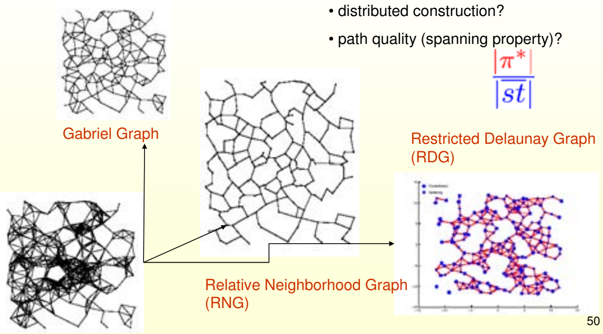
We must delete just enough edges to make the graph planar

We must do so in as local a manner as possible

49

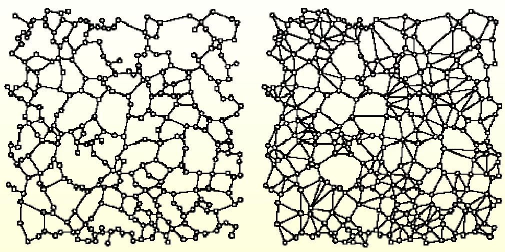
Planarization via Geometric Graphs

- defined by local rules?
- distributed construction?
- path quality (spanning property)?

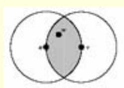


50

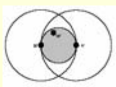
Larger RNG and GB Examples



Relative Neighborhood Graph



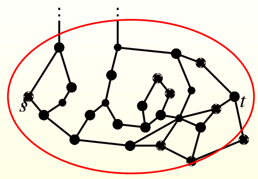
Gabriel Graph



51

Adaptive Algorithms

- We want the quality of paths we discover to be nearly optimal
- Alternatively, we want to discover optimal paths without searching the whole connectivity graph *G*
- If the optimal path between *s* and *t* has length *L*, then every node in that path is within an ellipse with foci *s* and t defined by *L*. This ellipse limits the part of *G* to be searched.
- If *L* is not known, it can be guessed, approximately

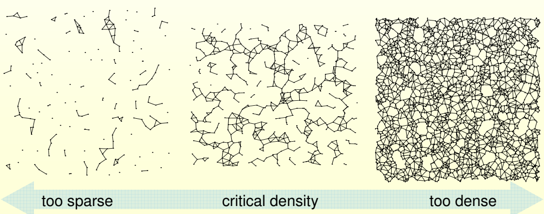


In general, finding a path of length *L* requires $O(L^2)$ work.

52

Average Path Quality

- Not interesting when graph not dense enough
- Not interesting when graph is too dense
- Critical density range ("percolation theory")
  - Shortest path is significantly longer than Euclidean distance

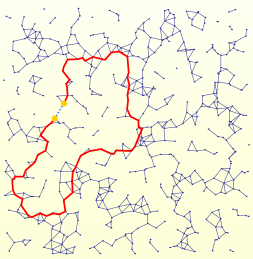


53

Critical Density: Shortest Path vs. Euclidean Distance

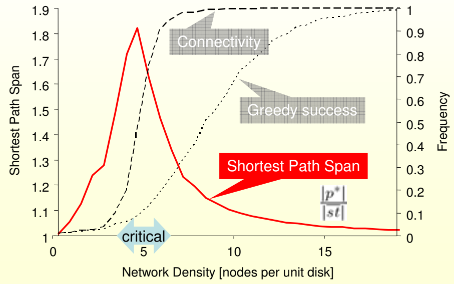
- Shortest path is significantly longer than Euclidean distance

$$\frac{|\pi^*|}{|st|}$$



54

Randomly Generated Graphs: Critical Density Range



55

Combining Greedy and Planarization Strategies: Greedy Perimeter Stateless Routing (GPSR)

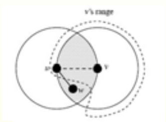
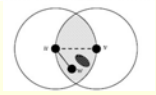
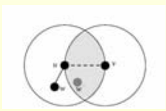
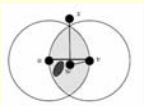
[Bose and Morin 1999,
Karp and Kung, Mobicom 2000]:
Planarize the connectivity graph *G*

- Use greedy distance protocol on the full graph *G*
- If stuck, switch to perimeter protocol on a planarization of *G*, until a node closer to the destination than the stuck node is encountered

56

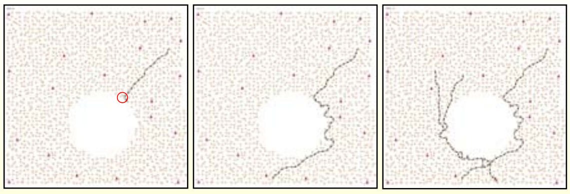
Planarization Process Is Not Robust

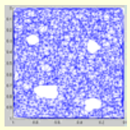
- Gabriel and RNG depend on the unit disk communication graph assumption
- Localization errors also cause serious problems

57

Greedy Protocols May Overload Critical Nodes





GPSR [Karp, Kung, '00]
GOAFR+ [Kuhn, Wattenhofer, Zhang, and Zollinger, '03]

58

Main Point

- Knowledge of the nodes' locations enables many powerful mechanisms for message transport and route discovery that avoid expensive flooding operations yet require no routing tables or other high-maintenance data structures.

59

Sensor Tasking and Control

60

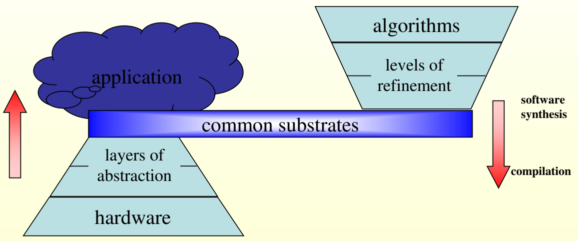
Programming Sensornets: Where the Two Sides Meet

Platform View

Spend more time designing system/component-level abstractions

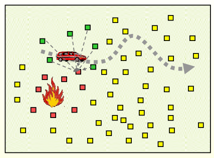
Information Processing View

Spend more time designing application-level abstractions



61

Collaborative Processing in Sensor Networks

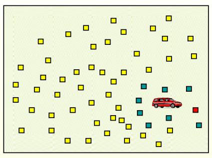


- What information is critical for the high-level tasks?
- What is the cost of accessing the information?
- Which nodes should participate in sensing, processing, or communication?
- How should the information be migrated?
- What is routing or querying in this context?

The information processing needs largely determine the roles of nodes, as well as the required support by other layers of a sensor network

62

Collaborative Processing
Group Formation in Sensor Networks

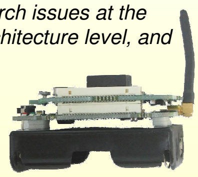


- Information needs and resource constraints define who should participate in the processing groups
- Group membership (e.g. location) defines the behavior of a node
- Challenges
  - Dynamic collaboration among nodes
  - Global property from local execution
  - Competing events/tasks
  - Real-time constraints/adaptation

63

Summary

- *Ubiquitous networked sensors provide a dense spatial and temporal sensing of the physical world*
- *They potentially provide low-latency access to information that is highly localized in time and space, and thus provide a way to sense and act on the physical world beyond what has been possible up to now*
- *Sensor networks raise many research issues at the physical node level, the system architecture level, and the algorithm deployment level*



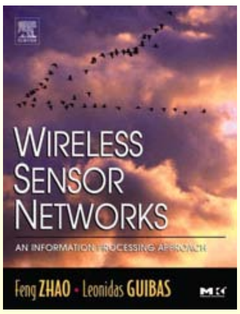
64

A Relevant Text

Wireless Sensor Networks: An Information Processing Approach

Feng Zhao and Leonidas Guibas

Morgan-Kaufmann 2004

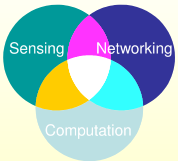


65

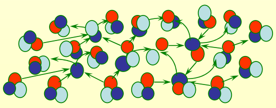
The End

66

Structure Discovery and Information Brokerage in Sensor Networks



Leonidas Guibas
Computer Science Dept.
Stanford University



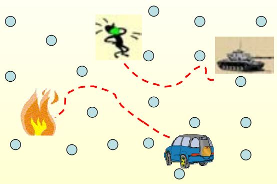
Structure Discovery

- A sensor network is a novel type of computing device -- a sensor computer
- One of its first tasks is to discover its own structure and establish
  - information highways
  - sensor collaboration groups
- as well as adapt to its signal landscape

[From D. Estrin]

Information Brokerage

- Information providers (sources, producers) and information seekers (sinks, consumers) need ways to find out about and rendez-vous with each other
- Challenges:
  - Neither knows where the other is
  - Highly dynamic environment
  - Limited computation and communication resources



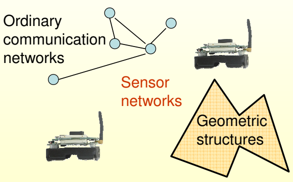
Talk Outline

- Naming and Routing
  - Landmarks and local coordinates
  - Hierarchical landmarks
- Sensor Layout Analysis
  - Boundary/hole detection
- Information Dissemination and Aggregation
  - Sweep
- Information Brokerage
  - Hierarchical geographic hashing
  - Double rulings
  - Information gradients

A Dilemma: Which Structure?

- Nodes are embedded in a physical space. Should we adopt the naming and routing structures already available in the host space?
- Or should we invent a space that better reflects the true network topology, and use that instead?

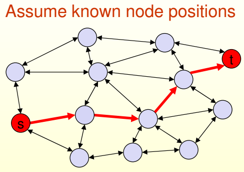
- What if our sampling is bad?



- What if the network is volatile?

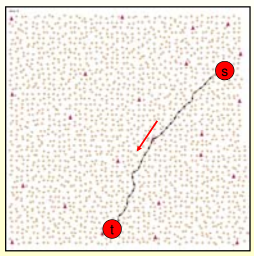
Greedy Geographic Routing

Assume known node positions

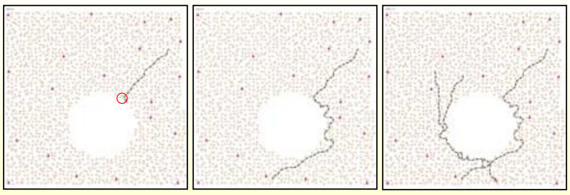


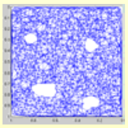
In a greedy manner, each node forwards a packet to its best neighbor

Note that there is little global state, yet no flooding is involved for route discovery



Such Greedy Protocols Get Stuck, May Overload Critical Nodes





GPSR [Karp, Kung, '00]

GOAFR+ [Kuhn, Wattenhofer, Zhang, and Zollinger, '03]

These require building a planar subgraph of the connectivity graph – not a robust process

Global Embedding Challenges

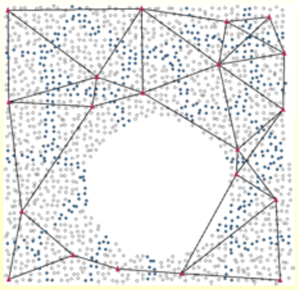
- Routing on geographic coordinates
  - Only works in 2-D space
  - Planarization is tricky (CLDP, etc.)
  - Sensitive to location inaccuracy
- Routing on virtual coordinates
  - Requires a global embedding of the link connectivity graph in the plane
  - Forcing a 2-D layout on a 3-D deployment may ignore much of the actual connectivity

Naming an Routing Based on Connectivity Information Only

- A two-tier approach utilizing combinatorial Delaunay complexes and local coordinates (GLIDER)
- A hierarchical approach using the Discrete Center Hierarchy (DCH)

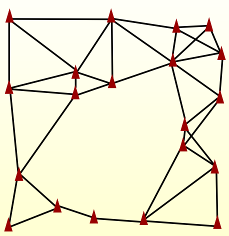
I. Using Landmarks and Local Coordinates: GLIDER

- Given a communication graph on sensor nodes with distances defined by hop counts
- Perform structure discovery:
  - Select a set of landmarks
  - Construct the Landmark Voronoi Complex (LVC)
  - Extract the Combinatorial Delaunay Triangulation (CDT) graph on landmarks



G is connected ⇔ CDT D(L) connected

- D(L) is compact – topology capture has complexity dependent on the number of large-scale features in the environment
- D(L) is stable – low- level link volatility unlikely to affect the combinatorial complex



D(L) is a global network atlas that can be known to all landmarks, or even all nodes

Local Routing with Global Guidance

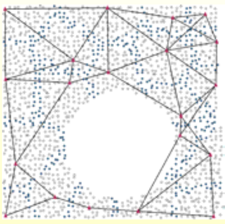
- Global Guidance

  the D(L) encodes global connectivity information that is accessible to every node for proactive route planning on tiles.
- Local Routing

  high-level routes on tiles are realized as actual paths in the network by using local reactive protocols.

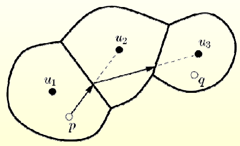
Information Stored at Each Node

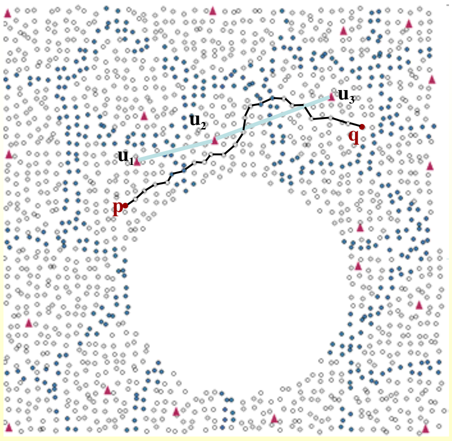
- The parents on the shortest paths to its home landmark, and its neighbor landmarks
- A bit to record if the node is on the boundary of a tile
- Its **coordinates** and those of its neighbors for intra-tile greedy routing
- Landmark nodes store the atlas D(L)



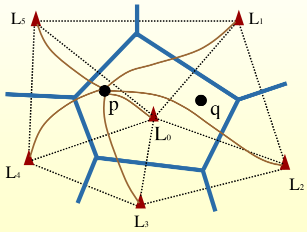
GLIDER -- Routing

- Routing
  - Global route plan
  - Local route
    - inter-tile
    - intra-tile





The Last Mile: Local Coordinates and Greedy Routing



Reference landmarks: $L_0,\ldots L_k$

$T(p) = L_0$

Let $s = \mathrm{mean}(pL_0^2, \ldots, pL_k^2)$

Local virtual coordinates:

$c(p) = (pL_0^2 - s, \ldots, pL_k^2 - s)$

(centered metric)

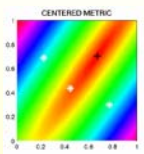
Distance function:

$d(p, q) = |c(p) - c(q)|^2$

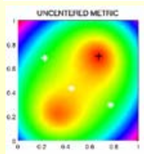
Greedy strategy: to reach q, do gradient descent on the function d(p, q)

Local Landmark Coordinates – No Local Minima

- Theorem: In the continuous Euclidean plane, gradient descent on the function d(p, q) always converges to the destination q, provided that there are at least three non-collinear landmarks.



- In the discrete case, we empirically observe that landmark gradient descending does not get stuck on networks with reasonable density (each node has on average six neighbors or more).

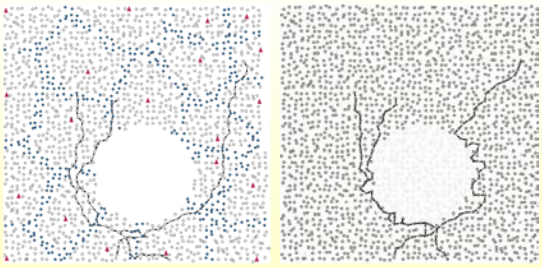


Centered vs. Uncentered Metrics

movie1.mpeg

movie2.mpeg

Simulations – Path Length and Load Balancing



GLIDER GPSR

Each node on average has six one-hop neighbors

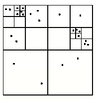
Simulations – Hot Spot Comparison

Randomly pick 45 source and destination pairs, each separated by more than 30 hops.

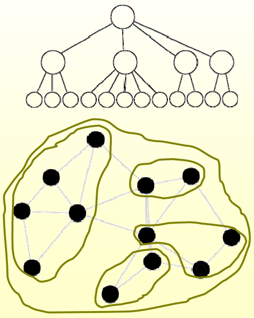
GLIDER GPSR

Blue (6-8 transit paths), orange (9-11 transit paths), black (>11 transit paths)

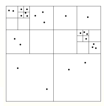
II. Naming and Routing via Hierarchical Decompositions of Graphs



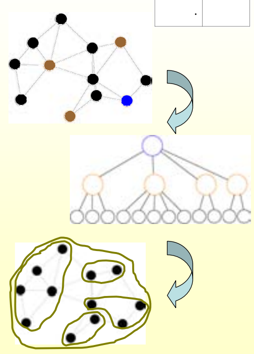
- Generalize quadtrees
- No node geographic coordinates assumed
- Required properties:
  - Clusters in level i of the decomposition have diameter at most $\alpha \cdot 2^i$, where α is a constant
  - Each cluster in level i+1 contains a small (constant) number of clusters in level i



Examples of HDs

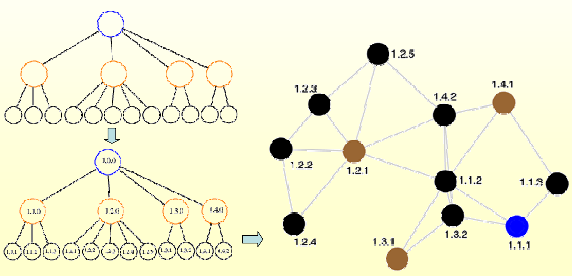


- A quad-tree induces a HD when the sensor field is dense and node coordinates are available.
- Discrete Center Hierarchy:
  - A hierarchical sampling of the nodes so that:
    - Nodes in level i are at least 2^i hops apart
    - Each node in level i is within 2^{i+1} hops from some node in level i+1

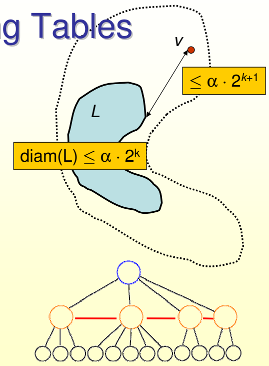


Addressing Scheme

- A HD yields an IP-type addressing scheme for nodes
- Clusters are also assigned addresses



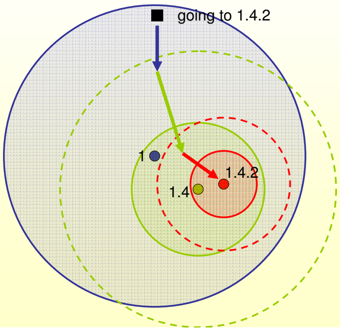
Neighboring Clusters and Local Routing Tables



- Def: A cluster L at level k is a neighboring cluster of a node v if dist(v, L) $\leq \alpha \cdot 2^{k+1}$
- A routing table is stored at each node, providing hop distances to all its neighboring clusters
- Under mild assumptions, each node has $O(\log n)$ neighboring clusters

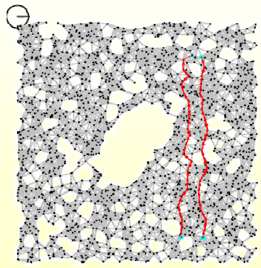
Getting to Your Destination

- Head in the direction of the cluster with the longest prefix that agrees with your destination address
- Use local routing tables to make the best local decision



Routing Scheme

- Routing quality:
  - By proof:
    - Efficient: $|\text{path}(u,v)| \leq 4 \cdot |d_{uv}|$
  - By simulation:
    - Balanced: Nodes high up in the hierarchy do not get overloaded
    - Robust: the failure of any given link does not affect many paths



Experimental Results

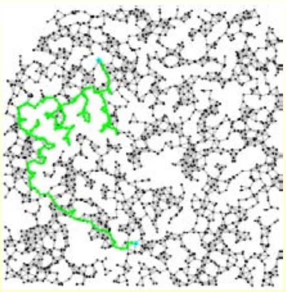
degree 6.21

Routing quality (2000 nodes)

Avg. deg.	Qual. of HD		Qual. of GPSR	
	avg	std	avg	std
6.21	1.08	0.18	4.91	6.79
6.80	1.06	0.11	4.04	7.41
7.39	1.05	0.09	3.25	7.02
7.93	1.09	0.16	2.04	3.10
8.53	1.07	0.12	1.59	2.22
8.94	1.07	0.10	1.51	2.42
9.82	1.07	0.10	1.35	1.62
10.2	1.06	0.09	1.31	1.80
10.8	1.05	0.09	1.44	2.90
12.0	1.06	0.11	1.15	1.30

Paths generated are near optimal

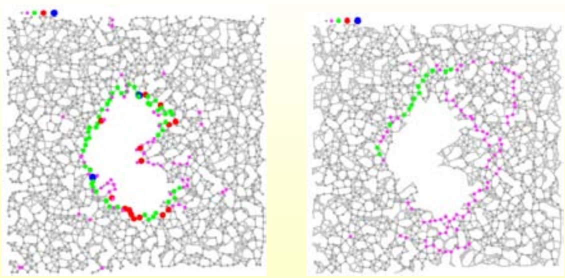
$$\text{Quality} = \frac{\text{HD path length}}{\text{Shortest path length}}$$



HD path — GPSR path Shortest path

Hot Spots

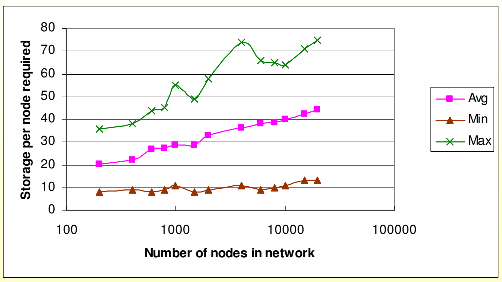
2000 nodes, perturbed grid
100 random paths
max load = 32



GPSR HD

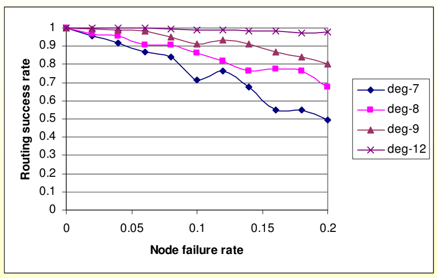
HD does not hug holes as much as GPSR

Routing Scalability



Storage used grows slowly
Network initialization cost ~ storage used

Routing Robustness



Routing performance degrades gracefully as node failure rate increases

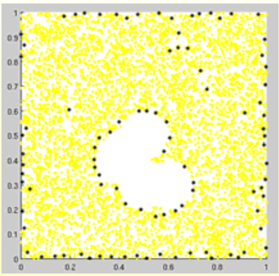
HD Names and Routes Summary

- HD effectively discovers the intrinsic geometry of the network
- Provides a hierarchy-based scheme with provable approximation quality on the routing paths
- Node/link failures affect mostly the low levels of the hierarchy

Sensor Layout Analysis

- Boundary/hole detection
- Robust planarization of the communication graph

Boundary Detection



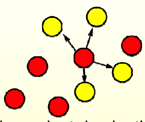
Information Dissemination and Aggregation

- Most sensor network applications need a robust and efficient implementation of certain basic data operations
- In such a data operations library one needs to include:
  - data dissemination (code images, parameter settings, etc.)
  - in-network data aggregation

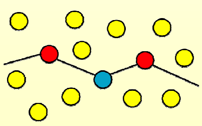
Related Work

Data Dissemination

- Epidemic approaches
  - Trickle/Deluge

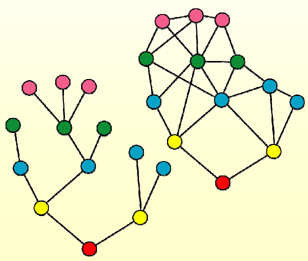


- Independent dominating sets



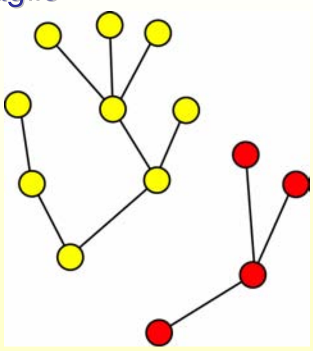
Data Aggregation

- Tree-based approaches
  - TAG/DAG
- Synopsis diffusion



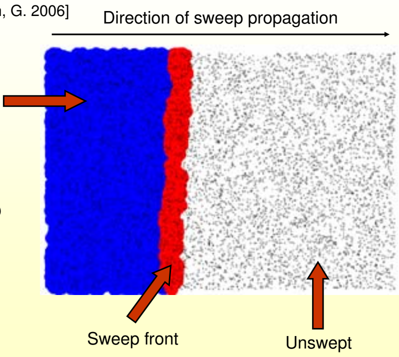
Tree-Based Approaches Are Fragile

- Single links are relied upon
- Long interval between establishment and use



Sweeps over a Sensor Network

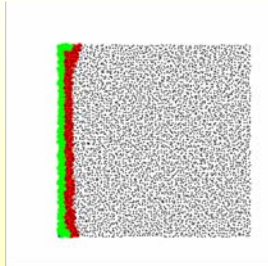
[Skraba, Fang, Nguyen, G. 2006]



- Unswept nodes
- Nodes in sweep
- Swept nodes

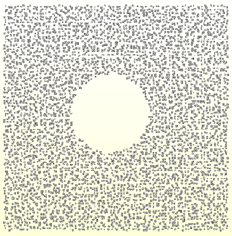
Global Picture

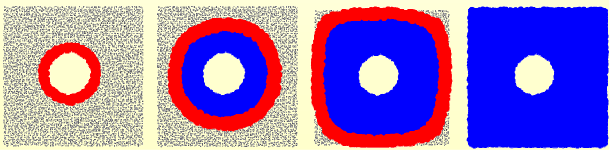
- Nodes are invited to join in the sweep by already active neighbors
- Nodes remove themselves from the sweep after they and all their neighbors have been processed
- A select subset of the nodes always holds the state of the sweep
- Active band moves across the network as nodes join and remove themselves
- Locality of advancing tests allows for parallelism



Sweep Requirements

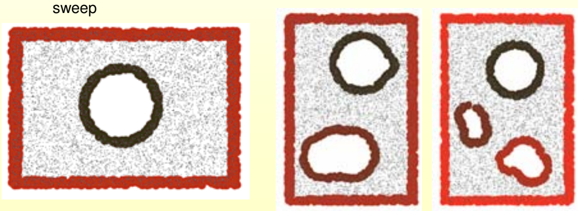
- Cover all nodes in a network, each exactly once
- Use a small active band
- Robust to link volatility
- Local, asynchronous control





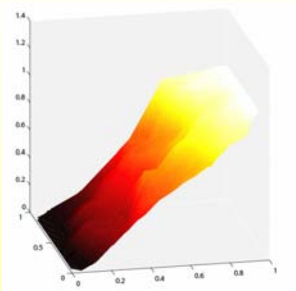
Preprocessing

- Structure discovery
  - Some global information needs to be extracted from the network
- Boundary detection
  - source and sink for the sweep
- Auxiliary info for sweep control



Potential Field for Wavefront Diffusion

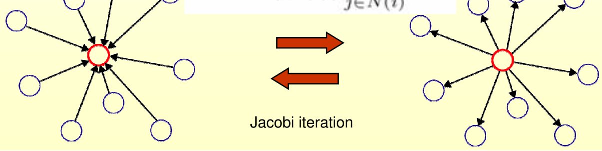
- Part of preprocessing
- Potential field
  - Sense of direction
  - Ordering
- Stable
- Used for many sweeps



Potential Field Construction

- Fix potential values at certain boundary nodes
- Regular nodes iteratively average over neighbors' values

$$\phi(i) \leftarrow \frac{1}{|N(i)|} \sum_{j \in N(i)} \phi(j)$$



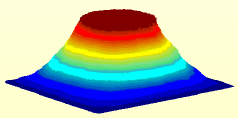
Intuition (Cont. Domain)

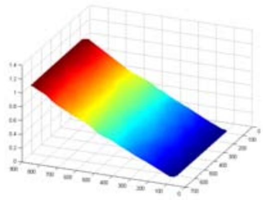
- Laplace's equation with Dirichlet boundary conditions

$$\nabla^2 \phi(i) = 0$$

$$\phi(\text{source}) = 1 \qquad \phi(\text{sink}) = 0$$

- Solving discretized version on communication graph
- Smoothing



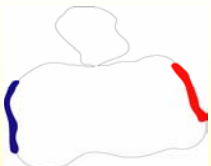


Properties

- Linear system
- Static case: Convergence guaranteed
  - Convergence can be slow (but done only once)
- Direct solution
- No strict maxima or minima at regular nodes
  - Harmonic functions
- Any non-plateau regular node has a strictly monotone path to a maximum or minimum

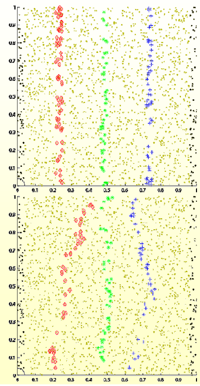
Plateaus

- All neighbors have same value
- Do not occur in continuous case
- Not strict extrema
- Cons:
  - Potential field provides no information
- Pros:
  - Easy to detect
  - Can also be addressed by building another potential within the pleateau
  - Boundary nodes can also be detected



Sweep Algorithm

- Begin *Unswept*
- Receive invitation
  - Enter *Sweep* and send data to inviter (aggregator)
- Possibly issue new invitations
- If all upstream neighbors have left the sweep
  - Forward data to downstream neighbor
- Leave *Sweep*



Simulations

- Stability of the potential field
- Robustness of the sweep
- 500 nodes
- Varying degrees of connectivity

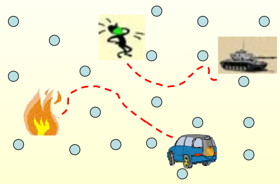
- TOSSIM implementation
  - 20x20 grid
  - Different radio models
  - Robust to link failures/collisions

Conclusions

- Class of global operations on a WSN
- Still uses a transient local tree
  - But the tree is local and is used soon after it is built
- Two-part solution
  - Potential field
    - Captures link connectivity and global structure
    - Relatively stable
  - Sweep
    - Uses the potential field for local control
    - Robust
    - Easily extendible

Information Brokerage

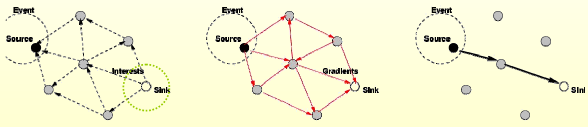
- Information providers (sources, producers) and information seekers (sinks, consumers) need ways to find out about and rendez-vous with each other
- Challenges:
  - Neither knows where the other is
  - Highly dynamic environment
  - Limited computation and communication resources



Current Approaches: Directed Diffusion

[Intanagonwiwat, Govindan, Estrin '00]

- Data-centric storage: data is named by attributes

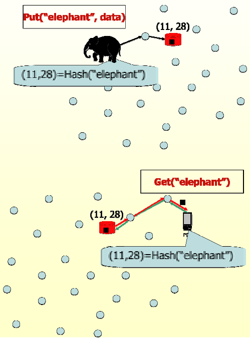


(a) Interest propagation (b) Initial gradients set up (c) Data delivery along reinforced path

Current Approaches: Geographic Hash Tables (GHT)

[Ratnasamy, Karp, Shenker, Estrin, Govindan, Yin, Yu '03]

- Event data is stored, by name, at home nodes; home nodes are selected by the named attributes, via a hash function
- Queries also go to the home nodes to retrieve the data (instead of to the nodes that detected the events)
- Routing usually done using a geographic routing protocol (GPSR)

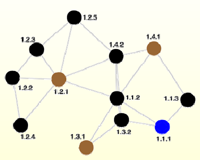


Information Brokerage Issues

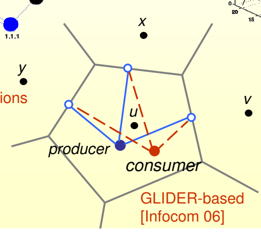
- Find a good balance between cost of information replication (storage size) and cost of information discovery (query time)
- Load balance
- Robustness
- Information brokerage is intimately coupled with
  - how network nodes are named
    - do we have coordinates?
  - how routing is done in the network

Distance-Sensitive Information Brokerage:
if producer and consumer are at a distance d, the query cost should be O(d)

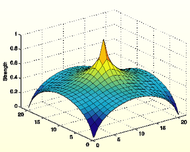
Approaches to Efficient Information Brokerage



Hierarchical Decompositions of Graphs



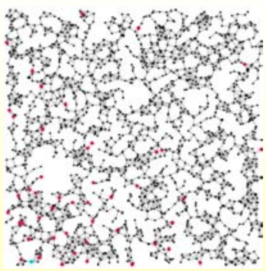
GLIDER-based [Infocom 06]



Information Gradients

Information Diffusion

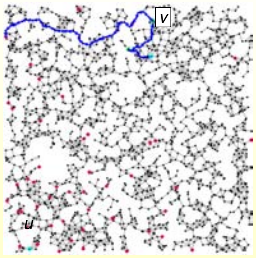
- Hash function: Generates a random valid address in a given cluster for any information type
  $\mu: \Sigma \times HD \rightarrow S$
- A producer hashes its information to all of its neighboring clusters
  - O(log n) hashes
  - Total path length to all hashes is O(D), where D is the diameter of the sensor field



A producer hashes its information to many nearby nodes and to few far away nodes.

Information Retrieval

- Consumer *v* looking for a particular information examines hash locations of that information in larger and larger clusters containing *v*
- Thm: The length of retrieval path from *v* is at most $4 \cdot d_{uv}$ where *u* is the producer (unknown to *v*)



Brokerage Costs

Size	# Info. servers		Query time	
	avg	std	avg	std
200	9.95	0.21	1.08	0.66
400	15.9	0.26	0.89	0.67
600	25.4	1.02	0.77	0.57
1000	31.5	4.00	0.83	0.57
2000	47.0	7.47	0.73	0.47
4000	61.2	11.3	0.73	0.42
6000	69.5	13.6	0.79	0.46
8000	79.0	16.5	0.77	0.39
10000	84.3	15.6	0.76	0.39

Storage cost grows slowly
Cost of query is distance sensitive

$$\text{Query time} = \frac{\text{Path length from consumer to hash location}}{\text{Shortest path length to producer}}$$

HD Brokerage Summary

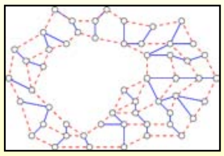
- HD effectively discovers the intrinsic geometry of the network
- Provides a hierarchy-based scheme with provable approximation quality on the routing paths
- Node/link failures affect mostly the low levels of the hierarchy
- Enables distance sensitive information brokerage

Resource Discovery Using Local Double Rulings

Associate with each node two connected `1-d' structures, call them roads – the red and blue

- All red roads together cover the network in a load-balanced fashion
- All blue roads together cover the network in a load-balanced fashion
- For any pair of nodes A and B, the red road from A has to intersect the blue road from B

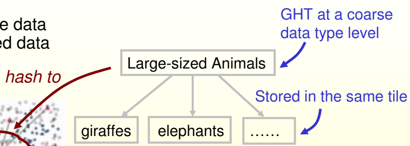
A double ruling derived via a Morse function ≡ distance to boundary

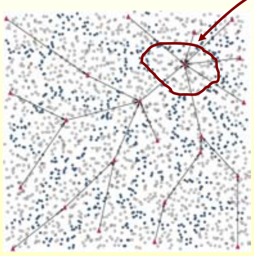


Information providers and seekers can meet by following blue and red roads respectively

GLIDE-based Brokerage: At the CDT Level – Do Content-Based GHT

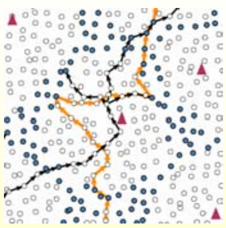
- Hashing on coarse data types for structured data storage





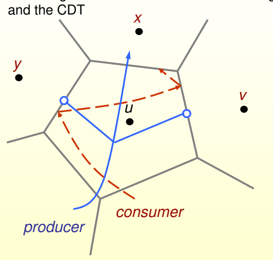
- Both producers and consumers of the same content type follow the shortest path tree to the hashed tile (the root of the tree).
- Consumers return once the data are retrieved, otherwise move on towards the hashed tile.

Within Each Tile – Double-Ruling Scheme for Transit Tiles



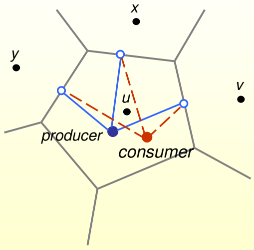
An example by simulation

Guides *v, x, y* are landmarks selected according to a set of rules based on hashing and the CDT



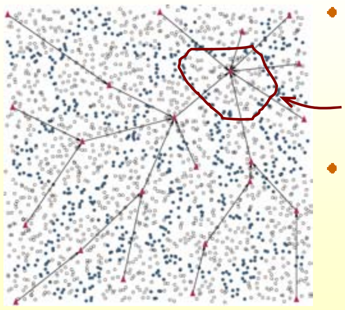
- Routes formed by following shortest paths to guides
- The two sets of curves always meet

Double-Ruling Scheme in Hashed Tile



- Producers and consumers are guaranteed to meet by following the two sets of curves.
- The consumers may not need to reach the hashed tile to fetch the data as the data are available at some transit tiles.

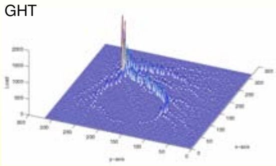
Reducing Producer Cost – *En Route* Data Aggregation



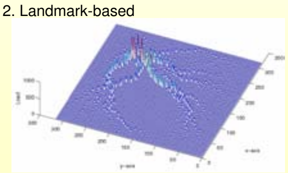
- Producers of the same content type share the shortest path tree (on CDT) rooted at the hashed tile.
- Data of the same type can be aggregated
  - Inside the tile if two producers share one
  - Inside the tile of their common ancestors

Locality Awareness Comparison with GHT by Simulations – Transmission Cost by Individual Node

1. GHT



2. Landmark-based



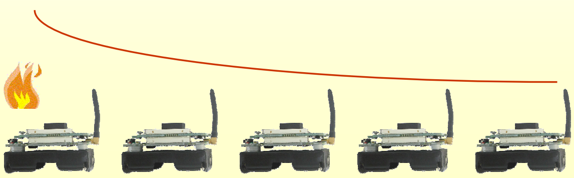
- Scenario: one producer; all nodes query for the producer data; one big hole in the network connectivity graph.
- Note the y-scale in figure 1 is twice of that in figure 2.
- The total load is much lower than using GHT.
- The load is also more balanced than using GHT.

GLIDER Brokerage Summary

- Distance-sensitive information brokerage is possible with very modest data replication
- Information discovery is closely coupled with the network node naming and routing
- In some ways, geometric methods and tools can be effectively used even when the connectivity graph is all we got

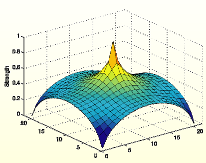
Information Potentials

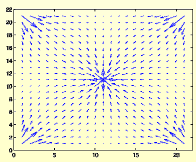
- Natural phenomena typically generate continuous fields (temperature, pressure ...)
- But it can be advantageous to also invent artificial potentials that diffuse information about event detections



Information Diffusion

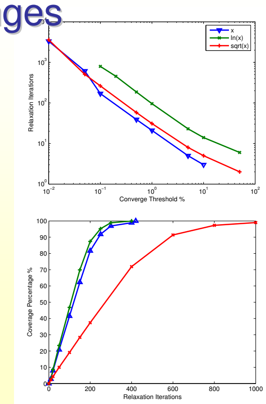
- Information sources can diffuse a quantity that we can think of as information strength via Laplace's equation (Dirichlet boundary c.)
- $\nabla^2\Phi(x) = 0$
- $\Phi(u) = \frac{1}{d(u)}\Sigma_{v\in N(u)}\Phi(v)$
- Information seekers can ascend the gradient of this potential to find a source
- A harmonic function Φ has no local maxima or minima – its gradient can guide a packet, or a vehicle, to its maximum
- Usually smooth Φ by computing a square root, or logarithm





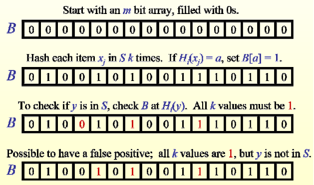
Diffusion Challenges

- Diffusion can be slow to converge
- What if there are multiple sources with the same type of information?
- What if there are many different types of sources?
- What about discretization effects?



Dealing with Many Potential Sources: Bloom Filters for Membership Testing

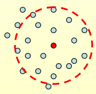
- Given a set $S = \{x_1, x_2, x_3, \ldots x_n\}$ on a universe U, want to answer queries of the form: Is $y \in S$?
- Example: a set of detection attributes
- Bloom filter provides an answer in
  - "Constant" time (time to hash).
  - Small amount of space.
  - But with some probability of being wrong.



Dealing with Many Potential Sources: Network Coding to Save Storage

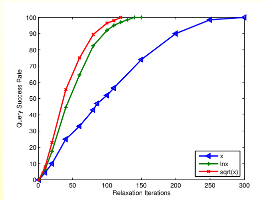
- Each node can compute a random linear combination of all the potentials it hears

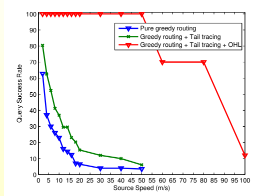
$$\sum_i \lambda_i \Phi_i$$



- If there are L active potential sources, then a neighborhood of size k around a node provides $\Omega(k^2)$ equations relating the local potential values
- But in a neighborhood of size k, the $O(k)$ boundary values determine all the interior values for each potential (harmonic function property)
- So we have enough equations to recover the unknown potentials if $k > L$ (k^2 constraints vs. kL unknowns)

Diffusion Brokerage Performance





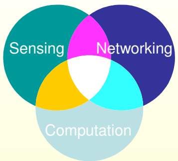
Information Diffusion Summary

- Diffused information potentials can guide both virtual and physical information seekers to the appropriate sources
- Multiple sources can be handled by having their potentials co-mingled and then decoded as necessary
- Sources may move and the potentials adapt in a smooth manner

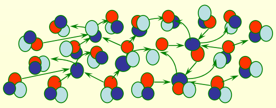
Conclusions

- Structure discovery and information brokerage are fundamental problems for WSNs
- With light preprocessing we can extract certain global quantities that can significantly help with local decisions
- These quantities reflect an understanding of the geometry or topology of the sensor filed and do not require localization
- The same quantities are also robust to local volatility in the network connectivity
- Such approaches integrate very well with current `narrow waist' sensor net protocols, such as SP (Berkeley)

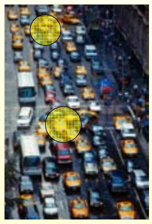
Lightweight Spatio-Temporal Reasoning in Sensor Networks



Leonidas Guibas
Computer Science Dept.
Stanford University



More Demanding Sensor Network Applications



- Beyond simple data collection and aggregation
  - dynamic, mobile foci of activity (tracking mobile objects)
  - Amidst clutter of irrelevant data
  - distributed attention: focus and context
  - acting on the world (closing the loop)
- Network must adapt to highly dynamic foci of activity
- Sensing is driven by user queries
- Sensing and communication tasks must be planned and allocated
- Resources must be apportioned between detection, tracking, etc.

Five Quick Vignettes on Lightweight Spatiotemporal Reasoning

1. Tracking Wide-Area Phenomena
2. Counting Moving Objects
3. Distributed Identity Management
4. Tracking Spatial Occupancy

Sensor selection

Sensor collaboration groups

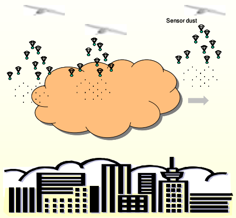
Network structure discovery

Lightweight information integration

Uncertainty and multiple hypotheses

V1. Large-Area Phenomena

- A chemical plume blows over a city after a factory leak.
- Networked sensors are air-dropped to track its extent and motion.
- The sensors organize themselves as appropriate, in order to perform this task.



Tracking a Large Shadow

We track a half-plane shadow across a field with light sensors.

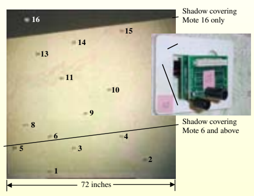
A projector is used to throw a moving shadow onto a wall.

Sensor nodes are Berkeley/Cossbow motes with a radio and a light detector

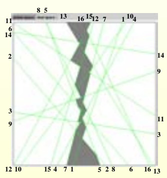
shadow

[Liu, Cheung, G., Zhao, WSNA '02]

Using Geometric Duality Points ↔ Lines



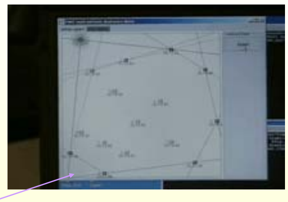
In a dual configuration space, the sensors become lines and the edge of the shadow a moving point, crossing cells in the arrangement of the lines.



Shaded cells indicate the trajectory of the shadow edge.

Selective Sensor Activation

primalspace



dualspace

Dual view

Primal view

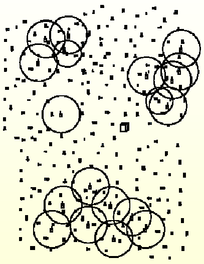
Only sensors dual to the lines bounding the current cell can be crossed next.

Thus, of all the sensors in the field, only four on the average need be active at once.

Some Lessons and Issues

- An appropriately chosen *configuration space* can transform a wide-area phenomenon into a localized one.
- Only a small fraction of the nodes in the field need be active at any one time.
- Most message traffic is along the shadow boundary: *the physical phenomenon dictates the communication paths*.

V2. Target Counting



[Fang, Zhao, G., MobiHoc'03]

Problem Spec:

- A sensor network with multiple targets present. Targets can be stationary or moving;
- Each sensor can detect the local superimposed amplitude of target signals (e.g., acoustic) at any instant of time;

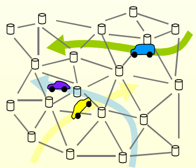
Objective:

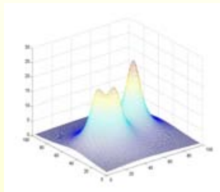
To determine the number of targets and their approximate locations in the field, forming an initial count and re-computing the count when targets move, enter, or leave the field.

Acoustic Signal Field Landscape

Goal: count and track the significant signal peaks in the field

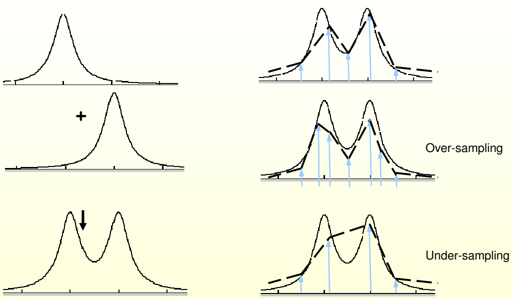
1. Signal attenuation rate, and the spacing and communication range of sensors have big impact on "signal resolution"
2. Number of detected peaks may not equal number of targets due to sampling artifacts and/or noise.





Combinatorial Signal Processing and Computational Topology

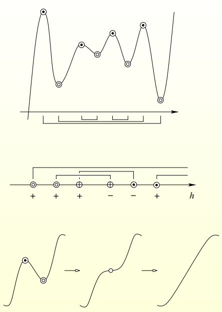
Scattered Amplitude Sampling



In 1-D, the number of peaks in sampled landscape cannot exceed the number of peaks in the true landscape. However, this is possible in 2-D.

Peak Landscape Simplification

- Some signal peaks may be noise
- Usually such peaks are near other critical points of the landscape
- Topological ideas, such as persistent homology can be used to simplify the landscape by canceling saddles with maxima, etc., removing noise

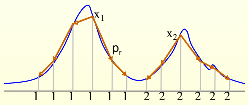


Downhill Flooding Protocol (DFP)

A sensor node is qualified as a **leader**, if its reading is higher than that of all its one hop neighbors.

Local *leader election* is conducted by sensors exchanging information with their neighbors via one hop broadcast.

p_r = received signal power at each node



x_1 and x_2 are elected leaders,

other nodes join one of the groups formed by the leaders: each node joins the highest leader it can reach by a monotone ascending path

1. Only sensors with myPr > threshhold will participate
2. Each sensor emits a packet in each protocol period, broadcasting its reading Pr and its ID
3. Sensors pass on or drop a packet P from their neighbors, according to the following rule:

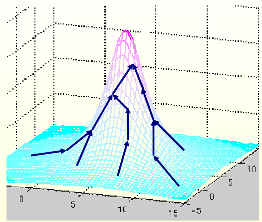
   If (Pr recorded in the packet > myPr & Pr of the sensor relaying this packet > myPr)

   broadcast(P);

   else

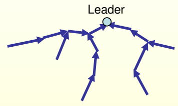
   drop(P);

Sensor Cluster Trees



For each node,

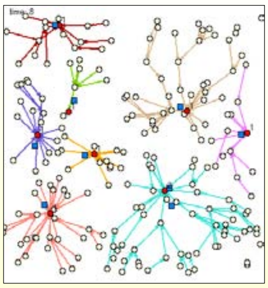
parent = the neighboring node with maximum Pr (received signal power)



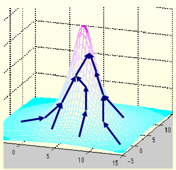
Bird's eye view of the cluster tree structure. Each node follows a strictly upwards path to the highest peak it can reach.

We call such groups of sensors aggregates, as they collaboratively perform a task.

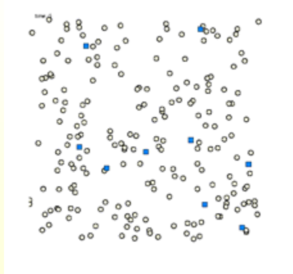
2-D View of a Sensor Field with Cluster Trees Formed Using DFP

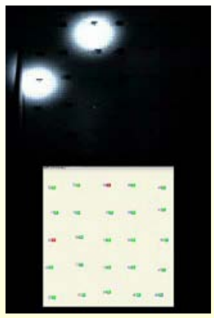


- Different colors mark different sensor clusters formed
- Each cluster has one leader



Target Counting Demo





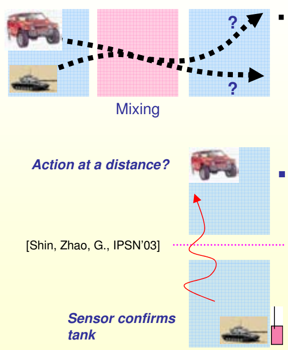
Simulation with 9 moving targets (above); Implementation on motes sensors (right)

jliebman

Some Lessons and Issues

- Sensors naturally form *collaboration groups*. Target localization and counting can be performed in-network.
- These sensor collaboration groups must be maintained as the physical phenomena of interest change over time.
- Aggregates may be easier to sense than individual objects
- Equivalently, physical phenomena are translated into networking behaviors.
- Can such behaviors be programmed without naming the nodes individually?

V3. Distributed Identity Management

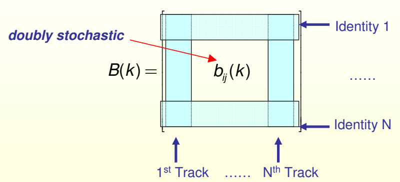


- Multi-target Tracking (MTT)
  - Basic application of sensor networks
  - Data association problem → *Target mixing*
- Multi-target Identity Management
  - Represent and manage additional quantity called target identity
  - Simplified version
    - Position estimates are given
    - Fixed number of targets assumed

Approach A: Belief Matrix

The belief matrix $B(k)$ is the main quantity that the algorithm maintains

$\rightarrow$ Which ID goes with which track, with what probability?



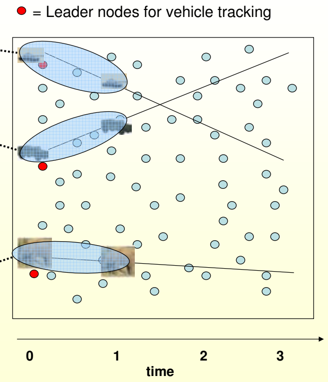
where $b_{ij}(k) = p(\mathrm{ID}(x_j(k)) = i)$ and $\vec{b}_i(k)$ is ID belief vector on i_{th} track.

Q: How to update $B(k)$, given $X(k+1)$?

Distributed Management of B(k), I

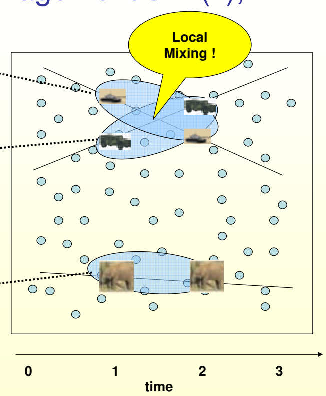
$$B(0) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$b_1(0) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad b_2(0) = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad b_3(0) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$



Distributed Management of B(k), II

$$b_1(1) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad b_2(1) = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad b_3(1) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

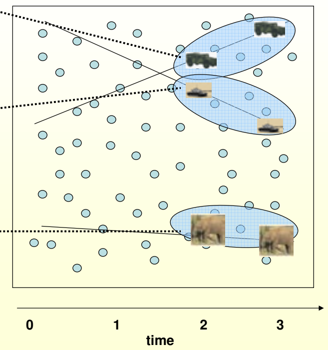


Distributed Management of B(k), III

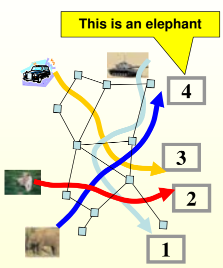
$$b_1(2) = \alpha b_1(1) + (1-\alpha) b_2(1) = \begin{bmatrix} \alpha \\ 1-\alpha \\ 0 \end{bmatrix}$$

$$b_2(2) = (1-\alpha) b_1(1) + \alpha b_2(1) = \begin{bmatrix} 1-\alpha \\ \alpha \\ 0 \end{bmatrix}$$

$$b_3(2) = b_3(1) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$



Incorporating Local Evidence



$$B = \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array}$$
$$B = \begin{bmatrix} 0.1737 & 0.0947 & 0.3447 & 0.3869 \\ 0.3527 & 0.6473 & 0 & 0 \\ 0.3201 & 0.1744 & 0.3043 & 0.2011 \\ 0.1535 & 0.0836 & 0.3509 & \boxed{0.4120} \end{bmatrix}$$

$\downarrow$

$$B = \begin{bmatrix} 0.1737 & 0.0947 & 0.3447 & 0 \\ 0.3527 & 0.6473 & 0 & 0 \\ 0.3201 & 0.1744 & 0.3043 & 0 \\ 0.1535 & 0.0836 & 0.3509 & 1 \end{bmatrix}$$

Not doubly-stochastic → inconsistent!

Renormalization, Given Local Evidence

1. Ideal solution: Bayesian normalization,
   given the priors, all the mixing events in history
   and all the sensor evidence.
   Exact solution in this framework. (Bayesian posterior)
   Used as a reference - Desirable properties of the solution.

2. Realistic solution: Sinkhorn Iteration (repeatedly normalize rows and columns)

[Sinkhorn 1964,1967; Sinkhorn and Knopp 1967]

What Do We Want?

- The belief matrix represents a probability distribution. The matrix A represents our *a priori* belief, but violates sum constraints.
- We would like to find the sum-constrained (feasible) matrix B that is the *closest distribution* to A (which is infeasible).
- Use Kullback-Leibler distance (measure of distance between distributions):

$$KL(B:A) = \sum_{j=1}^{n} \sum_{i=1}^{m} b_{ij} \log \frac{b_{ij}}{a_{ij}}$$

Sinkhorn Scaling and the Kullback-Leibler Distance

$$KL(B:A) = \sum_{j=1}^{n} \sum_{i=1}^{m} b_{ij} \log \frac{b_{ij}}{a_{ij}}$$

Theorem: Given a prior matrix $A \in \mathbb{R}^{m \times n}$, the matrix B that satisfies the row and column sum constraints, and minimizes the KL-distance from the prior matrix A is **always** the solution of the Sinkhorn scaling process.

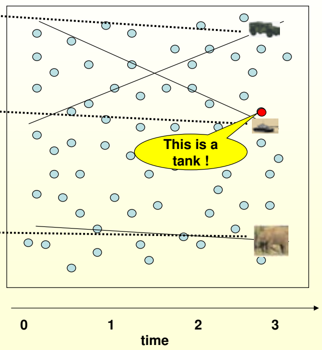
[Balakrishnan, Hwang,Tomlin '04]

Solve by interior point methods:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{m} \sum_{j=1}^{n} b_{ij} \log \frac{b_{ij}}{a_{ij}} \\ \text{subject to} & \sum_{j=1}^{n} b_{ij} = r_i \; \forall i = 1 \cdots m \\ & \sum_{i=1}^{m} b_{ij} = c_j \; \forall j = 1 \cdots n \\ & b_{ij} \geq 0 \; \forall i = 1 \cdots m; \; j = 1 \cdots n \end{array}$$

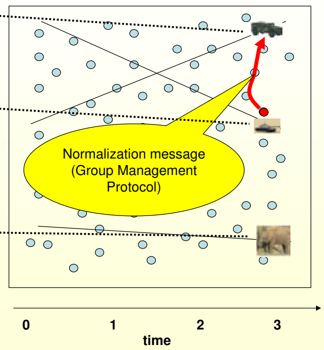
Distributed Management of B(k), IV

$$b_1(3) = \begin{bmatrix} \alpha \\ 1-\alpha \\ 0 \end{bmatrix} \quad b_2(3) = \begin{bmatrix} 1-\alpha \\ \alpha \\ 0 \end{bmatrix} \quad b_3(3) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$



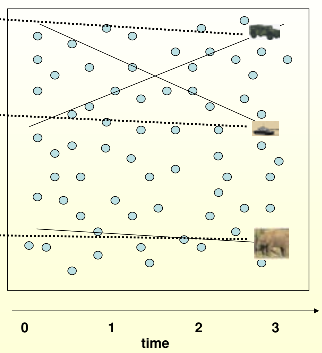
Distributed Management of B(k), V

$$b_1(3) = \begin{bmatrix} \alpha \\ 1-\alpha \\ 0 \end{bmatrix} \quad b_2(3) = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad b_3(3) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

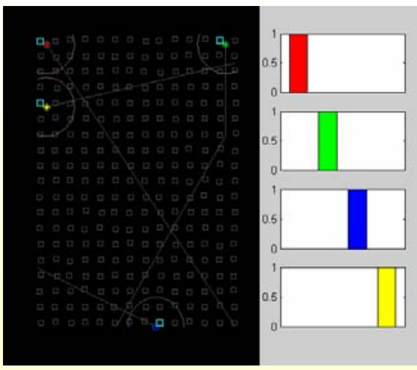


Distributed Management of B(k), VI

$$b_1(3) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad b_2(3) = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad b_3(3) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$



Target Mixing Video



Particles filters are used to keep track about multiple hypotheses about the location of each vehicle

Distributed Implementation of Tracking and Normalization

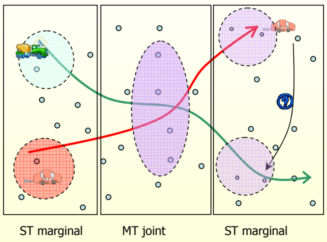
Q: Which nodes store and/or compute what information?

-- When do we track in joint space?

-- How to distribute B(k)?

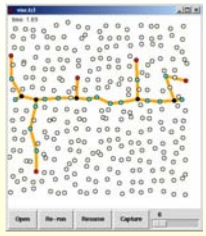
-- How to implement the probabilistic normalization?

-- What is required at the communication/network layer to make the above happen?



Managing Sensor Groups

- We have distributed columns of B(k) to leader nodes tracking targets.
- When a leader initiates a normalization based on local evidence, it has to know where are the other leaders that have non-zero mass on the evidence ID.
- Group Management Protocol: Maintains the group membership based on the ID probability mass.
- Communication needs to be minimized.



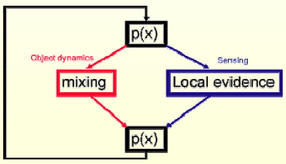
RoamHBA protocol
[Fang, Liu, G., Zhao, IPSN'04]

Approach B: Information Matrix

- Keep unnormalized beliefs by simply adding log-likelihoods after each local evidence event
- No communication necessary, except when mixing events occur or queries are made

$$L = \begin{bmatrix} l_{11} & \cdots & l_{1j} & \cdots & l_{1N} \\ l_{21} & \cdots & l_{2j} & \cdots & l_{2N} \\ & & \vdots & & \\ l_{N1} & \cdots & l_{Nj} & \cdots & l_{NN} \end{bmatrix}$$

$$l_{ij} = \sum_t \log(p(Z^t = (k,j)|x_j = i), k \in \{1, \cdots, N\}$$



[Shin, Lee, Thrun, G. '05 and Schumitch, Thrun, Bradski, Olukotun '05]

Some Lessons and Issues

- Collaboration groups need not always be physically clustered.
- Different attributes of a phenomenon can be tracked at different rates (target location, identity).
- A change of information representation can have a deep impact on cost trade-offs.
- How do information providers and information seekers locate each other?

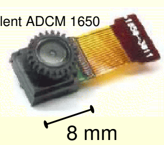
V4. Image Sensor Networks

- CMOS technology enables the production of small, low-cost and low-power integrated image sensors
- Cameras (still or video) and other image sensors are becoming cheaper, smaller, and nearly ubiquitous
- However, truly distributed networked systems of image sensors are still not here

Wireless Camera Node

- CMOS image sensor
  - Small, cheap, battery-powered
  - Integrated CPU and radio

Agilent ADCM 1650



Video Sensing
OV6650 CMOS
352 x 288
30 fps
20 mW

Computation
TI MSP430
10 KB
8 MHz
3 mW

Communication
RFM TR1001
300 m
20 Kbps
20 mW

Current Multi-Imager Networks

- Data is transported over a wired network to a central location
- Human operators look at the data

- This approach cannot scale:
  - vast amounts of data to move
  - wiring is expensive
  - automatic ways to filter the data are needed

Distributed Imager Challenges

- Imagers are high data rate sensors; therefore data must be compressed and summarized
  - compression must take into account shared data
  - goal of compression need not be reconstruction
- Vision algorithms can be expensive to run on weak capability, low power devices
- Visibility is non-local and discontinuous (occlusions, etc)
- Issues of privacy, etc.

Collaborative, Task-Driven Image Sensing

- Large numbers of simple, inexpensive cameras collaborate over a wireless network to accomplish a task
- Data is compressed locally and aggregated within the network
- Cameras are only tasked as the situation demands
- The system can be expanded incrementally to large numbers of nodes

The goal is to estimate certain high-level, global attributes of the environment.

The Initial Effort

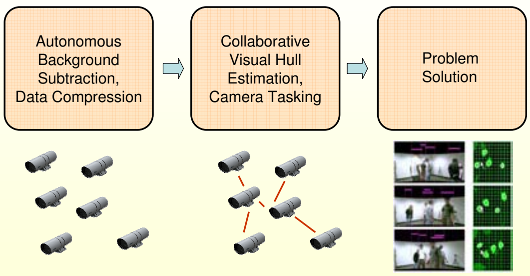
- Use a camera network to obtain information about space occupancy by people.
- Useful for aggregate tracking, counting, etc.
- Crowd density implies multiple occlusions – no one camera by itself can do this.
- No image reconstruction --just high-level distributed spatial reasoning.

Packard 013

The Experimental Setup

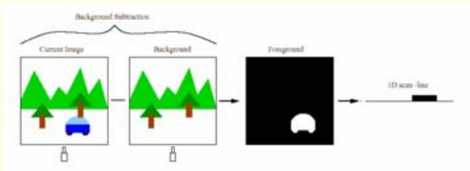
- Web cameras:
  - 16 firewire webcams with 49 degree FOV
  - Placed around a 22 x 19 foot room
- Linux compputers
  - A PC is connected to 2 webcams
  - A separate process is running for each webcam to simulate an individual camera node
  - All processes can communicate with each other over the network

System Architecture

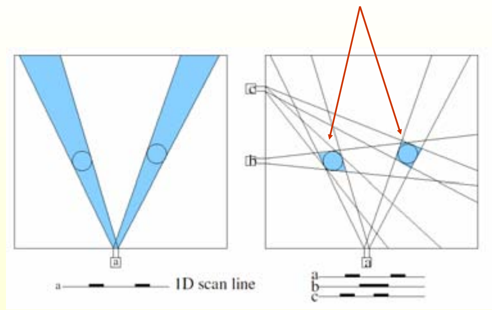


Local Processing

- Perform background subtraction
- Collapse to a single scan-line
- 640x480 RGB image → 640 bit scan-line (which can be further compressed)

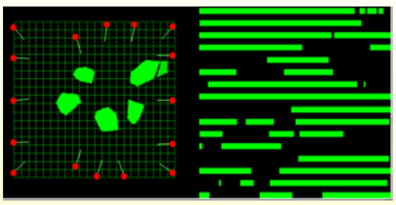


Occupancy Representation: The 2-D Visual Hull



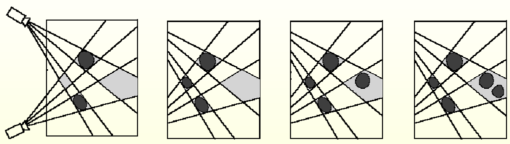
A Visual Hull Example

Top view of room with 5 people



Scanlines from 16 cameras

The Visual Hull Overestimates Occupancy

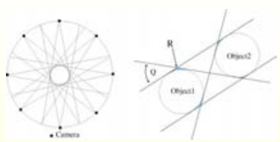


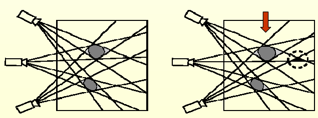
Visual hull regions surround each object.

Visual hull regions may also be empty; we call these the phantom regions

Pruning the Visual Hull

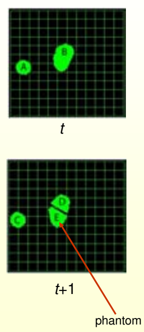
- Using more cameras reduces the overestimation – but it can never be fully eliminated
- Motion can allow the pruning of phantom regions

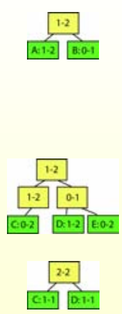




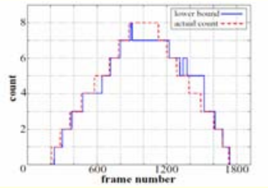
An Application: Counting People

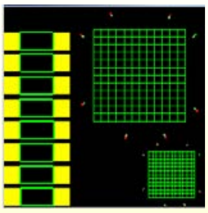
- Given occupancy, bound the number of objects in each polygon of the visual hull
- The bounds over time can be used to constrain the count, using a tree data structure.





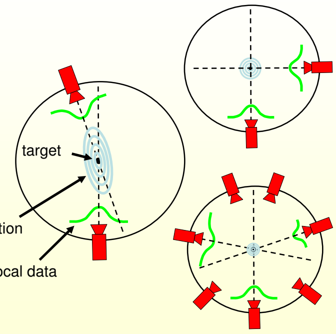
A Counting Example





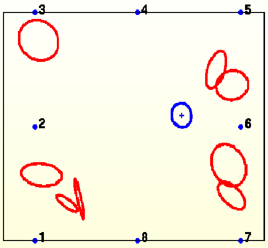
Localization – Where?

- Global: **Target localization**
- Best camera placement
- Best camera tasking
- **Contributions**
  - Analysis of target localization error
  - Solutions for camera placement and tasking

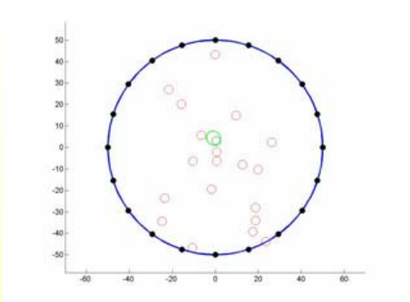


Localization w. Occlusions

- Suppose *M* moving objects in a room
- Given priors on these objects
- And camera positions
- We are interested in localizing one of them (blue prior)
- Which *k* cameras are best for localization?



Algorithm Animation



circular

Scaling to Large Camera Networks

- Few foci of activity
- Episodic events of interest
- Camera control (aiming, panning, zooming)
- Camera selection (how many, which ones)

•Analysis
•Simulation
•Real Experiments

Some Lessons and Issues

- A surprising amount of spatial information can be captured by cameras sharing very few bits.
- Small subsets of cameras, when appropriately tasked, can provide accurate estimates.
- The system can perform counting without tracking, thus raising no privacy issues.
- Alternatively, low-res video cameras for occupancy can be combined with a few high-res still cameras that can be commanded to snap a few high-detail photos, capturing the essentials of a scene.

Five Quick Vignettes on Lightweight Spatiotemporal Reasoning

1. Tracking Wide-Area Phenomena
2. Counting Moving Objects
3. Distributed Identity Management
4. Tracking Spatial Occupancy

Sensor selection

Sensor collaboration groups

Network structure discovery

Lightweight information integration

Uncertainty and multiple hypotheses

Conclusions

- Ubiquitous networked sensors provide a dense spatial and temporal sampling of the physical world
- They allow low-latency access to information that is highly localized in time and space, and thus provide a way to sense and act on the physical world beyond what has been possible up to now
- Sensor networks raise many research issues at the physical node level, the system architecture level, and the algorithm deployment level
- A combination of global and local methods promise to provide robust tools for network structure discovery, understanding the global aspects of sensor layouts and signal landscapes

Acknowledgements

Stanford Students: Qing Fang, Jie Gao, Ali Ozer Ercan, Jaewon Shin, Feng Xie, Danny Yang

Stanford Postdocs: Vin de Silva

External Collaborators: Patrick Cheung, Julia Liu (PARC), Jie Liu, Feng Zhao (Microsoft), Hector Gonzales-Banos (Honda), Li Zhang (HP Labs), John Hershberger (Mentor Graphics)

Funding Sponsors: NSF, DARPA, ONR, Stanford NRC, Honda, Xerox

The End

Another kind of sensor network exhibiting lightweight spatiotemporal reasoning

Games in Networks:
Routing, Network Design, Potential Games, and Equilibria and Inefficiency

Éva Tardos
Cornell University

1

Part I

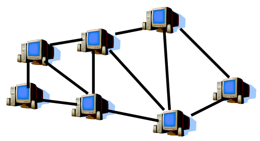
- what is a game?
- Pure and randomized equilibria
- Load balancing and routing as games

2

Why care about Games?

Users with a multitude of diverse economic interests sharing a Network (Internet)
- browsers
- routers
- servers

Selfishness:
Parties deviate from their protocol if it is in their interest



Model Resulting Issues as

Games on Networks

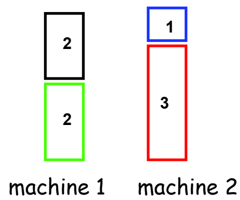
3

A simple game: load balancing

Each job wants to be on a lightly loaded machine.

With coordination we can arrange them to minimize load

Example: load of 4

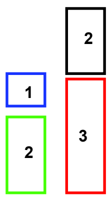


4

A simple game: load balancing

Each job wants to be on a lightly loaded machine.

- Without coordination?
- Stable arrangement:
No job has incentive to switch
- Example: some have load of 5



5

Games: setup

- A set of players (in example: jobs)
- for each player, a set of strategies
(which machine to choose)

Game: each player picks a strategy

For each strategy profile (a strategy for each player) $\rightarrow$ a payoff to each player
(load on selected machine)

Nash Equilibrium: stable strategy profile: where no player can improve payoff by changing strategy

6

Games: setup

Deterministic (pure) or randomized (mixed) strategies?

Pure: each player selects a strategy.
simple, natural, but stable solution may not exists

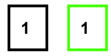
Mixed: each player chooses a probability distribution of strategies.

- equilibrium exists (Nash),
- but pure strategies often make more sense

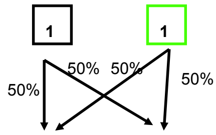
7

Pure versus Mixed strategies in load balancing

- Pure strategy: load of 1



- A mixed equilibrium
Expected load of 3/2 for both jobs



8

Quality of Outcome: Goal's of the Game

Personal objective for player i:
min load L_i or expected load $E(L_i)$

Overall objective?

- **Social Welfare:** $\Sigma_i L_i$ or
expected value $E(\Sigma_i L_i)$
- Makespan: $\max_i L_i$ or
max expected value $\max_i E(L_i)$ or
expected makespan $E(\max_i L_i)$

9

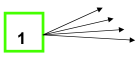
Example: simple load balancing

n identical jobs and n machines

1 1 1 1 1

All pure equilibria: load of 1 (also optimum)

A mixed equilibrium: prob 1/n each machine



expected load: $E(L_i) = 1+(n-1)\frac{1}{n} < 2$ for each i

$E(\max_i L_i)$: balls and bins: log n/log log n

10

Results on load balancing:

Theorem for $E(\max_i L_i)$:

- w/uniform speeds, p.o.a $\leq$ log m/log log m
- w/general speeds, worst-case p.o.a. is
Θ(log m/log log log m)

11

Results on load balancing:

Theorem for $E(\max_i L_i)$:

- w/uniform speeds, p.o.a $\leq$ log m/log log m
- w/general speeds, worst-case p.o.a. is
Θ(log m/log log log m)

Proof idea: balls and bins is worst case??

Requence of results by
[Koutsoupias/Papadimitriou 99],
[Mavronicolas/Spirakis 01],
[Koutsoupias/Mavronicolas/Spirakis 02],
[Czumaj/Vöcking 02]

12

Today:

focus on pure equilibria

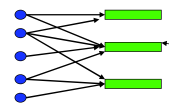
Does a pure equilibria exists?
Does a high quality equilibria exists?
Are all equilibria high quality?

some of the results extend to
sum/max of $E(L_i)$

13

load balancing and routing

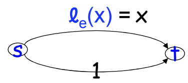
Load balancing:



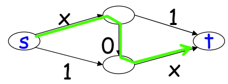
Delay as a function of load:

x unit of load → causes delay $\ell_e(x)$

Routing network:



Allow more complex networks

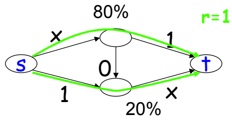


14

Atomic vs. Non-atomic Game

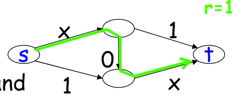
Non-atomic game:

- Users control an infinitesimally small amount of flow
- equilibrium: all flow path carrying flow are minimum total delay



Atomic Game:

- Each user controls a unit of flow, and
- selects a single path or machine

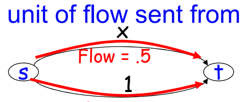


Both congestion games: cost on edge e depends on the congestion (number of users)

15

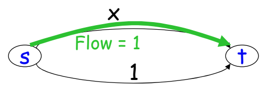
Example of nonatomic flow on two links

- One unit of flow sent from s to t



Traffic on lower edge is envious.

An envy free solution:



No-one is better off

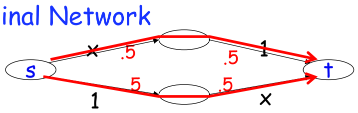
Infinite number of players

- will make analysis cleaner by continuous math

16

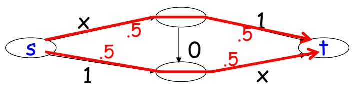
Braess's Paradox

Original Network



Cost of Nash flow = 1.5

Added edge:

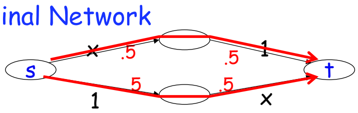


Effect?

17

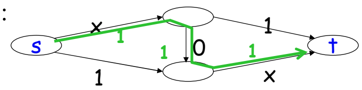
Braess's Paradox

Original Network



Cost of Nash flow = 1.5

Added edge:



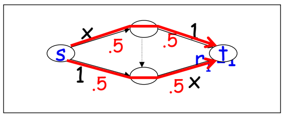
Cost of Nash flow = 2

All the flow has increased delay!

18

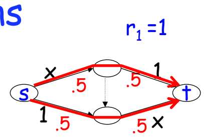
Model of Routing Game

- A directed graph $G = (V,E)$
- source-sink pairs s_i, t_i for $i=1,..,k$
- rate $r_i \geq 0$ of traffic between s_i and t_i for each $i=1,..,k$



- Load-balancing jobs wanted min load
- Here want minimum delay:
  - delay adds along path
  - edge-delay is a function $\ell_e(\cdot)$ of the load on the edge e

Delay Functions

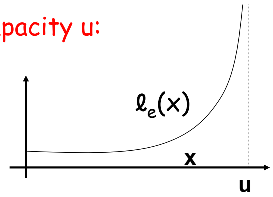


Assume $\ell_e(x)$ continuous and monotone increasing in load x on edge

No capacity of edges for now

Example to model capacity u:

$\ell_e(x) = a/(u-x)$



Goal's of the Game

Personal objective: minimize

$\ell_P(\mathbf{f})$ = sum of latencies of edges along **P** (wrt. flow **f**)

No need for mixed strategies

Overall objective:

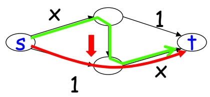
C(f) = total latency of a flow **f**: $= \Sigma_P\, f_P \cdot \ell_P(f)$

=social welfare

Routing Game??

Flow represents

- cars on highways
- packets on the Internet



individual packets or small → continuous model

User goal: Find a path selfishly minimizing user delay

≈ true for cars,
packets?: users do not choose paths on the Internet: routers do!

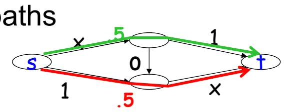
With delay as primary metric → router protocols choose shortest path!

Connecting Nash and Opt

- Min-latency flow
  - for one s-t pair for simplicity
- minimize $C(f) = \Sigma_e f_e \cdot \ell_e(f_e)$
- subject to: f is an s-t flow
- carrying r units
- By summing over edges rather than paths where f_e = amount of flow on edge e

Characterizing the Optimal Flow

- Optimality condition: all flow travels along minimum-gradient paths

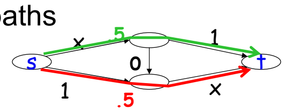


gradient is:
$(x\, \ell(x))'$
$= \ell(x) + x\, \ell'(x)$

Characterizing the Optimal Flow

- Optimality condition: all flow travels along minimum-gradient paths

gradient is:
$(x\,\ell(x))'$
$= \ell(x)+x\,\ell'(x)$



Recall: flow f is at Nash equilibrium iff all flow travels along minimum-latency paths

25

Nash $\leftrightarrow$ Min-Cost

Corolary 1: min cost is "Nash" with delay
$\ell(x)+x\,\ell'(x)$

Corollary 2: Nash is "min cost" with cost
$\Phi(f) = \Sigma_e \int_0^{f_e} \ell_e(x)\,dx$

Why?
gradient of:
$(\int_0^{f_e} \ell_e(x)\,dx)' = \ell(x)$

26

Using function Φ

- Nash is the solution minimizing Φ

Theorem (Beckmann'56)

- In a network latency functions $\ell_e(x)$ that are monotone increasing and continuous,
- a deterministic Nash equilibrium exists, and is essentially unique

27

Using function Φ (con't)

- Nash is the solution minimizing value of Φ
- Hence,

$\Phi(\text{Nash}) < \Phi(\text{OPT})$.

Suppose that we also know for any solution
$\Phi \le \text{cost} \le A\ \Phi$

$\rightarrow$ cost(Nash) $\le A\ \Phi(\text{Nash}) \le A\ \Phi(\text{OPT}) \le A$ cost(OPT).
$\rightarrow$ There exists a good Nash!

28

Example: $\Phi \le \text{cost} \le A\ \Phi$

Example: $\ell_e(x) = x$ then
- total delay is $x \cdot \ell_e(x) = x^2$
- potential is $\int \ell_e(\xi)\,d\xi = x^2/2$

More generally: linear delay $\ell_e(x) = a_e x + b_e$
- delay on edge $x \cdot \ell_e(x) = a_e x^2 + b_e x$
- potential on edge: $\int \ell_e(\xi)\,d\xi = a_e x^2/2 + b_e x$
- ratio at most 2

Degree d polynomials:
- ratio at most d+1

29

Sharper results for non-atomic games

Theorem 1 (Roughgarden-Tardos'00)
- In a network with linear latency functions
  - i.e., of the form $\boldsymbol{\ell_e(x)=a_e x+b_e}$
- the cost of a Nash flow is at most 4/3 times that of the minimum-latency flow

30

Sharper results for non-atomic games

Theorem 1 (Roughgarden-Tardos'00)

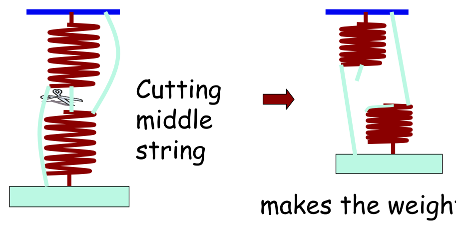
- In a network with linear latency functions
  - i.e., of the form $\ell_e(x)=a_e x+b_e$
- the cost of a Nash flow is at most 4/3 times that of the minimum-latency flow

Nash cost 1 optimum 3/4

Nash cost 2 optimum 1.5

31

Braess paradox in springs (aside)



makes the weight rise

and decreases power flow along springs

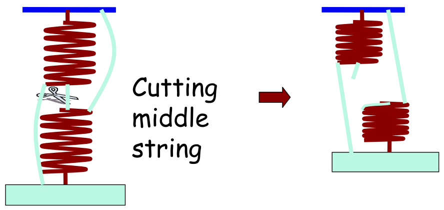
Flow=power; delay=distance

32

Bounds for spring paradox

Theorem 1' (Roughgarden-Tardos'00)

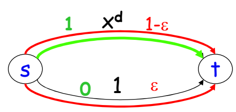
In a network with springs and strings cutting some strings can increase the height by at most a factor of 4/3.



33

General Latency Functions

- **Question:** what about more general edge latency functions?
- **Bad Example:** ($r = 1$, d large)



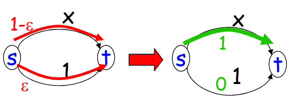
A Nash flow can cost arbitrarily more than the optimal (min-cost) flow

34

Sharper results for non-atomic games

Theorem 2 (Roughgarden'02):

- In any network with any class of convex continuous latency functions
- the worst price of anarchy is always on two edge network

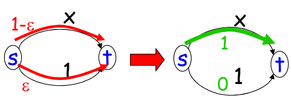


35

Sharper results for non-atomic games

Theorem 2 (Roughgarden'02):

- In any network with any class of convex continuous latency functions
- the worst price of anarchy is always on two edge network

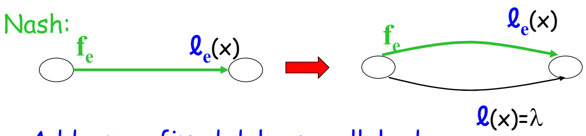


Corollary: price of anarchy for degree d polynomials is O(d/log d).

36

Another Proof idea

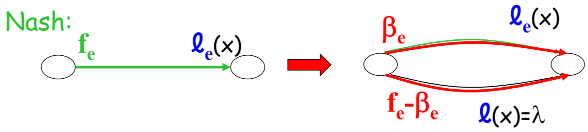
Modify the network



- Add a new fixed delay parallel edge
  - fixed cost set $\lambda = \ell_e(f_e)$
- Nash not effected
- Optimum can only improve

37

Modified Network



- fixed cost set $\lambda = \ell_e(f_e)$

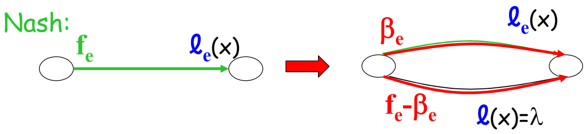
- Optimum on modified network

splits flow so that marginal costs are equalized

- and common marginal cost is $\lambda = \ell_e(f_e)$

38

Proof of better bound



- Theorem 2: the worst price of anarchy is always two edge network

- Proof: Prize of anarchy on G is median of ratios for the edges

39

More results for non-atomic games

Theorem 3 (Roughgarden-Tardos'00):

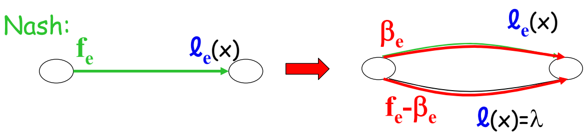
- In any network with continuous, nondecreasing latency functions

cost of Nash with rates r_i for all i	$\leq$	cost of opt with rates $2r_i$ for all i

Proof ...

40

Proof of bicriteria bound



common marginal cost on two edges in opt is $\lambda = \ell_e(f_e)$

- Proof: Opt may cost very little, but marginal cost is as high as latency in Nash
- $\rightarrow$ Augmenting to double rate costs at least as much as Nash

41

More results for non-atomic games

Theorem 3 (Roughgarden-Tardos'00):

- In any network with continuous, nondecreasing latency functions

cost of Nash with rates r_i for all i	$\leq$	cost of opt with rates $2r_i$ for all i

Morale for the Internet: build for double flow rate

42

Morale for IP versus ATM?

Corollary: with M/M/1 delay fns: $\ell(x)=1/(u-x)$, where u=capacity

Nash w/cap. $2u \le$ opt w/cap. u

Doubling capacity is more effective than optimized routing (IP versus ATM)

43

Part II

- Discrete potential games:
- network design
- price of anarchy stability

44

Continuous Potential Games

Continuous potential game: there is a function $\Phi(f)$ so that Nash equilibria are exactly the local minima of Φ

also known as Walrasian equilibrium Φ convex then Nash equilibrium are the minima. For example

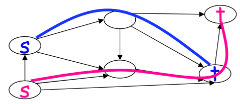
$$\Phi(f) = \Sigma_e \int_0^{f_e} \ell_e(x)\, dx$$

45

Discrete Analog

Atomic Game

- Each user controls one unit of flow, and
- selects a single path



Theorem Change in potential is same as function change perceived by one user [Rosenthal'73, Monderer Shapley'96,]

$\Phi(f) = \Sigma_e\, (\ell_e(1)+\ldots+\ell_e(f_e)) = \Sigma_e\, \Phi_e$

Even though moving player ignores all other users

46

Potential: Tracking Happiness

Theorem Change in potential is same as function change perceived by one user [Rosenthal'73, Monderer Shapley'96,]

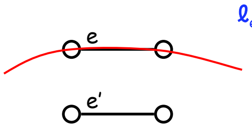
$\Phi(f) = \Sigma_e\, (\ell_e(1)+\ldots+\ell_e(f_e)) = \Sigma_e\, \Phi_e$

Reason?

Potential before move:

$\ell_e(1)+\ldots\ \ell_e(f_e-1) + \ell_e(f_e)$

$+ \quad \ell_{e'}(1)+\ldots+\ell_{e'}(f_{e'})$



47

Potential: Tracking Happiness

Theorem Change in potential is same as function change perceived by one user [Rosenthal'73, Monderer Shapley'96,]

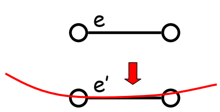
$\Phi(f) = \Sigma_e\, (\ell_e(1)+\ldots+\ell_e(f_e)) = \Sigma_e\, \Phi_e$

Reason?

Potential after move:

$\ell_e(1)+\ldots\ \ell_e(f_e-1) + \cancel{\ell_e(f_e)}$

$+ \quad \ell_{e'}(1)+\ldots+\ell_{e'}(f_{e'}) + \ell_{e'}(f_{e'}+1)$



Change in Φ is $-\ell_e(f_e) + \ell_{e'}(f_{e'}+1)$

same as change for player

48

What are Potential Games

Discrete potential game: there is a function $\Phi(f)$ so that change in potential is same as function change perceived by one user

Theorem [Monderer Shapley'96] Discrete potential games if and only if congestion game (cost of using an element depends on the number of users).

Proof of "if" direction $\Phi(f) = \Sigma_e (\ell_e(1)+\ldots+\ell_e(f_e))$

Corollary: Nash equilibria are local min. of $\Phi(f)$

49

Why care about

Best Nash/Opt ratio?

Papadimitriou-Koutsoupias '99

Nash = outcome of selfish behavior

$\Rightarrow$worst Nash/Opt ratio: Price of Anarchy

Non-atomic game: Nash is unique…

Atomic Nash not unique!

50

Best Nash is good quality…

$$\text{Price of Stability} = \frac{\text{cost of best selfish outcome}}{\text{“socially optimum” cost}}$$

$$\text{Price of Anarchy} = \frac{\text{cost of worst selfish outcome}}{\text{“socially optimum” cost}}$$

Potential argument $\Rightarrow$ Low price of stability

But do we care?

51

Atomic Game: Routing with Delay

Theorems 1&2 true for the Nash minimizing the potential function, assuming all players carry the same amount of flow

Worst case on 2 edge network

52

Atomic Game: Price of Anarchy?

Theorem: Can be bounded for some classes of delay functions

e.g., polynomials of degree at most d at most exponential in d.

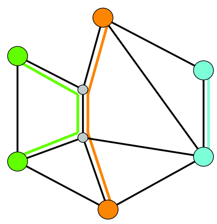
Suri-Toth-Zhou SPAA'04 + Awerbuch-Azar-Epstein STOC'05+ Christodoulou-Koutsoupias STOC'05

53

Network Design as Potential Game

Given: $G = (V,E)$,
costs $c_e(x)$ for all $e \in E$,
k terminal sets (colors)

Have a player for each color.



54

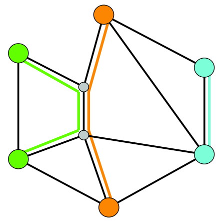
Network Design as Potential Game

Given: $G = (V,E)$,
costs $c_e(x)$ for all $e \in E$,
k terminal sets (colors)

Have a player for each color.

Each player wants to build a network in which his nodes are connected.

Player strategy: select a tree connecting his set.



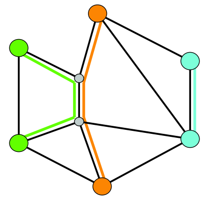
55

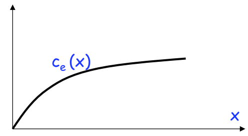
Costs in Connection Game

Players pay for their trees, want to minimize payments.

What is the cost of the edges?
$c_e(x)$ is cost of edge e for x users.

Assume economy of scale for costs:





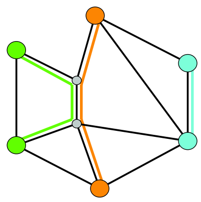
56

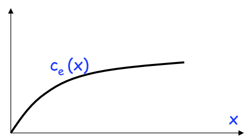
Costs in Connection Game

Players pay for their trees, want to minimize payments.

What is the cost of the edges?
$c_e(x)$ is cost of edge e for x users.

Assume economy of scale for costs:





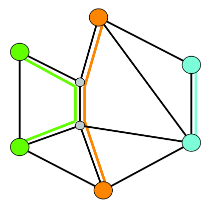
How do players share the cost of an edge?

57

A Connection Game

How do players share the cost of an edge?

Natural choice is fair sharing, or Shapley cost sharing:



58

A Connection Game

How do players share the cost of an edge?

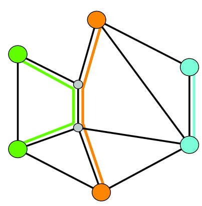
Natural choice is fair sharing, or Shapley cost sharing:

Players using e pay for it evenly:

$$c_i(P) = \sum c_e(k_e)/k_e$$

where k_e number of users on edge e

[Herzog, Shenker, Estrin'97]



59

A Connection Game

How do players share the cost of an edge?

Natural choice is fair sharing, or Shapley cost sharing:

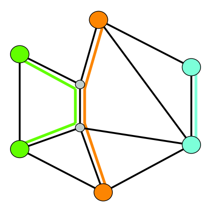
Players using e pay for it evenly:

$$c_i(P) = \sum c_e(k_e)/k_e$$

where k_e number of users on edge e

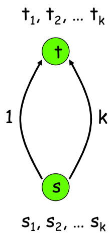
[Herzog, Shenker, Estrin'97]

This is congestion game: $\ell_e(x) = c_e(x)/x$
with decreasing "latency"



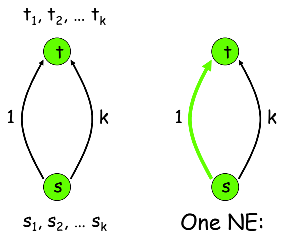
60

A Simple Example



61

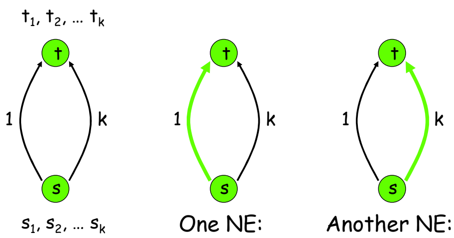
A Simple Example



One NE:
each player
pays 1/k

62

A Simple Example



One NE:
each player
pays 1/k

Another NE:
each player
pays 1

63

Maybe Best Nash is good?

We know price of anarchy is bad.

Game is a potential game so maybe Price of Stability is better.

$$\text{Price of Stability} = \frac{\text{cost of best selfish outcome}}{\text{“socially optimum” cost}}$$

Do we care?

64

Nash as Stable Design

Need to Find a Nash equilibrium

- Stable design: as no user finds it in their interest to deviate

Need to find a “good” Nash

- Best Nash/Opt ratio? = Price of Stability [ADKTWR 2004]

Design with a constraint for stability

65

Results for Network Design

Theorem [Anshelevich, Dasgupta, Kleinberg, Tardos, Wexler, Roughgarden FOCS'04]
Price of Stability is at most O(log k) for k players

proof:

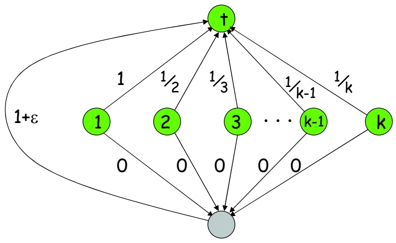
- edge cost c_e with $k_e > 0$ users
- edge potential with $k_e > 0$ users

$$\Phi_e = c_e \cdot (1+1/2+1/3+\ldots+1/k)$$

$\rightarrow$ Ratio at most $H_k = O(\log k)$

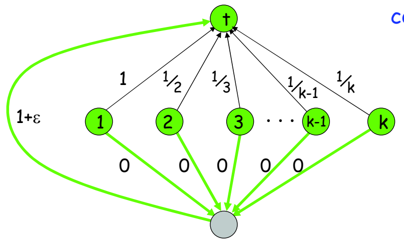
66

Example: Bound is Tight



67

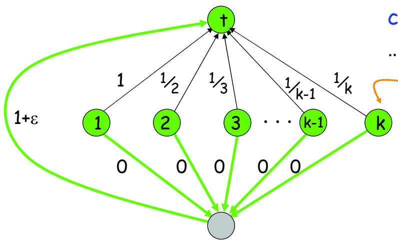
Example: Bound is Tight



cost(OPT) = 1+ ε

68

Example: Bound is Tight



cost(OPT) = 1+ ε

...but not a NE:

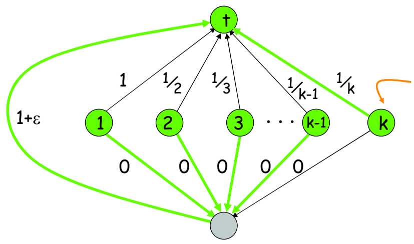
player k

pays (1+ ε)/k,

could pay 1/k

69

Example: Bound is Tight

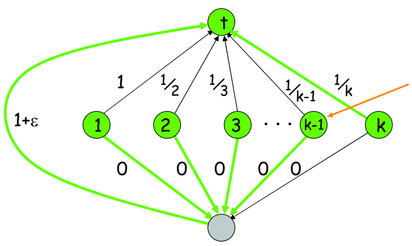


so player k

would deviate

70

Example: Bound is Tight



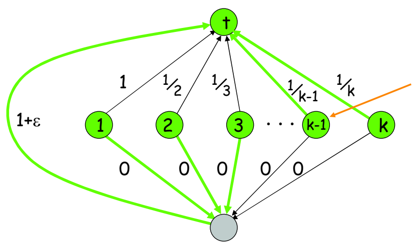
now player k-1

pays (1+ ε)/(k-1),

could pay 1/(k-1)

71

Example: Bound is Tight

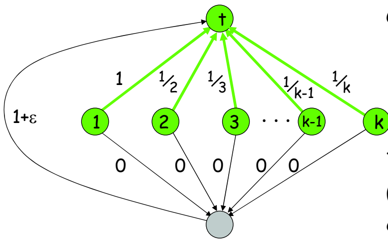


so player k-1

deviates too

72

Example: Bound is Tight



Continuing this process, all players defect.

This is a NE! (the only Nash) cost = $1 + \frac{1}{2} + \dots + \frac{1}{k}$

Price of Stability is $H_k = \Theta(\log k)$!

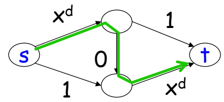
73

Congestion games

Routing with delay:

- cost increasing with congestion

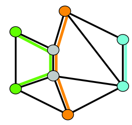
  e.g., $c_e(x) = x \cdot \ell_e(x) = x^{d+1}$



Network Design Game:

- cost decreasing with congestion

  e.g., $\ell_e(x) = c(x)_e / x$

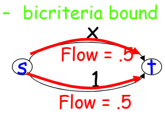


74

Contrast with Routing Games

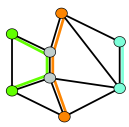
<u>Routing games</u>

- $c_e(x)$ increasing
- Traffic maybe non-atomic OK? to split traffic
- Nash is unique
- Price of Stability grows with steepness of c:
  - worst case on 2 links
  - bicriteria bound



<u>Design with Fair Sharing</u>

- $c_e(x)$ decreasing
- Choice atomic need to select single path
- Many equilibria
- Price of Stability bounded by ≅ log n



75

Part III

Is Nash a reasonable concept?

Is the price of anarchy always small? and what can be do when its too big (mechanism design)

Examples:

- Network design and
- Resource allocation

76

Why stable solutions?

Plan: analyze the quality of Nash equilibrium.

But will players find an equilibrium?

- Can a stable solution be found in poly. time?
- Does natural game play lead to an equilibrium?
- We are assuming non-cooperative players, what if there is cooperation?

77

Why stable solutions?

Plan: analyze the quality of Nash equilibrium.

But will players find an equilibrium?

- Can a stable solution be found in poly. time?
- Does natural game play lead to an equilibrium?
- We are assuming non-cooperative players, what if there is cooperation?

Answer 1: A clean solution concept and exists ([Nash 1952] if game finite)

Does life lead to clan solutions?

78

Why stable solutions?

- Finding an equilibrium?

Nonatomic games: we'll see that equilibrium can be found via convex optimization [Beckmann'56]

Atomic game: finding an equilibrium is polynomial local search (PLS) complete [Fabrikant, Papadimitriou, Talwar STOC'04]

79

Why stable solutions?

- Does natural game play lead to equilibrium?

we'll see that natural "best response play" leads to equilibrium if players change one at-a-time

See also:

Fischer¥Räcke¥Vöcking'06, Blum¥Even-Dar¥Ligett'06 also if players simultaneously play natural learning strategies

80

Why stable solutions?

- We are assuming non-cooperative players

Cooperation? No great models,
see some partial results on Thursday.

81

How to Design "Nice" Games?
(Mechanism Design)

Traditional Mechanism Design (VCG):

- use payments to induce all players to tell us his utility for connection
- Select a network to maximize social welfare (minimize cost)

82

How to Design "Nice" Games?
(Mechanism Design)

Traditional Mechanism Design (VCG):

- use payments to induce all players to tell us his utility for connection
- Select a network to maximize social welfare (minimize cost)

Cost lot of money; lots of information to share

83

How to Design "Nice" Games?
(Mechanism Design)

Traditional Mechanism Design (VCG):

- use payments to induce all players to tell us his utility for connection
- Select a network to maximize social welfare (minimize cost)

Cost lot of money; lots of information to share

Here:

- design a simple/natural Nash game where users select their own graphs and
- analyze the Prize of Anarchy

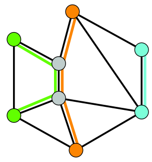
84

Network Design Mechanism

How should multiple players on a single edge split costs?

We used fair sharing …

[Herzog, Shenker, Estrin'97]



$c_i(P) = \sum c_e(k_e)/k_e$

where k_e number of users on edge e

which makes network design a potential game

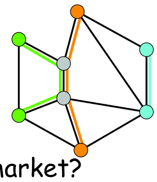
85

Network Design Game Revisited

How should multiple players on a single edge split costs?

We used fair sharing …

[Herzog, Shenker, Estrin'97]



Another approach: Why not free market?

players can also agree on shares? …any division of cost agreed upon by players is OK.

Near-Optimal Network Design with Selfish Agents STOC '03 Anshelevich, Dasgupta, Tardos, Wexler.

86

Network Design without Fairness

Results [Anshelevich, Dasgupta, Tardos, Wexler STOC'03]

Good news: Price of Stability 1 when all users want to connect to a common source

(as compared to log n for fair sharing)

But: with different source-sink pairs

- Nash may not exists (free riding problem)
- and may be VERY bad when it exists

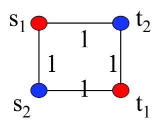
Partial good news: ∃ low cost Approximate Nash

87

No Deterministic Nash: Free Riding problem

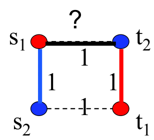
Network Design

[ADTW STOC'03]



Users bid contribution on individual edges.

- Single source game: Price of Anarchy = 1



- Multi source: no Nash

88

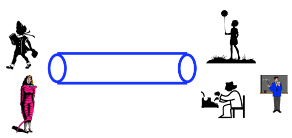
Mechanism Design

Example: Network design.

Results can be used to answer question: Should one promote "fair sharing" or "free market"?

89

Another Example: Bandwidth Allocation



Many Users with diverse utilities for bandwidth.

How should we share a given B bandwidth?

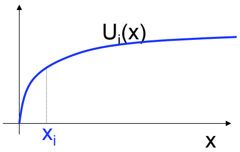
90

Bandwidth Sharing Game

Assumption:

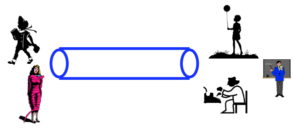
Users have a utility function $U_i(x)$ for receiving x bandwidth.

Assume elastic users (concave utility functions)



91

A Mechanism:



Many Users with diverse utilities for bandwidth.

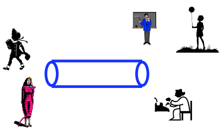
How should we share a given B bandwidth?

Kelly: proportional sharing

- Players offer money w_i for bandwidth.
- Bandwidth allocated proportional to payments:
  - effective price $p = (\Sigma_i\, w_i)/B$
  - player allocation $x_i = w_i\ /p$

92

A Mechanism:



Many Users with diverse utilities for bandwidth.

How should we share a given B bandwidth?

Kelly: proportional sharing

- Players offer money w_i for bandwidth.
- allocation proportional:
  - unit price $p = (\Sigma_i\, w_i)/B$
  - player i gets $x_i = w_i\ /p$

Thm: If players are price-takers (do not anticipate the effect of their bid on the price)

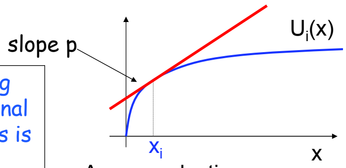
→ Selfish play results in optimal allocation

93

Price Taking Users

Given price p:

how much bandwidth does user i want?



Answer: keeps asking for more until marginal increase in happiness is at least p:

$U_i'(x)=p$

Assume elastic users (concave utility functions)

94

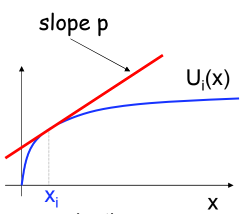
Price Taking Users: Kelly Mechanism Optimal

Equilibrium at price p:

each user i wants x_i such that $U_i'(x_i)=p$

Total bandwidth used up at price p

→ result optimal division of bandwidth

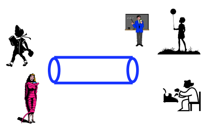


Assume elastic users (concave utility functions)

Price taking users standard assumption if many players

95

Kelly Proportional Sharing:



Players offer money w_i for bandwidth.

Bandwidth allocated proportional to payments

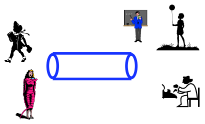
Johari-Tsitsiklis, 2004:

what if players do anticipate their effect on the price?

Theorem: Price of Anarchy at most $\frac{3}{4}$ on any networks, and any number of users

96

Kelly Proportional Sharing:



Players offer money w_i for bandwidth.

Bandwidth allocated proportional to payments

Theorem [Johari-Tsitsikis, 2004] Price of Anarchy at most $\frac{3}{4}$ on any networks, and any number of users

Why not optimal? big users "shade" their price. User choice

$$U_i'(x_i)(1-x_i)=p$$

assuming total bandwidth is 1

Worst case: one large user and many small users

97

Summary

We talked about many issues

Price of Anarchy/Stability/Coalitions in the context of some Network Games:

- routing, load balancing, network design, bandwidth sharing

- Designing games (mechanism design)
  - network design

98

Algorithmic Game Theory

- The main ingredients:
  - Lack of central control like distributed computing
  - Selfish participants game theory
- Common in many settings e.g., Internet

Most results so far:

- Price of anarchy/stability in many games, including many I did not mention
- e.g. Facility location (another potential game) [Vetta FOCS'02] and [Devanur-Garg-Khandekar-Pandit-Saberi'04]:

99

Some Open Directions:

- Other natural network games with low lost of anarchy
- Design games with low cost of anarchy
- Better understand dynamics of natural game play
- Dynamics of forming coalitions

100

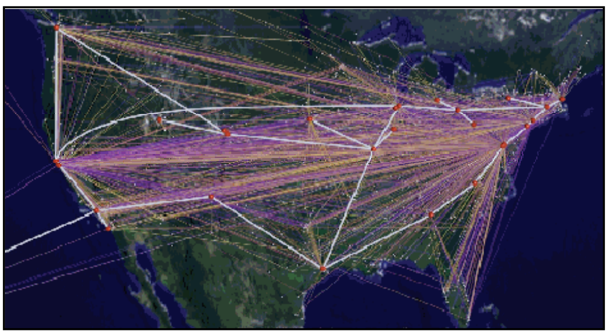
Polynomial Time Algorithms For Market Equilibria

Vijay V. Vazirani

Markets

Stock Markets

Internet



- Revolution in definition of markets

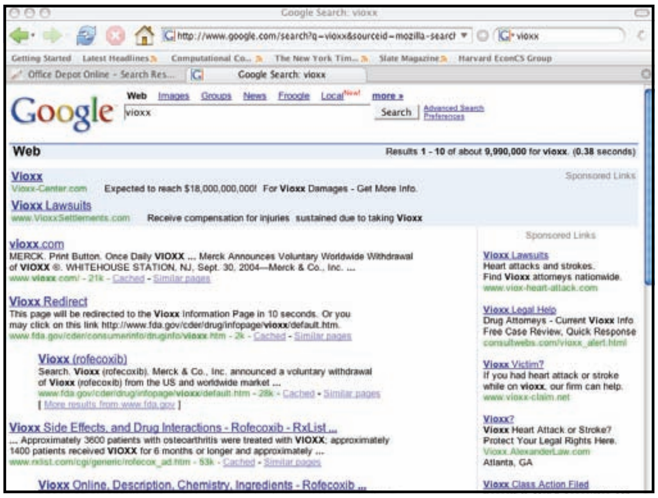
- Revolution in definition of markets
- New markets defined by
  - Google
  - Amazon
  - Yahoo!
  - Ebay

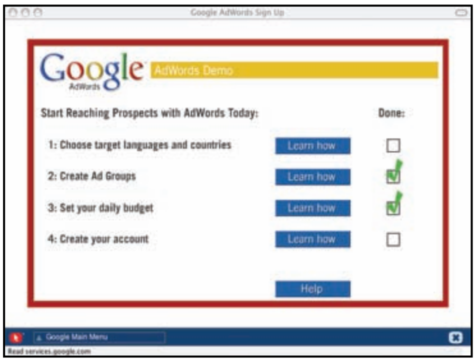
- Revolution in definition of markets
- Massive computational power available

- Revolution in definition of markets
- Massive computational power available
- Important to find good models and algorithms for these markets

Adwords Market

- Created by search engine companies
  - Google
  - Yahoo!
  - MSN
- Multi-billion dollar market
- Totally revolutionized advertising, especially by small companies.





How will this market evolve??

- The study of market equilibria has occupied center stage within Mathematical Economics for over a century.

- The study of market equilibria has occupied center stage within Mathematical Economics for over a century.

- This talk: Historical perspective
  & key notions from this theory.

2). Algorithmic Game Theory

- Combinatorial algorithms for
  traditional market models

3). New Market Models

- Resource Allocation Model of Kelly, 1997

3). New Market Models

- Resource Allocation Model of Kelly, 1997

- For mathematically modeling
  TCP congestion control

- Highly successful theory

A Capitalistic Economy

Depends crucially on
pricing mechanisms to ensure:

- Stability
- Efficiency
- Fairness

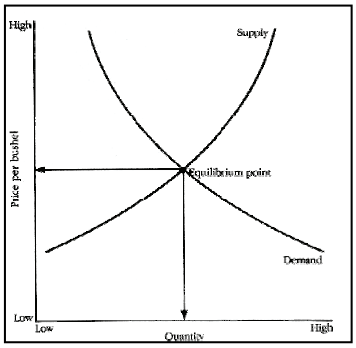
Adam Smith

- *The Wealth of Nations*
  2 volumes, 1776.

Adam Smith

- *The Wealth of Nations*
  2 volumes, 1776.
- 'invisible hand' of the market

Supply-demand curves



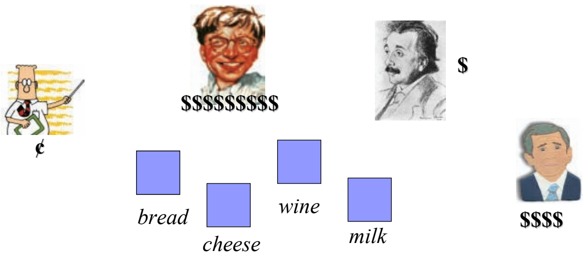
Leon Walras, 1874

- Pioneered general equilibrium theory

Irving Fisher, 1891

- First fundamental market model

Fisher's Model, 1891



- People want to maximize happiness – assume linear utilities.

Find prices s.t. market clears

Fisher's Model

- n buyers, with specified money, $m(i)$ for buyer i
- k goods (unit amount of each good)
- Linear utilities: u_{ij} is utility derived by i on obtaining one unit of j
- Total utility of i,

$$u_i = \sum_j u_{ij} x_{ij}$$

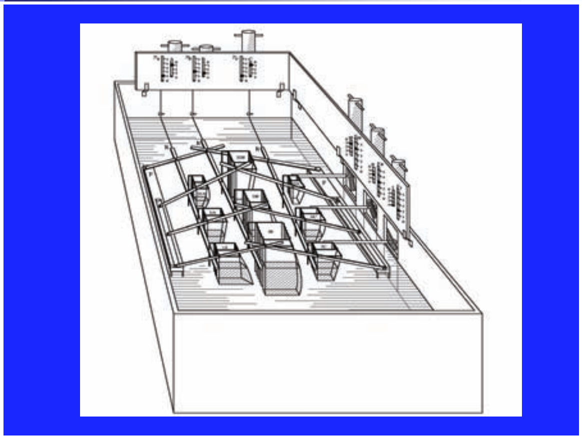
$$x_{ij} \in [0,1]$$

Fisher's Model

- n buyers, with specified money, $m(i)$
- k goods (each unit amount, w.l.o.g.)
- Linear utilities: u_{ij} is utility derived by i on obtaining one unit of j
- Total utility of i,

$$u_i = \sum_j u_{ij} x_{ij}$$

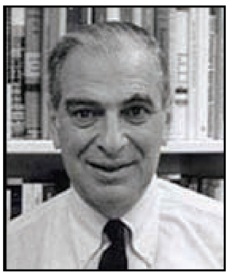
- Find prices s.t. market clears, i.e., all goods sold, all money spent.



Arrow-Debreu Model, 1954
Exchange Economy

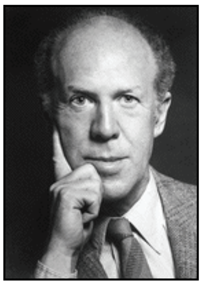
- Second fundamental market model
- Celebrated theorem in Mathematical Economics

Kenneth Arrow



- Nobel Prize, 1972

Gerard Debreu



- Nobel Prize, 1983

Arrow-Debreu Model

- n agents, k goods

Arrow-Debreu Model

- n agents, k goods
- Each agent has: initial endowment of goods,
  & a utility function

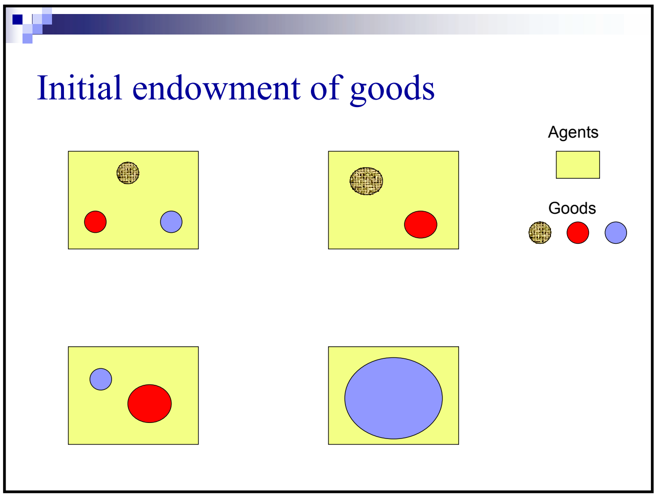
Arrow-Debreu Model

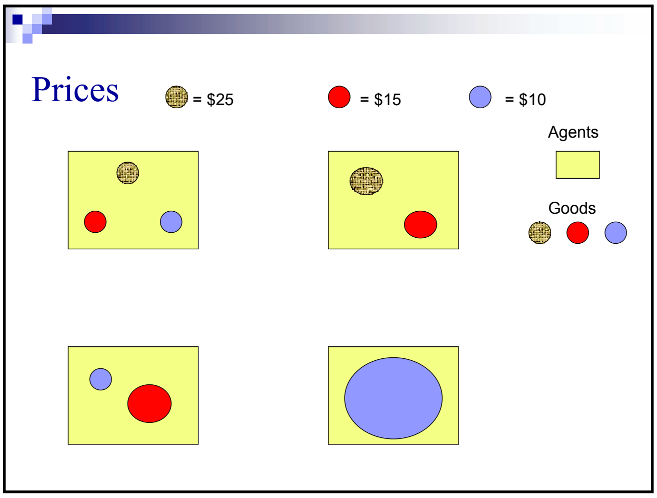
- n agents, k goods
- Each agent has: initial endowment of goods,
  & a utility function
- Find market clearing prices, i.e., prices s.t. if
  - Each agent sells all her goods
  - Buys optimal bundle using this money
  - No surplus or deficiency of any good

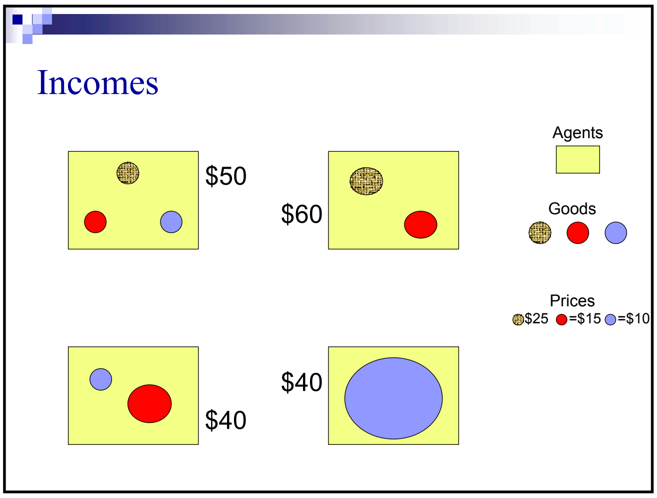
Utility function of agent *i*

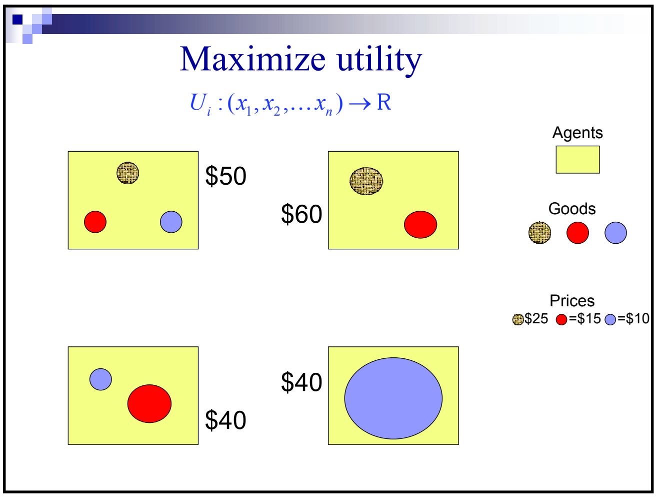
- $u_i : R^k \to R$
- Continuous, monotonic and strictly concave
- For any given prices and money *m*,
  there is a unique utility maximizing bundle
  for agent *i*.

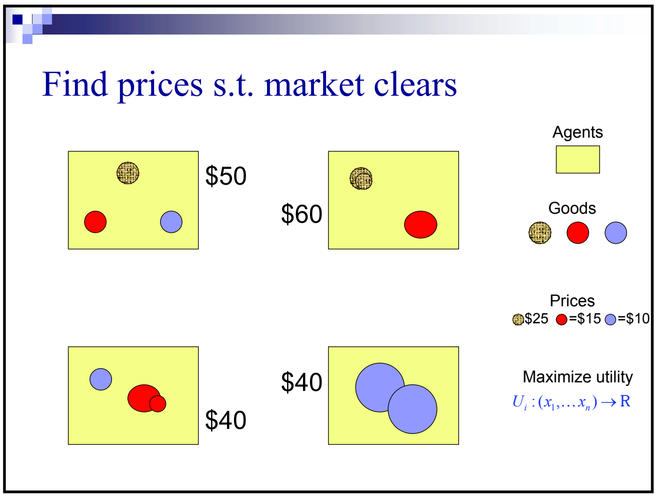
Arrow-Debreu Model



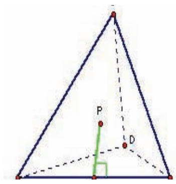








- Observe: If p is market clearing prices, then so is any scaling of p
- Assume w.l.o.g. that sum of prices of k goods is 1.
- Δ_k : k-1 dimensional unit simplex



Arrow-Debreu Theorem

- For continuous, monotonic, strictly concave utility functions, market clearing prices exist.

Proof

- Uses Kakutani's Fixed Point Theorem.
  - Deep theorem in topology

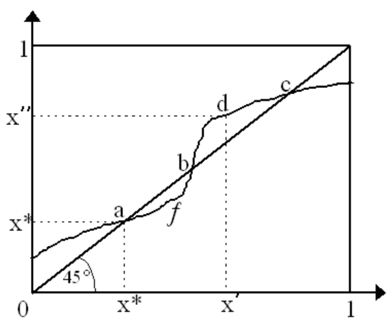
Proof

- Uses Kakutani's Fixed Point Theorem.
  - Deep theorem in topology
- Will illustrate main idea via Brouwer's Fixed Point Theorem (buggy proof!!)

Brouwer's Fixed Point Theorem

- Let $S \subseteq R^n$ be a non-empty, compact, convex set
- Continuous function $f : S \to S$
- Then $\exists x \in S : f(x) = x$

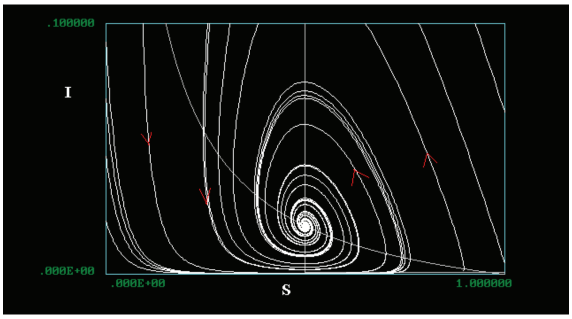
Brouwer's Fixed Point Theorem



Idea of proof

- Will define continuous function $f : \Delta_k \to \Delta_k$
- If p is not market clearing, $f(p)$ tries to 'correct' this.
- Therefore fixed points of f must be equilibrium prices.

Use Brouwer's Theorem



When is p an equilibrium price?

- *s(j):* total supply of good *j*.
- *B(i):* unique optimal bundle which agent *i* wants to buy after selling her initial endowment at prices *p*.
- *d(j):* total demand of good *j*.

When is p an equilibrium price?

- *s(j):* total supply of good *j*.
- *B(i):* unique optimal bundle which agent *i* wants to buy after selling her initial endowment at prices *p*.
- *d(j):* total demand of good *j*.
- For each good *j*: *s(j) = d(j)*.

What if p *is* not an equilibrium price?

- $s(j) < d(j)$ => $\uparrow$ $p(j)$
- $s(j) > d(j)$ => $\downarrow$ $p(j)$
- Also ensure $\underline{p} \in \Delta_k$

- Let $f(p) = p'$
- $S(j) < d(j)$ => $p'(j) = \frac{p(j) + [d(j) - s(j)]}{N}$
- $S(j) > d(j)$ => $p'(j) = \frac{p(j)}{N}$
- N is s.t. $\sum_j p'(j) = 1$

$\forall i : u_i$ is a cts. fn.

=> $\forall i : B(i)$ is a cts. fn. of $\underline{p}$

=> $\forall j : d(j)$ is a cts. fn. of $\underline{p}$

=> f is a cts. fn. of $\underline{p}$

$\forall i : u_i$ is a cts. fn.

=> $\forall i : B(i)$ is a cts. fn. of $\underline{p}$

=> $\forall j : d(j)$ is a cts. fn. of $\underline{p}$

=> f is a cts. fn. of $\underline{p}$

By Brouwer's Theorem, equilibrium prices exist.

$\forall i : u_i$ is a cts. fn.

=> $\forall i : B(i)$ is a cts. fn. of $\underline{p}$

=> $\forall j : d(j)$ is a cts. fn. of $\underline{p}$

=> f is a cts. fn. of $\underline{p}$

By Brouwer's Theorem, equilibrium prices exist.

q.e.d.!

Kakutani's Fixed Point Theorem

- $S \subseteq R^n$ convex, compact set
- $f : S \to 2^S$ non-empty, convex, upper hemi-continuous correspondence

$\Rightarrow \exists x \in S$ s.t. $x \in f(x)$

Fisher reduces to Arrow-Debreu

- Fisher: n buyers, k goods
- AD: $n+1$ agents
  - first n have money, utility for goods
  - last agent has all goods, utility for money only.

Combinatorial Algorithms for Market Equilibria

Vijay V. Vazirani

- Arrow-Debreu Theorem: Equilibria exist.

- Arrow-Debreu Theorem: Equilibria exist.
- Do markets operate at equilibria?

- Arrow-Debreu Theorem: Equilibria exist.
- Do markets operate at equilibria?
- Can equilibria be computed efficiently?

Arrow-Debreu is highly non-constructive

Arrow-Debreu is highly non-constructive

- "Invisible hand" of the market: Adam Smith
- Scarf, 1973: approximate fixed point algs.
- Convex programs:
  - Fisher: Eisenberg & Gale, 1957
  - Arrow-Debreu: Newman and Primak, 1992

- Used for deciding tax policies, price of new products etc.
- New markets on the Internet

Algorithmic Game Theory

- Use powerful techniques from modern algorithmic theory and notions from game theory to address issues raised by Internet.
- Combinatorial algorithms for finding market equilibria.

Two Fundamental Models

- Fisher's model
- Arrow-Debreu model, also known as exchange model

Combinatorial Algorithms

- Primal-dual schema based algorithms
  - Devanur, Papadimitriou, Saberi & V., 2002
    Combinatorial algorithm for Fisher's model
- Auction-based algorithms
  - Garg & Kapoor, 2004
    Approximation algorithms.

Approximation

- Find prices s.t. all goods clear
- Each buyer get goods providing at least $(1-\varepsilon)\times$optimal utility.

Primal-Dual Schema

- Highly successful algorithm design technique from exact and approximation algorithms

Exact Algorithms for Cornerstone Problems in P:

- Matching (general graph)
- Network flow
- Shortest paths
- Minimum spanning tree
- Minimum branching

Approximation Algorithms

set cover	facility location
Steiner tree	*k*-median
Steiner network	multicut
k-MST	feedback vertex set
scheduling . . .	

Main new idea

- Previous: problems captured via linear programs
- DPSV algorithm: problem captured via a nonlinear convex program

Fisher's Model

- *n* buyers, with specified money, *m(i)* for buyer *i*
- *k* goods (unit amount of each good)
- Linear utilities: u_{ij} is utility derived by *i* on obtaining one unit of *j*
- Total utility of *i*,

$$u_i = \sum_j u_{ij} x_{ij}$$

$$x_{ij} \in [0,1]$$

Fisher's Model

- *n* buyers, with specified money, *m(i)*
- *k* goods (each unit amount, w.l.o.g.)
- Linear utilities: u_{ij} is utility derived by *i* on obtaining one unit of *j*
- Total utility of *i*,

$$u_i = \sum_j u_{ij} x_{ij}$$

- Find prices s.t. market clears

Eisenberg-Gale Program, 1959

$$\max \sum_i m(i) \log u(i)$$

$$s.t.$$

$$\forall i : u(i) = \sum_j u_{ij} x_{ij}$$

$$\forall j : \sum_i x_{ij} \le 1$$

$$\forall ij : x_{ij} \ge 0$$

DPSV Algorithm

- "primal" variables: allocations of goods
- "dual" variables: prices
- algorithm: primal & dual improvements

Allocations Prices

Market Equilibrium

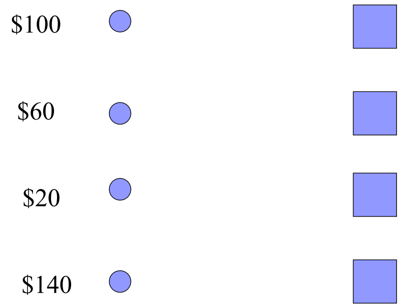
- Buyer i's optimization program:

$$\begin{array}{ll} \text{maximize} & \sum_j u_{ij} x_{ij} \\ \text{subject to} & \sum_j p_j x_{ij} \le m_i \end{array}$$

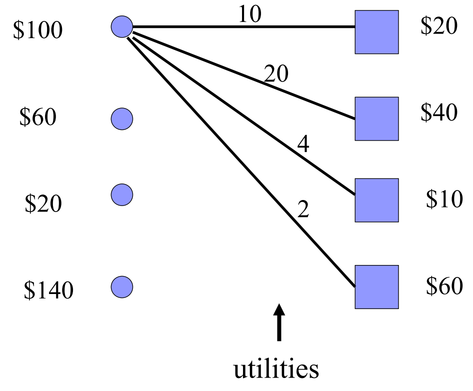
- Global Constraint:

$$\sum_i x_{ij} = 1 \qquad \forall j$$

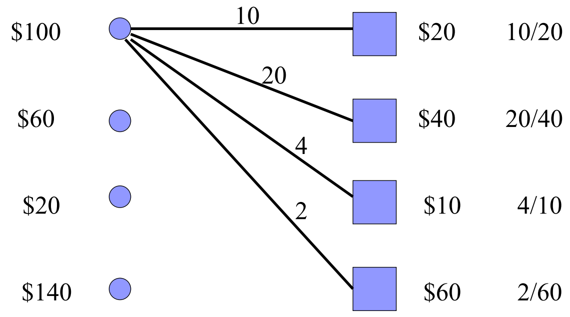
People Goods



Prices and utilities



Bang per buck

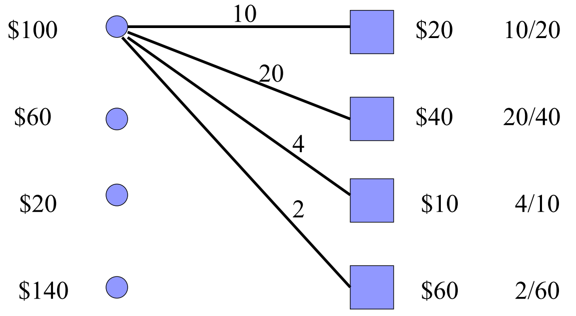


Bang per buck

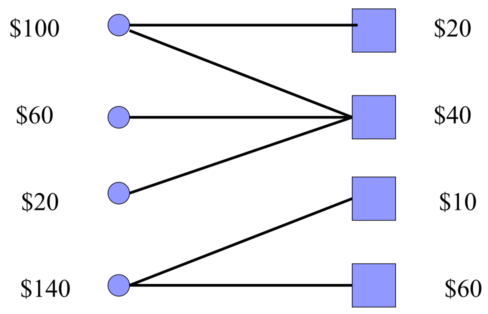
- Utility of $1 worth of goods
- Buyers will only buy goods providing maximum bang per buck

$$\max_j \frac{u_{ij}}{p_j}$$

Equality subgraph



Equality subgraph



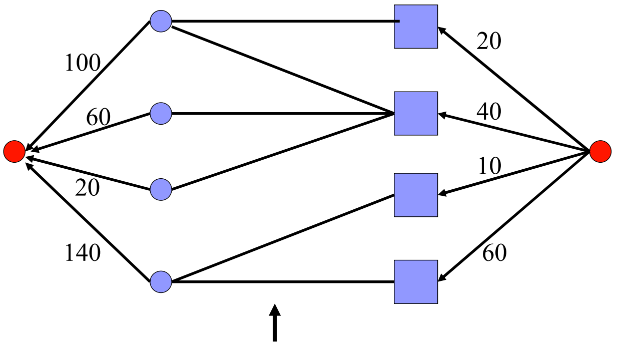
Most desirable goods for each buyer

- Any goods sold in equality subgraph make agents happiest
- How do we maximize sales in equality subgraph?

- Any goods sold in equality subgraph make agents happiest
- How do we maximize sales in equality subgraph?

Use max-flow!

Max flow



infinite capacities

Idea of Algorithm

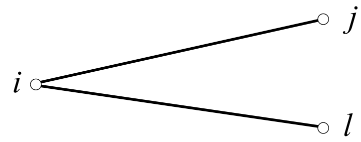
- Invariant: source edges form min-cut (agents have surplus)
- Iterations: gradually raise prices, decrease surplus
- Terminate: when surplus = 0, i.e., sink edges also form a min-cut

Ensuring Invariant initially

- Set each price to $1/n$
- Assume buyers' money integral

How to raise prices?

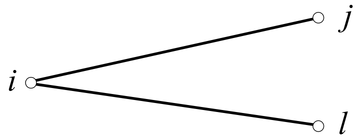
- Ensure equality edges retained



$$\frac{u_{ij}}{p_j} = \frac{u_{il}}{p_l}$$

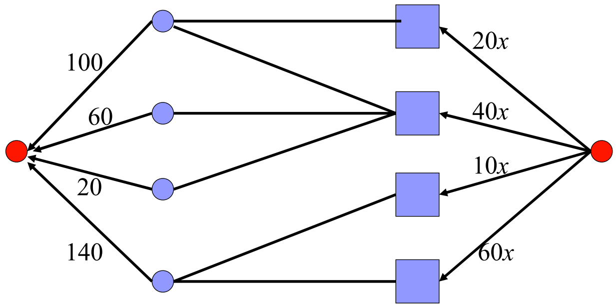
How to raise prices?

- Ensure equality edges retained



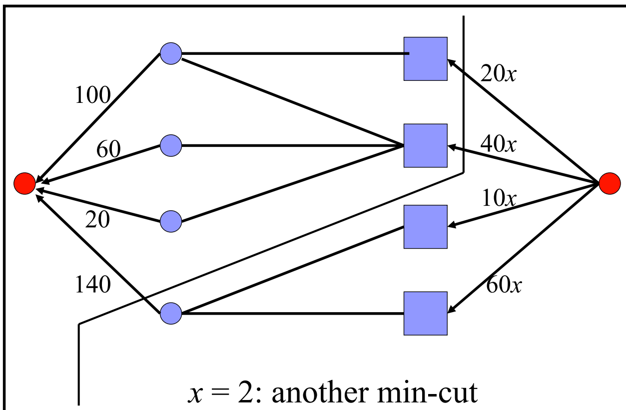
$$\frac{u_{ij}}{p_j} = \frac{u_{il}}{p_l}$$

- Raise prices proportionally $\frac{p_j}{p_l} = \frac{u_{ij}}{u_{il}}$



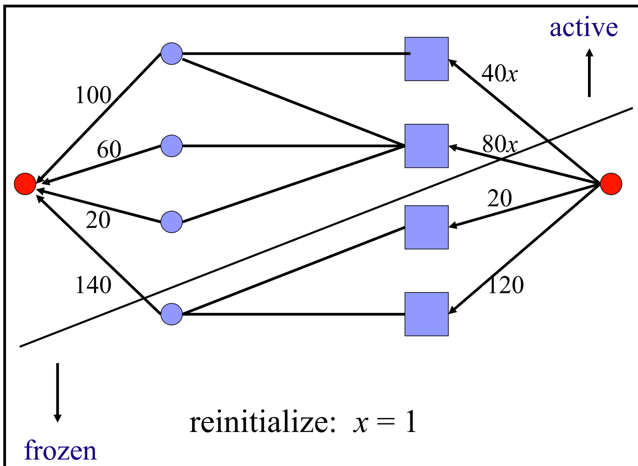
initialize: $x = 1$

$x \uparrow$

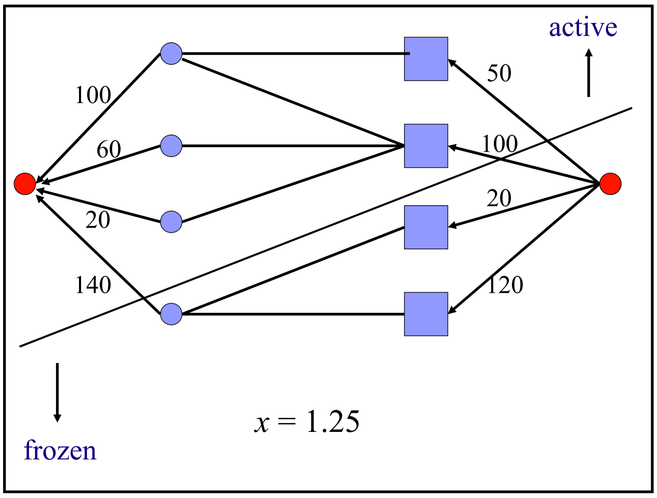


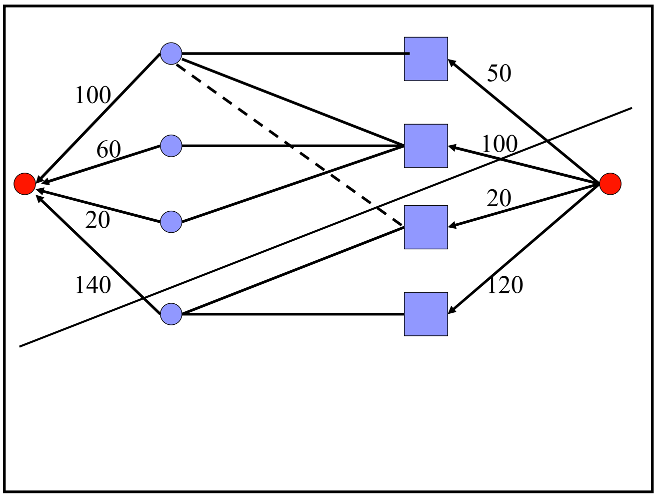
$x = 2$: another min-cut

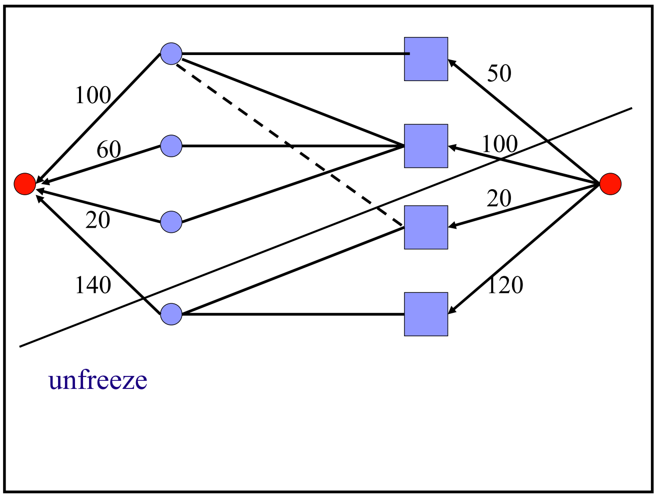
$x > 2$: Invariant violated

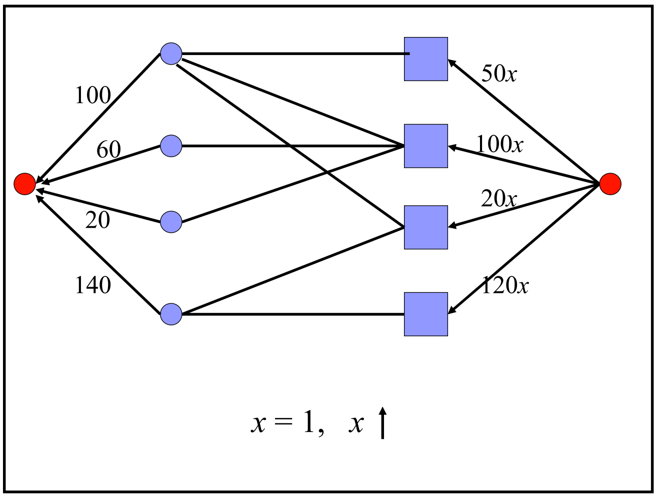


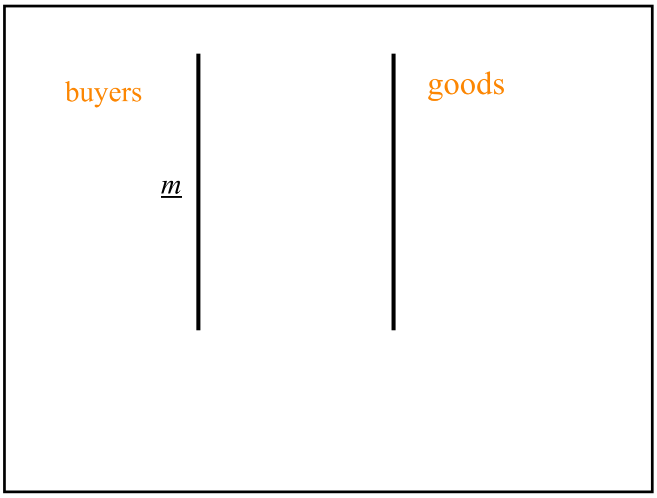
reinitialize: $x = 1$

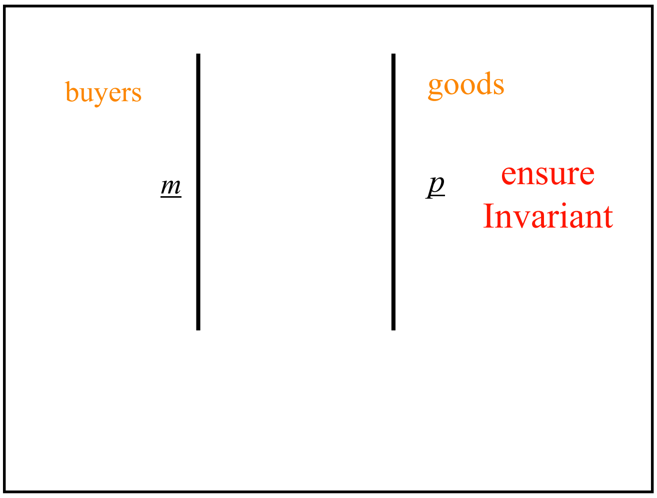

active
50
100
60
100
20
20
140
120
frozen
$x = 1.25$

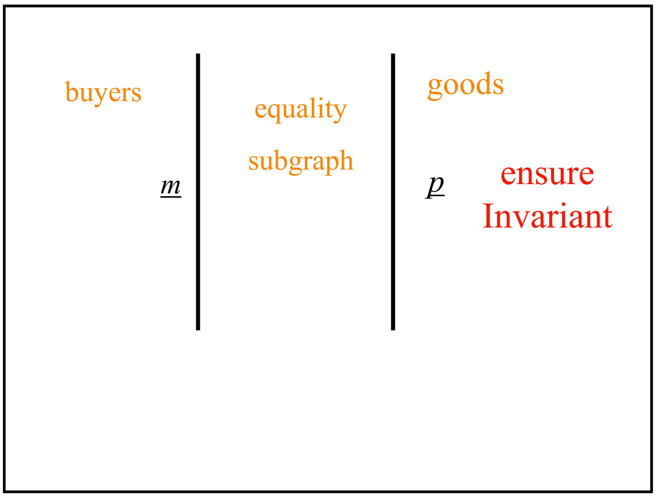

50
100
60
100
20
20
140
120

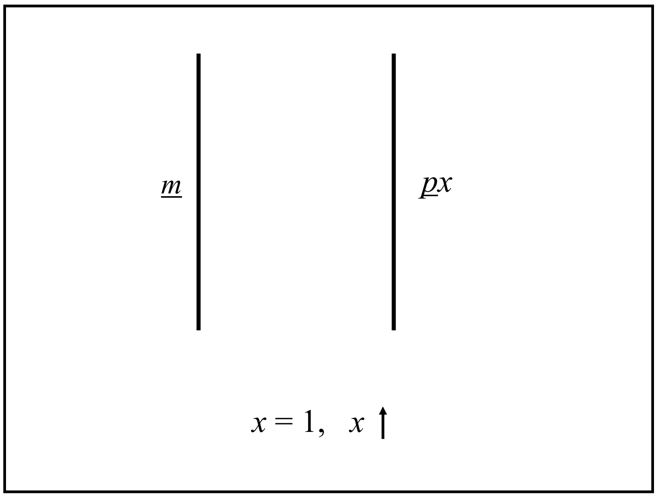

50
100
60
100
20
20
140
120
unfreeze

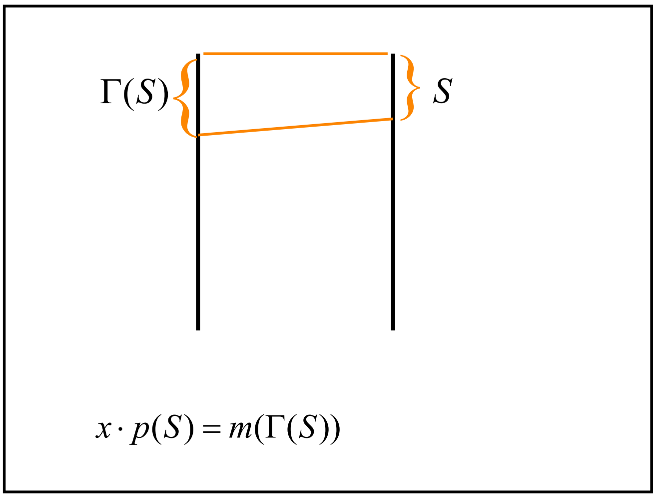

$50x$
100
60
$100x$
$20x$
20
140
$120x$
$x = 1, \quad x \uparrow$

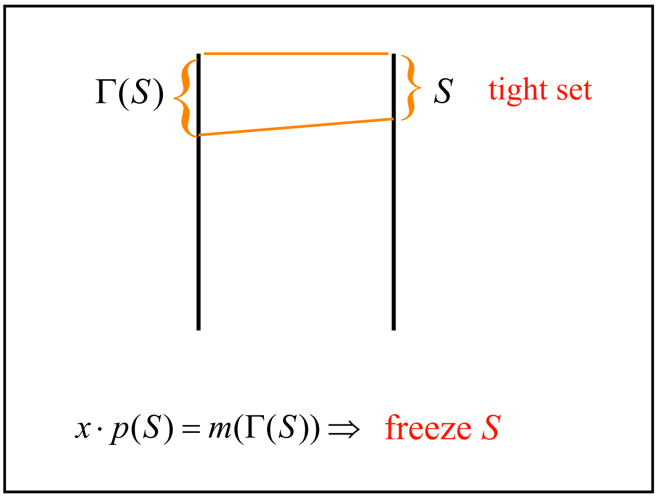

buyers
goods
$\underline{m}$

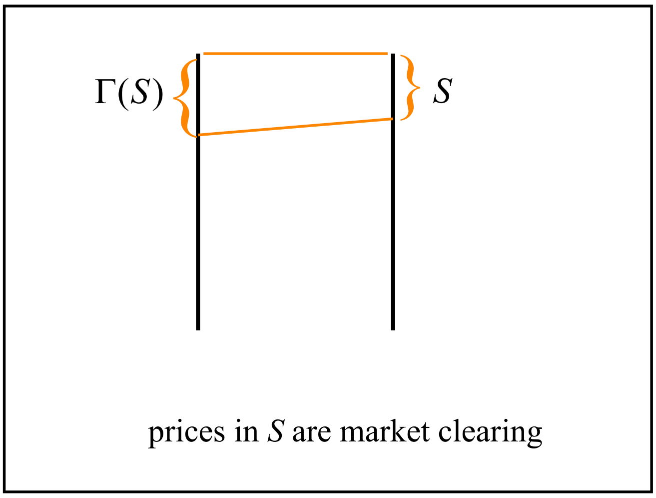

buyers
goods
$\underline{m}$
$\underline{p}$
ensure
Invariant

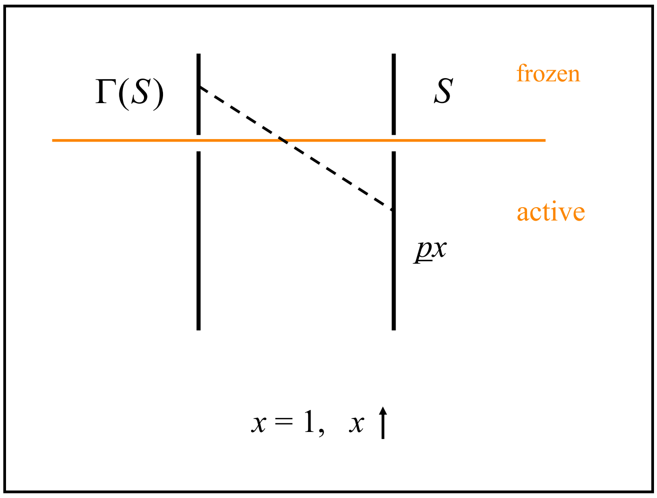

buyers
equality
subgraph
goods
$\underline{m}$
$\underline{p}$
ensure
Invariant

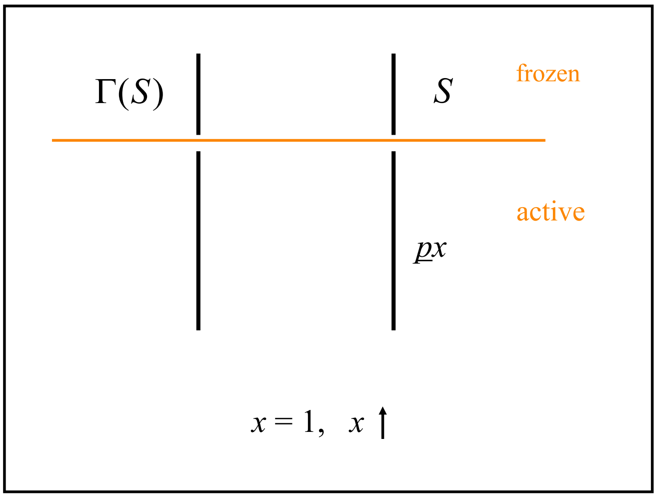

$\underline{m}$
$\underline{p}x$
$x = 1, \quad x \uparrow$

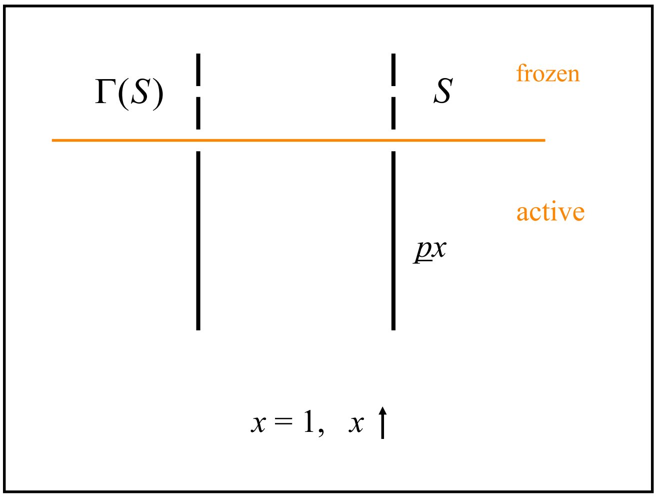

$\Gamma(S)$
S
$x \cdot p(S) = m(\Gamma(S))$

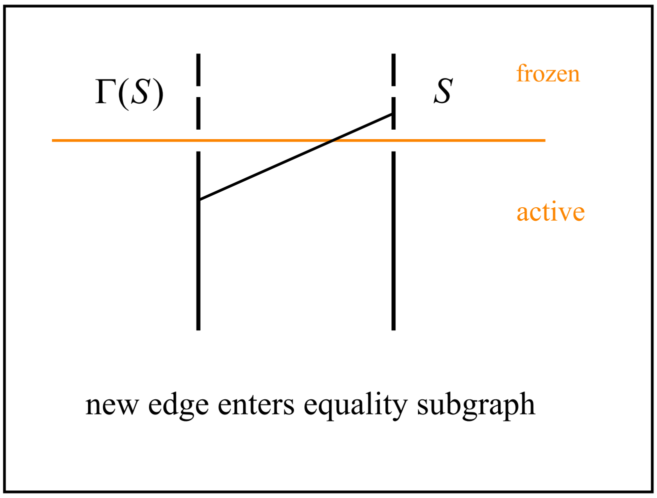

$\Gamma(S)$
S
tight set
$x \cdot p(S) = m(\Gamma(S)) \Rightarrow$ freeze S

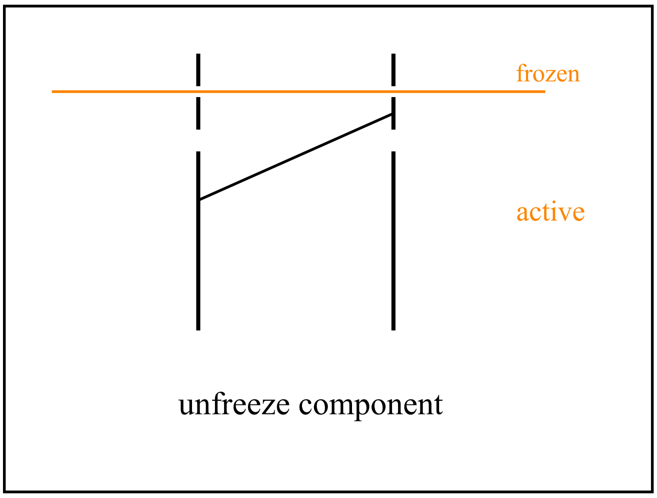

$\Gamma(S)$
S
prices in S are market clearing


$\Gamma(S)$
S
frozen
active
$\underline{p}x$
$x = 1, \quad x \uparrow$









- All goods frozen => terminate

  (market clears)

- All goods frozen => terminate

  (market clears)

- When does a new set go tight?

  •Solve as parametric cut problem

Termination

- Prices in S^* have denominators $\leq \Delta = nU^n$,
- Terminates in $Mn^2\Delta^2$ max-flows.

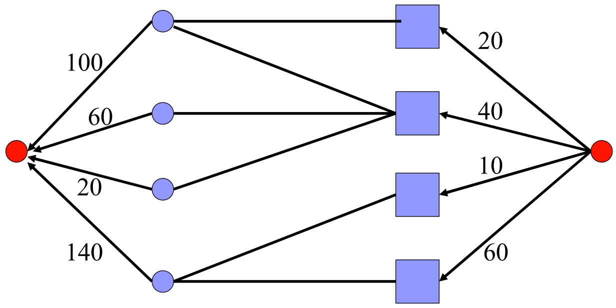
Polynomial time?

- Problem: very little price increase between freezings

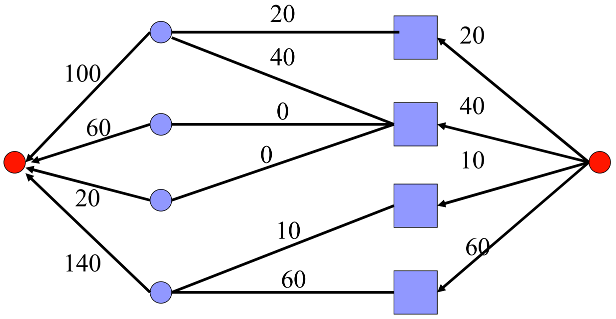
Polynomial time?

- Problem: very little price increase between freezings
- Solution: work with buyers having large surplus

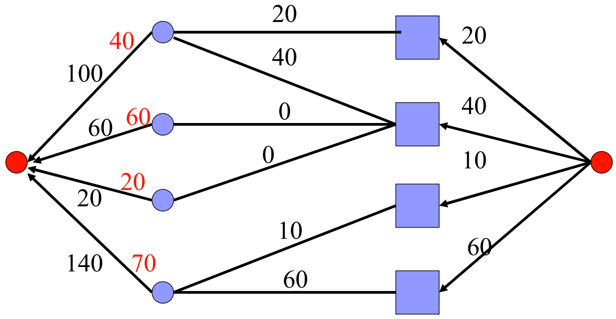
Max flow



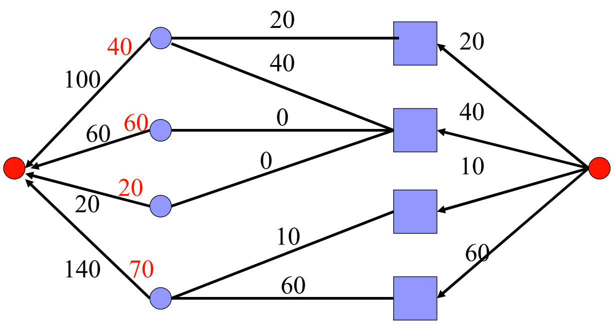
Max flow



$surplus(i) = m(i) - f(i)$



surplus(i) = m(i) – f(i)



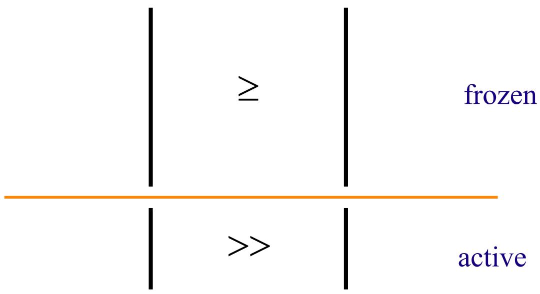
Surplus vector = (40, 60, 20, 70)

Balanced flow

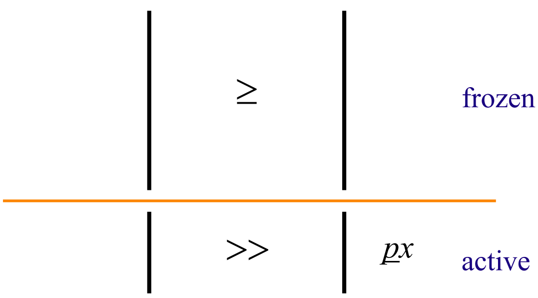
- A max-flow that minimizes l_2 norm of surplus vector
  - tries to make surpluses as equal as possible

Algorithm

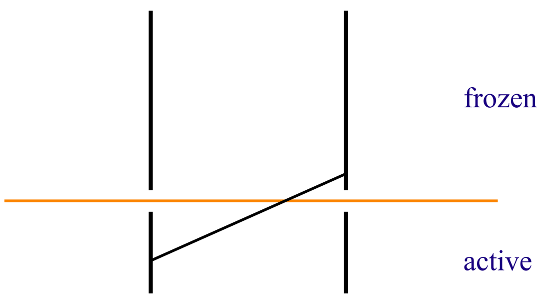
- Compute balanced flow



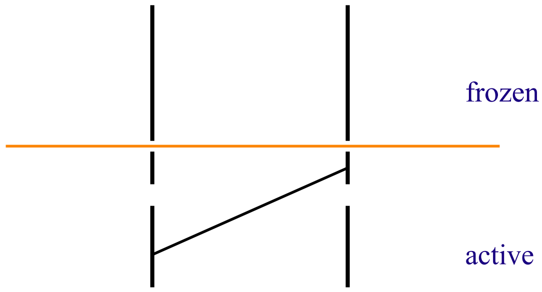
Active subgraph: Buyers with maximum surplus



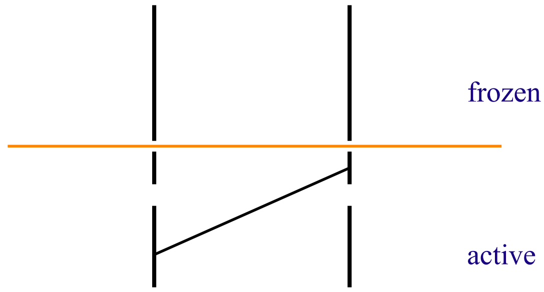
$x = 1, \quad x \uparrow$



new edge enters equality subgraph



Unfreeze buyers having residual path to active subgraph



Unfreeze buyers having residual path to active subgraph

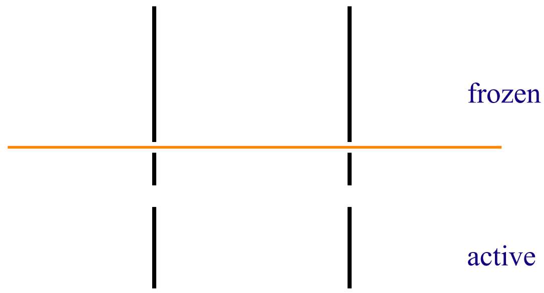
Do they have large surplus?

f: balanced flow

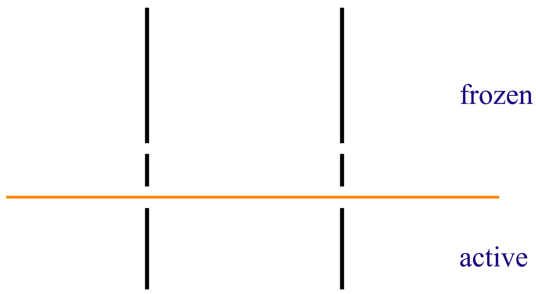
R(f): residual graph

Theorem: If *R(f)* has a path from *i* to *j* then

surplus(i) > *surplus(j)*



New set tight



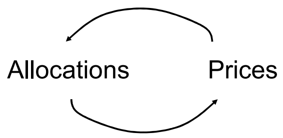
New set tight: freeze

Theorem: After each freezing, l_2 norm of surplus vector drops by $(1 - 1/n^2)$ factor.

- Two reasons:
  - total surplus decreases
  - flow becomes more balanced

Idea of Algorithm

- algorithm: primal & dual improvements



- measure of progress: l_2-norm of surplus vector

Polynomial time

Theorem: $O(n^2(n \log U + \log Mn^2))$

max-flow computations suffice.

Weak gross substitutability

- Increasing price of one good cannot decrease demand for another good.

Weak gross substitutability

- Increasing price of one good cannot decrease demand for another good.

=> never need to decrease prices (dual variables).

Weak gross substitutability

- Increasing price of one good cannot decrease demand for another good.

=> never need to decrease prices (dual variables).

- Almost all primal-dual algs work this way.

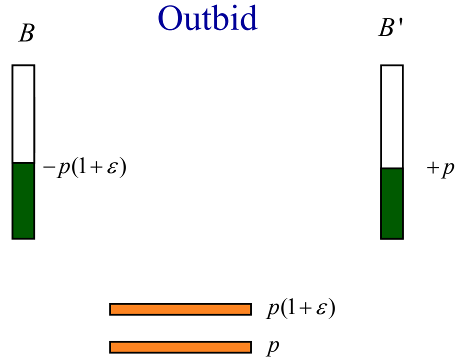
Arrow-Debreu Model

- Approximate equilibrium algorithms:
  - Jain, Mahdian & Saberi, 2003: Use DPSV as black box.
  - Devanur & V., 2003: More efficient, by opening DPSV.

Garg & Kapoor, 2004
Auction-based algorithm

- Start with very low prices
- Keep increasing price of good that is in demand
- *B* has excess money. Favorite good: *g*
  Currently at price *p* and owned by *B'*
- *B* outbids *B'*

Outbid



Auction-based algorithm

- Go in rounds: $B_1, B_2, \ldots B_n$
- In each round, total surplus decreases by $(1+\varepsilon)$ factor
- Hence $\log_{(1+\varepsilon)} M$ iterations suffice,
  M= total money

Arrow-Debreu Model

- Start with all prices 1
- Allocate money to agents (initial endowment)
- Perform outbid and update agents' money

Arrow-Debreu Model

- Start with all prices 1
- Allocate money to agents (initial endowment)
- Perform outbid and update agents' money
- Any good with price >1 is fully sold

Arrow-Debreu Model

- Start with all prices 1
- Allocate money to agents (initial endowment)
- Perform outbid and update agents' money
- Any good with price >1 is fully sold

$$\frac{\max price}{\min price} \leq \frac{\max u_{ij}}{\min u_{ij}}$$

- Eventually every good will have price >1

- Garg, Kapoor & V., 2004:

Auction-based algorithms for
additively separable concave utilities
satisfying weak gross substitutability

- Kapoor, Mehta & V., 2005:

Auction-based algorithm for
a (restricted) production model

Q: Distributed algorithm for equilibria?

- Appropriate model?
- Primal-dual schema operates via
local improvements

New Market Models

Resource Allocation Markets

Vijay V. Vazirani

Fisher's Model

- *n* buyers, with specified money, *m(i)* for buyer *i*
- *k* goods (unit amount of each good)
- Linear utilities: u_{ij} is utility derived by *i* on obtaining one unit of *j*
- Total utility of *i*,

$$u_i = \sum_j u_{ij} x_{ij}$$
$$x_{ij} \in [0,1]$$

Fisher's Model

- *n* buyers, with specified money, *m(i)*
- *k* goods (each unit amount, w.l.o.g.)
- Linear utilities: u_{ij} is utility derived by *i* on obtaining one unit of *j*
- Total utility of *i*,

$$u_i = \sum_j u_{ij} x_{ij}$$

- Find prices s.t. market clears

Eisenberg-Gale Program, 1959

$$\max \sum_i m(i) \log u(i)$$
$$s.t.$$
$$\forall i : u(i) = \sum_j u_{ij} x_{ij}$$
$$\forall j : \sum_i x_{ij} \le 1$$
$$\forall ij : x_{ij} \ge 0$$

Via KKT Conditions can establish:

- Optimal solution gives equilibrium allocations
- Lagrange variables give prices of goods

Eisenberg-Gale program helps establish:

- Equilibrium exists (under mild conditions)
- Equilibrium utilities and prices are unique

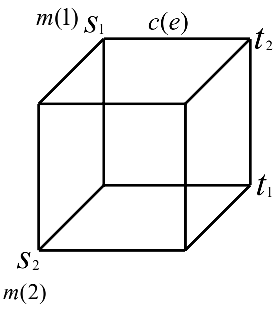
Eisenberg-Gale program helps establish:

- Equilibrium exists (under mild conditions)
- Equilibrium utilities and prices are unique
- Rational!!

Kelly's resource allocation model, 1997

Mathematical framework for understanding TCP congestion control

Kelly's model



Given:

network $G = (V,E)$

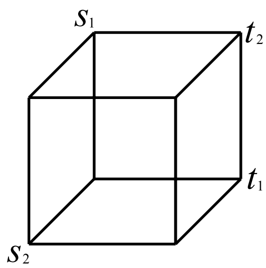
(directed or undirected)

capacities on edges

source-sink pairs (agents)

m(i): money/unit time agent *i* is willing to pay

Kelly's model



Network determines:

f(i): flow rate of agent *i*

Assume utility $u(i) = m(i) \log f(i)$

Total utility is additive

Convex Program for Kelly's Model

$$\max \sum_i m(i) \log f(i)$$

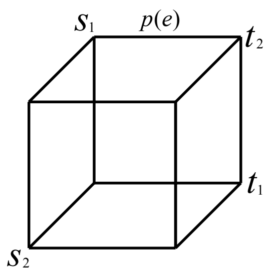
$$s.t.$$

$$\forall i : f(i) = \sum_p f_i^p$$

$$\forall e : flow(e) \le c(e)$$

$$\forall i,p : f_i^p \ge 0$$

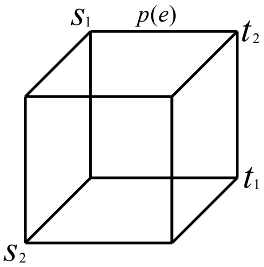
Kelly's model



Lagrange variables:

p(e): price/unit flow

Kelly's model



Optimum flow and edge prices are in equilibrium:

1). $p(e)>0$ only if e is saturated

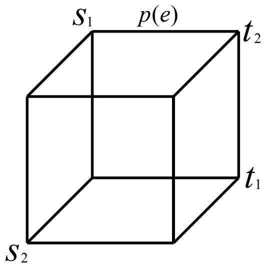
2) flows go on cheapest paths

3) money of each agent is fully used

Let rate(i) = cost of cheapest path for i

$m(i) = f(i)$ rate(i)

Kelly's model



Optimum flow and edge prices are in equilibrium:

1). $p(e)>0$ only if e is saturated

2) flows go on cheapest paths

3) money of each agent is fully used

Let rate(i) = cost of cheapest path for i

$f(i)$'s and rate(i)'s are unique!

TCP Congestion Control

- $f(i)$: source rate
- $p(e)$: prob. of packet loss (in TCP Reno)
  queueing delay (in TCP Vegas)

TCP Congestion Control

- $f(i)$: source rate
- $p(e)$: prob. of packet loss (in TCP Reno)
  queueing delay (in TCP Vegas)

Kelly: Equilibrium flows are proportionally fair: only way of increasing an agent's flow by 5% is to decrease other agents' flow by at least 5%

TCP Congestion Control

- $f(i)$: source rate
- $p(e)$: prob. of packet loss (in TCP Reno)
  queueing delay (in TCP Vegas)

- Low, Doyle, Paganini: continuous time algs. for computing equilibria (not poly time).

TCP Congestion Control

- $f(i)$: source rate
- $p(e)$: prob. of packet loss (in TCP Reno)
  queueing delay (in TCP Vegas)

- Low, Doyle, Paganini: continuous time algs. for computing equilibria (not poly time).

- AIMD + RED converges to equilibrium primal-dual (source-link) alg.

TCP Congestion Control

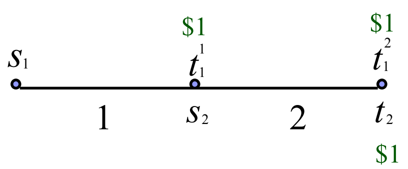
- *f(i):* source rate
- *p(e):* prob. of packet loss (in TCP Reno)
  queueing delay (in TCP Vegas)
- Low, Doyle, Paganini: continuous time algs. for computing equilibria (not poly time).
- FAST: for high speed networks with large bandwidth

Combinatorial Algorithms

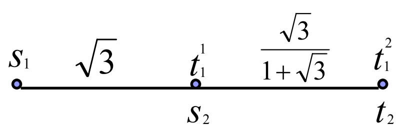
- Devanur, Papadimitriou, Saberi & V., 2002: for Fisher's linear utilities case
- Kelly & V., 2002: Kelly's model is a generalization of Fisher's model.

Find comb. poly time algs!

Irrational for 2 sources & 3 sinks

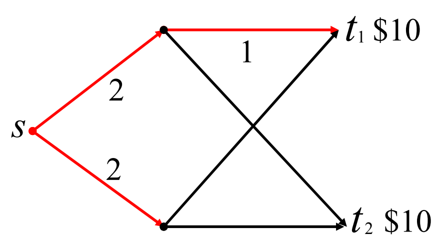


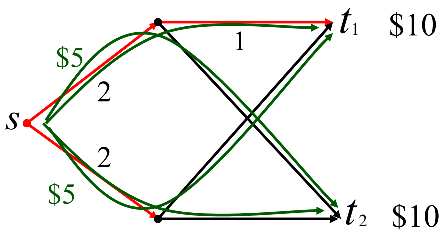
Irrational for 2 sources & 3 sinks

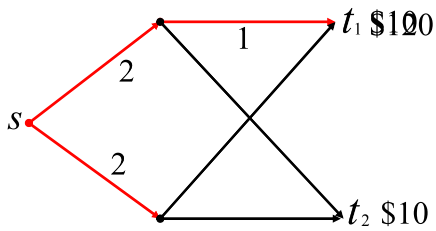


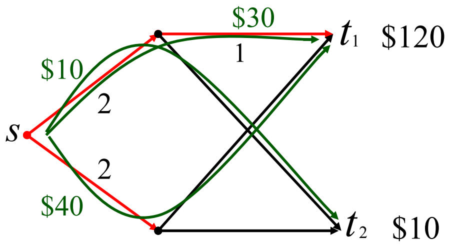
Equilibrium prices

- 1 source & multiple sinks
- 2 source-sink pairs



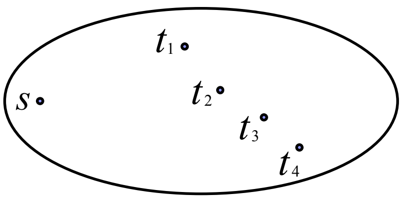




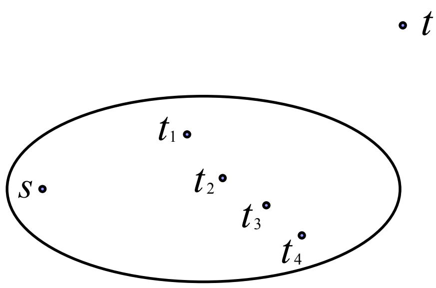


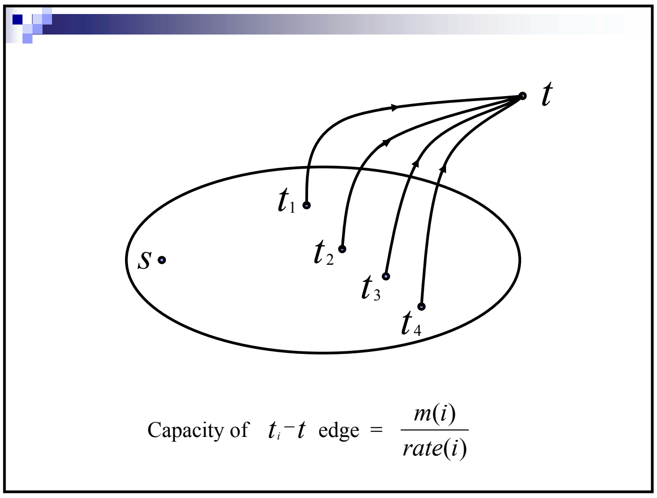
Jain & V., 2005: strongly poly alg

- Primal-dual algorithm
  - Usual: linear programs & LP-duality
  - This: convex programs & KKT conditions
- Ascending price auction
  - Buyers: sinks (fixed budgets, maximize flow)
  - Sellers: edges (maximize price)

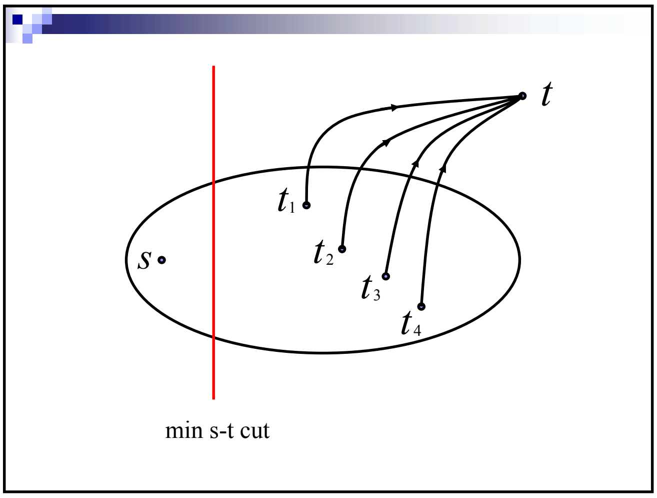


rate(i): cost of cheapest $s - t_i$ path

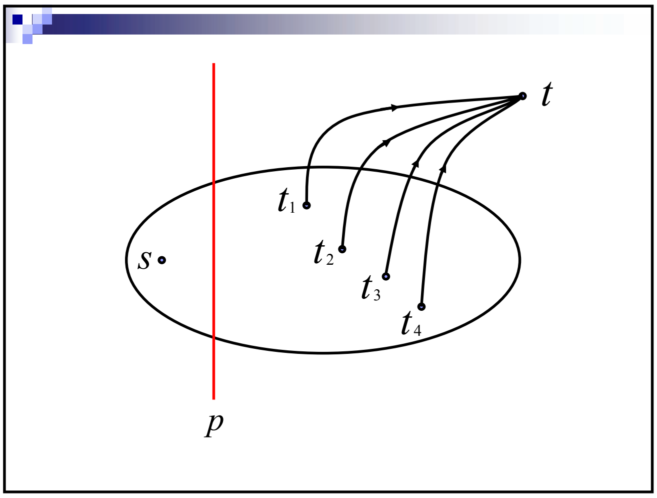




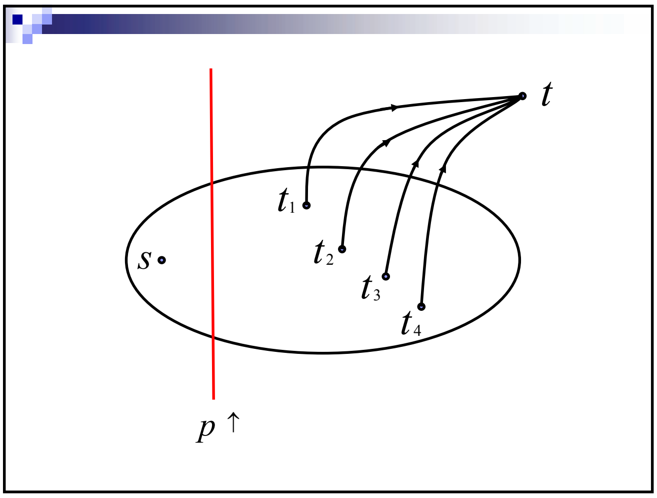
t
t_1
s
t_2
t_3
t_4
Capacity of $t_i - t$ edge = $\frac{m(i)}{rate(i)}$



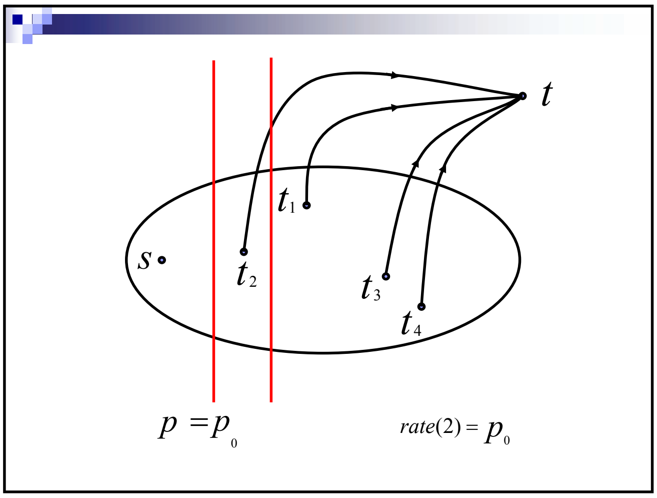
t
t_1
s
t_2
t_3
t_4
min s-t cut



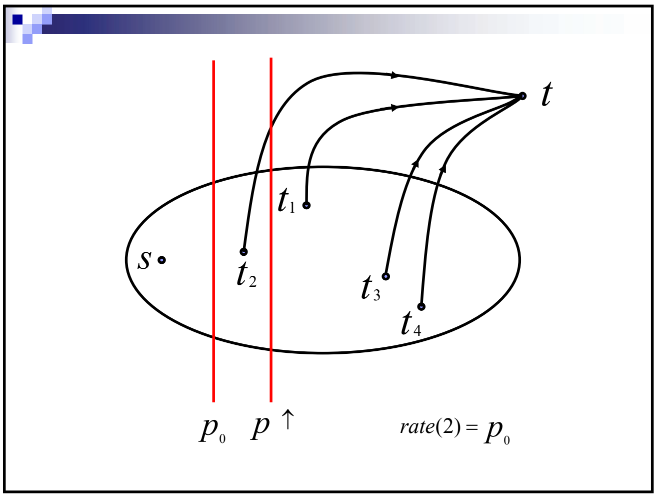
t
t_1
s
t_2
t_3
t_4
p



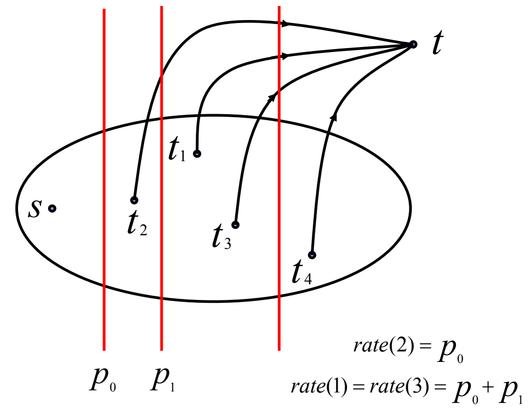
t
t_1
s
t_2
t_3
t_4
$p \uparrow$

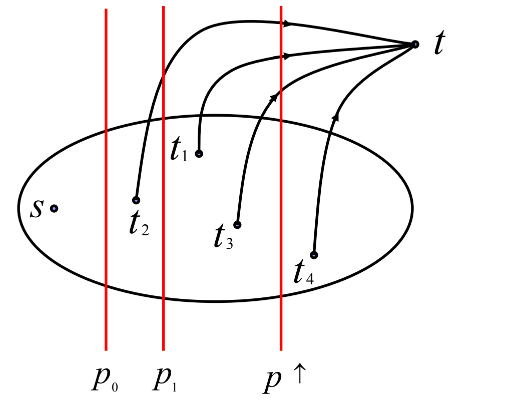


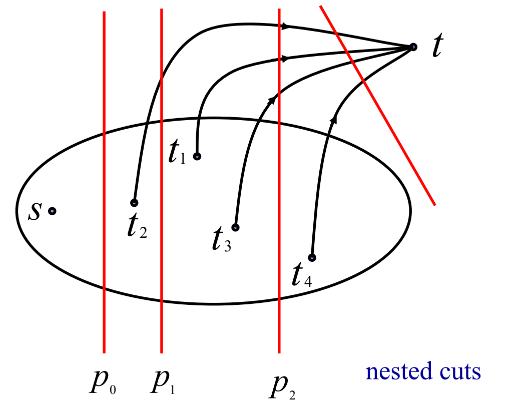
t
t_1
s
t_2
t_3
t_4
$p = p_0$
$rate(2) = p_0$

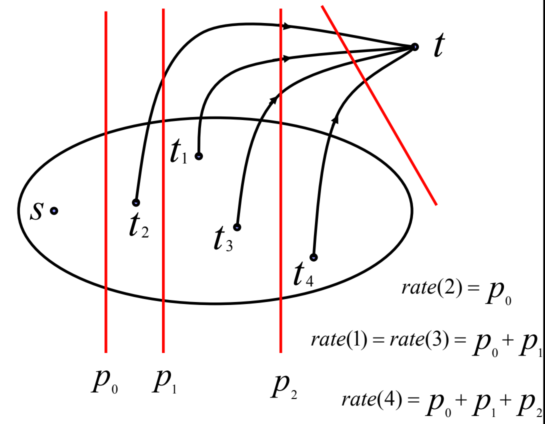


t
t_1
s
t_2
t_3
t_4
p_0 $p \uparrow$
$rate(2) = p_0$

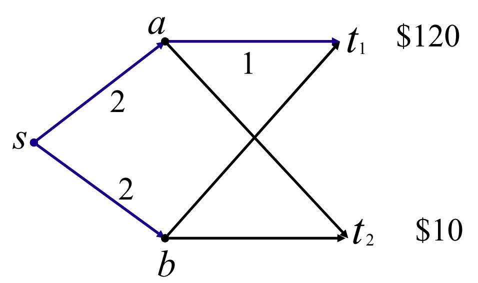


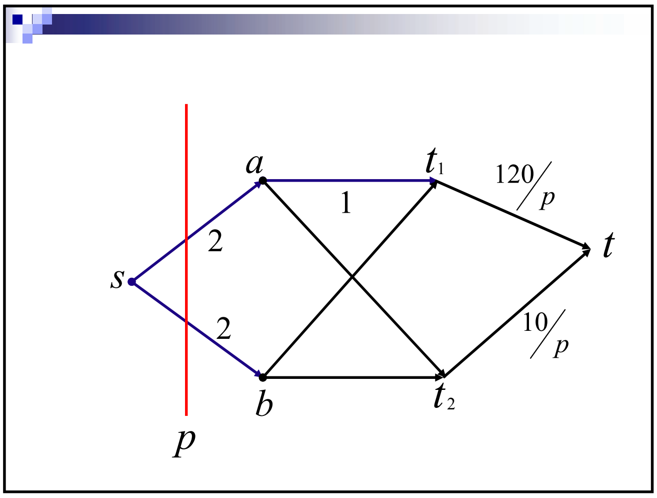




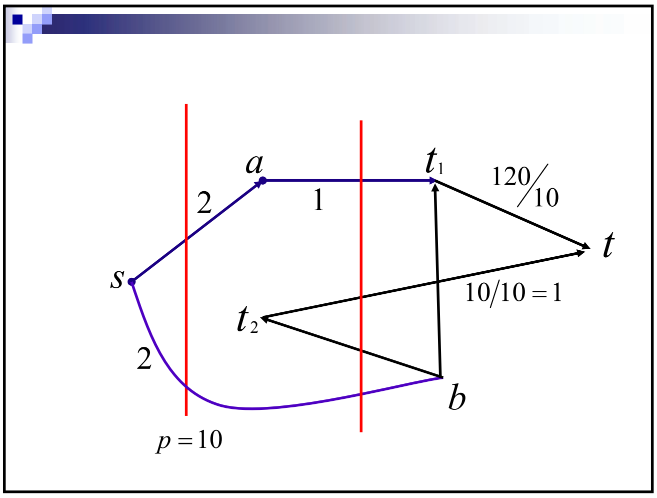


- Find s-t max flow
- Flow and prices will:
  - Saturate all red cuts
  - Use up sinks' money
  - Send flow on cheapest paths

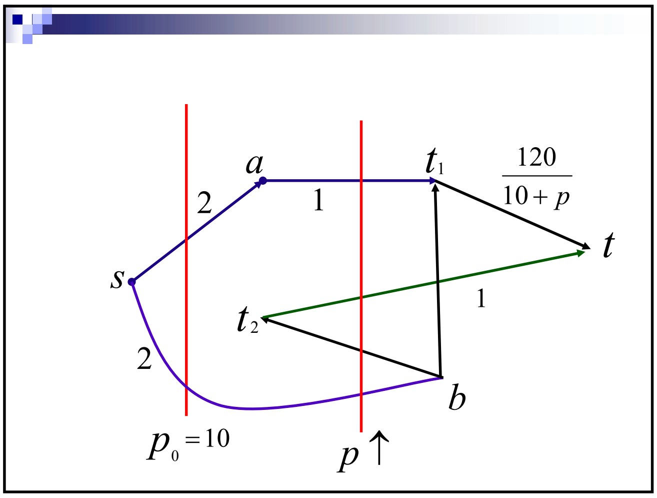




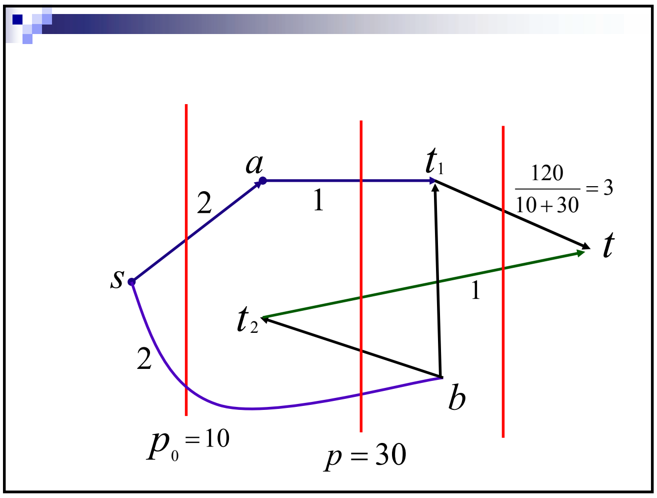
a
t_1
120/p
1
2
s
t
2
10/p
b
t_2
p



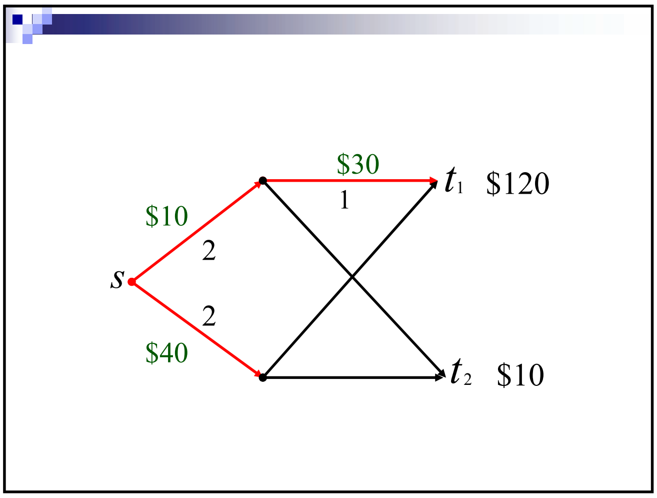
a
t_1
120/10
2
1
s
t
t_2
10/10 = 1
2
b
p = 10



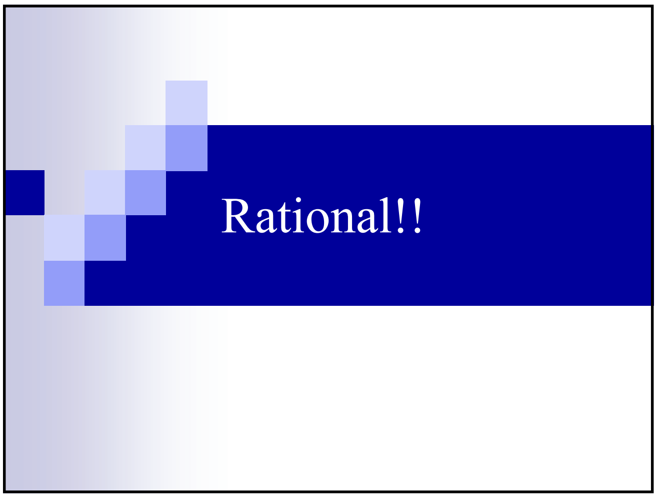
a
t_1
$\frac{120}{10+p}$
2
1
s
t
t_2
1
2
b
$p_0 = 10$
$p\uparrow$



a
t_1
$\frac{120}{10+30} = 3$
2
1
s
t
t_2
1
2
b
$p_0 = 10$
p = 30



$30
t_1
$120
$10
1
2
s
2
$40
t_2
$10



Rational!!

Max-flow min-cut theorem

Other resource allocation markets

- 2 source-sink pairs (directed/undirected)
- Branchings rooted at sources (agents)
- Spanning trees
- Network coding

Eisenberg-Gale-Type Convex Program

$$\max \sum_i m(i) \log u(i)$$

s.t. packing constraints

Eisenberg-Gale Market

- A market whose equilibrium is captured as an optimal solution to an Eisenberg-Gale-type program

---

- Megiddo, 1974: Let T = set of sinks (agents)
- For $S \subseteq T$ define $v(S)$ to be the max-flow possible from s to sinks in S.
- Then v is a submodular function, i.e., for

$$A \subseteq B \subseteq T,$$
$$t \notin A,$$
$$v(B+t) - v(B) \le v(A+t) - v(A)$$

Simpler convex program for single-source market

$$\max \sum_i m(i) \log f(i)$$
$$s.t.$$
$$\forall S \subseteq T : \sum_{i \in S} f(i) \le v(S)$$
$$\forall i : f(i) \ge 0$$

Submodular Utility Allocation Market

- Any market which has simpler program and v is submodular

Submodular Utility Allocation Market

- Any market which has simpler program and v is submodular
- Theorem: Strongly polynomial algorithm for SUA markets.

Submodular Utility Allocation Market

- Any market which has simpler program and v is submodular
- Theorem: Strongly polynomial algorithm for SUA markets.
- Corollary: Rational!!

- Theorem: Following markets are SUA:
  - 2 source-sink pairs, undirected (Hu, 1963)
  - spanning tree (Nash-William & Tutte, 1961)
  - 2 sources branching (Edmonds, 1967 + JV, 2005)
- 3 sources branching: irrational

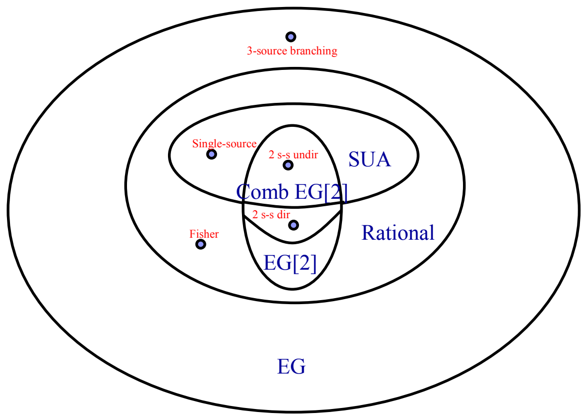
- Theorem: Following markets are SUA:
  - 2 source-sink pairs, undirected (Hu, 1963)
  - spanning tree (Nash-William & Tutte, 1961)
  - 2 sources branching (Edmonds, 1967 + JV, 2005)
- 3 sources branching: irrational
- Open (no max-min thoerems):
  - 2 source-sink pairs, directed
  - 2 sources, network coding

- Theorem: Following markets are SUA:
  - 2 source-sink pairs, undirected (Hu, 1963)
  - spanning tree (Nash-William & Tutte, 1961)
  - 2 sources branching (Edmonds, 1967 + JV)
- 3 sources branching: irrational
- Open (no max-min thoerems):
  - 2 source-sink pairs, directed
  - 2 sources, network coding

Chakrabarty, Devanur & V., 2006

- EG[2]: Eisenberg-Gale markets with 2 agents
- Theorem: EG[2] markets are rational.

- EG[2]: Eisenberg-Gale markets with 2 agents
- Theorem: EG[2] markets are rational.
- Combinatorial EG[2] markets: polytope of feasible utilities can be described via combinatorial LP.
- Theorem: Strongly poly alg for Comb EG[2].



Other properties:

- Efficiency
- Fairness (max-min + min-max fair)
- Competition monotonicity

Open issues

- Strongly poly algs for approximating
  - nonlinear convex programs
  - equilibria
- Insights into congestion control protocols?

Random Sampling Techniques and Approximation of CSP Problems

Marek Karpinski
University of Bonn

(NHC Spring School Lectures, Tokyo, March 1, 2006)

Abstract. We present some recent results and new *sampling* techniques for *absolute* and *relative* approximation of general *Constraint Satisfaction Problems* (CSP). The methods used are threefold and based on: Smooth or Linearized Integer Programs, combinatorial arguments, and special linear algebraic techniques. In particular we apply those techniques to construct *polynomial time approximation schemes (PTASs)* for certain instances of both *MAX*- and *MIN-CSP* including dense and subdense instances and general *metric* and *quasimetric* instances of those problems. In that context we study the generic sample complexity for approximating arbitrary CSP instances and try to establish tight upper bounds for their underlying core-sample sizes. We go also beyond CSP optimization problems and design first PTASs for general *metric* and *quasimetric* size-constraint Partitioning Problems.

Selected References

[1] N. Alon, W. F. de la Vega, R. Kannan and M. Karpinski, *Random Sampling and Approximation of MAX-CSPs*, J. Computer and System Sciences 67 (2003), 212-243.

[2] S. Arora, D. Karger and M. Karpinski, *Polynomial Time Approximation Schemes for Dense Instances of NP-Hard Problems*, J. Computer and System Sciences 58 (1999), 193-210.

[3] W. F. de la Vega, R. Kannan, M. Karpinski and S. Vempala, *Tensor Decomposition and Approximation Schemes for Constraint Satisfaction Problems*, Proc. 27th ACM STOC (2005), 747-754.

[4] W. F. de la Vega and M. Karpinski, *Approximation Complexity of Nondense Instances of MAX-CUT*, in preparation, 2006.

[5] W. F. de la Vega, M. Karpinski and C. Kenyon, *Approximation Schemes for Metric Bisection and Partitioning*, Proc. 15th ACM-SIAM SODA (2004), 506-515.

[6] W. F. de la Vega, M. Karpinski, C. Kenyon and Y. Rabani, *Approximation Schemes for Clustering Problems*, Proc. 35th ACM STOC (2003), 50-58.

[7] M. Karpinski, *Polynomial Time Approximation Schemes for Some Dense Instances of NP-Hard Problems*, Algorithmica 30 (2001), 386-397.

□

APPROXIMATION SCHEMES FOR METRIC BISECTION AND PARTITIONING

MAREK KARPINSKI
UNIVERSITY OF BONN

0

APPROXIMATING METRIC BISECTION AND RELATED PARTITIONING PROBLEMS

MAREK KARPINSKI
UNIVERSITY OF BONN

(JOINT WORK WITH W.F. DE LA VEGA AND CLAIRE KENYON)

1

MOTIVATED BY
“BALANCED“ METRIC
AND QUASIMETRIC
CLUSTERING
AND
PARTITIONING
PROBLEMS

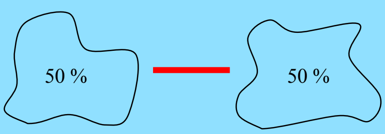
2

MOTIVATED BY
“BALANCED“ METRIC
AND QUASIMETRIC
CLUSTERING
AND
PARTITIONING
PROBLEMS

(MINIMIZING HAMILTONIANS)

2

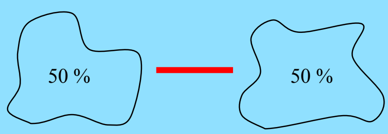
MIN-BISECTION
OF
GRAPHS,
METRIC SPACES,
SET SYSTEMS, ...



´MIN-CONNECTIVITY´

3

MIN-BISECTION
OF
GRAPHS,
METRIC SPACES, √
SET SYSTEMS, ...



´MIN-CONNECTIVITY´

3

“DENSITY“

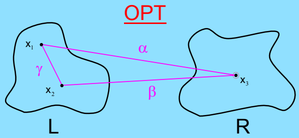
$\Downarrow$

METRIC SITUATIONS:
PARTITIONING
OF
FINITE METRIC SPACES (X, d).

4

[METRIC PARTITIONING PROPERTY:]

OPT



$W = \sum_{V \times V} d(x, y)$,

$W_A = \sum_{A \times A} d(x, y)$.

$Av\_\alpha \geq \frac{1}{2} \cdot Av\_\gamma \quad \Rightarrow \quad \boxed{W_L + W_R \leq 4 \cdot OPT}$

WE HAVE: $W = W_L + W_R + 2 \cdot OPT$.
SO, $OPT \geq W/6$.

5

$\Downarrow$

RED. TO ABSOLUTE (ADD.)
APPROX. TO WITHIN $\varepsilon \cdot W$
FOR EVERY $\varepsilon > 0$.

DENSE
METHOD.

6

THE PROBLEM:
COMPUTING MIN/MAX BISECTION
OF A GIVEN METRIC SPACE (V, d).

THE PROBLEM
KNOWN TO BE NP-HARD EVEN
WHEN RESTRICTED
TO (1,2)-METRIC.

7

METRIC SPACE (V, d)
EQUICUT:
PARTITIONING*) (V_1, V_2) OF V
WITH $|V_1| = |V_2|$.

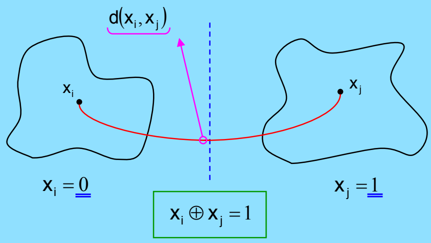
MINIMIZING/
MAXIMIZING

$$\sum_{\substack{\{v,w\}\ \text{CUT} \\ \text{BY}\ (V_1, V_2)}} d(v, w)$$

8

$\Downarrow$ *)

$\boxed{\mathscr{S} = \{ x_i \oplus x_j = 1 \mid \{i, j\} \in E \}}$



9

MIN-CSP

$\oplus$

GLOBAL CONSTRAINT:

*) $\boxed{\sum_{i=1}^{n} x_i = n/2}$

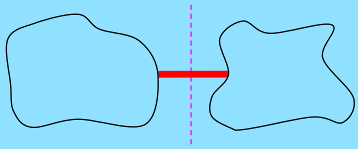
$\uparrow$

[• MIN-BISECTION]

10

GIVEN A FINITE METRIC SPACE (V,d).

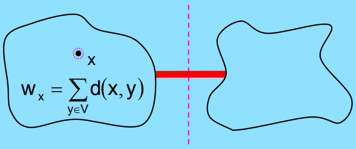
HOW LARGE
A VALUE OF
AN ARBITRARY EQUICUT ?



11

GIVEN A FINITE METRIC SPACE (V,d).

HOW LARGE
A VALUE OF
AN ARBITRARY EQUICUT ?



$W = \sum_{x \in V} w_x$

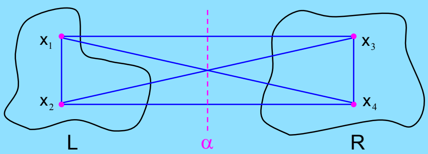
11

$$W_L = \sum_{L \times L} d(x,y) \; ,$$

$$W_R = \sum_{R \times R} d(x,y) \; .$$

12

$\boxed{\text{MIN-BISECTION}((V,d)) \geq W/6 \, .}$



$\alpha = d(x_1,x_3)+d(x_1,x_4)+d(x_2,x_3)+d(x_2,x_4)$

$\Downarrow$ (Δ)

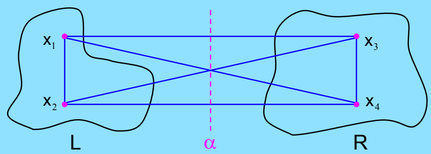
$d(x_1,x_2)+d(x_3,x_4) \leq \alpha$

$W_L + W_R \leq 4$ OPT

$W \leq 6$ OPT

13

$\boxed{\text{MIN-BISECTION}((V,d)) \geq W/6 \, .}$



$\alpha = d(x_1,x_3)+d(x_1,x_4)+d(x_2,x_3)+d(x_2,x_4)$

$\Downarrow$ (Δ)

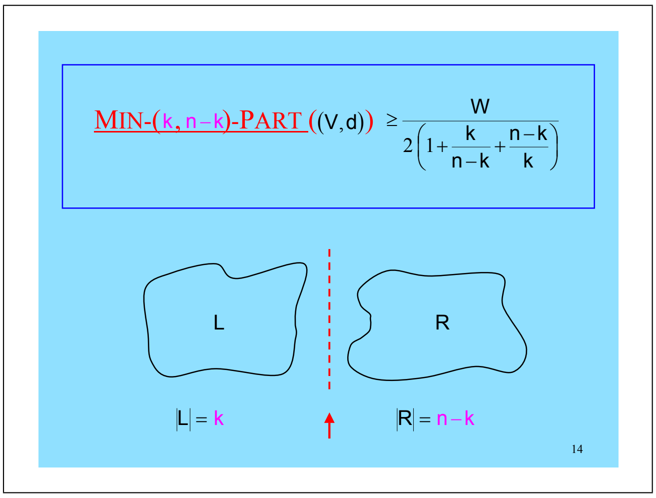
$\beta = \quad d(x_1,x_2)+d(x_3,x_4) \leq \alpha$

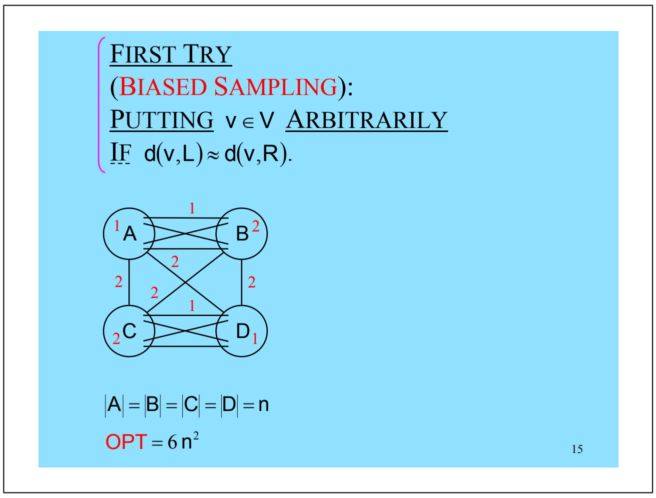
$W_L + W_R \leq 4$ OPT

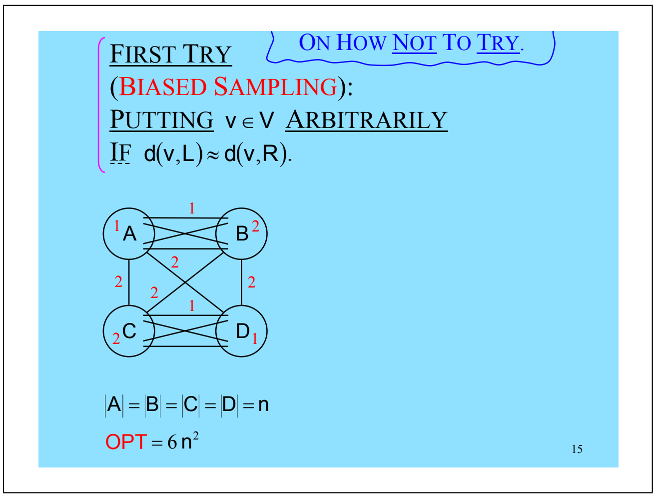
$W \leq 6$ OPT

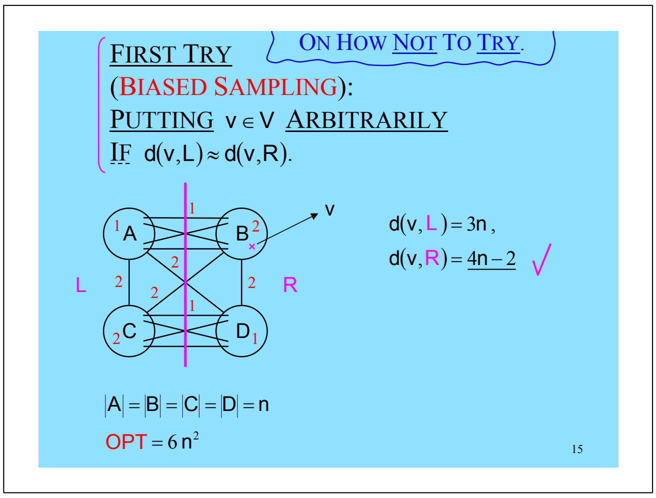
$E[\beta] = \dfrac{4\,(W_L + W_R)}{n^2}$

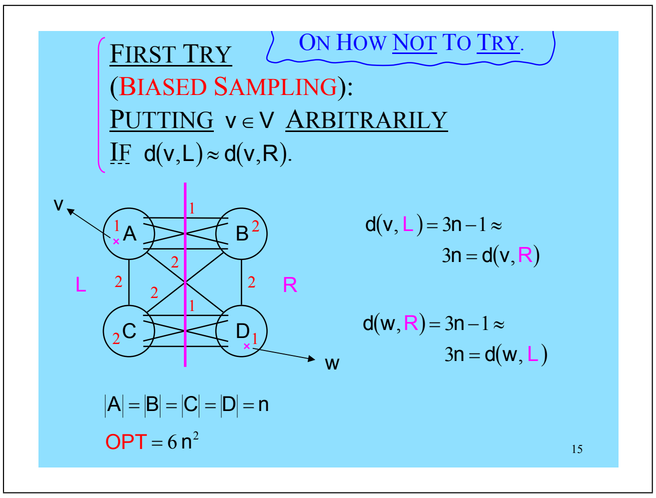
13

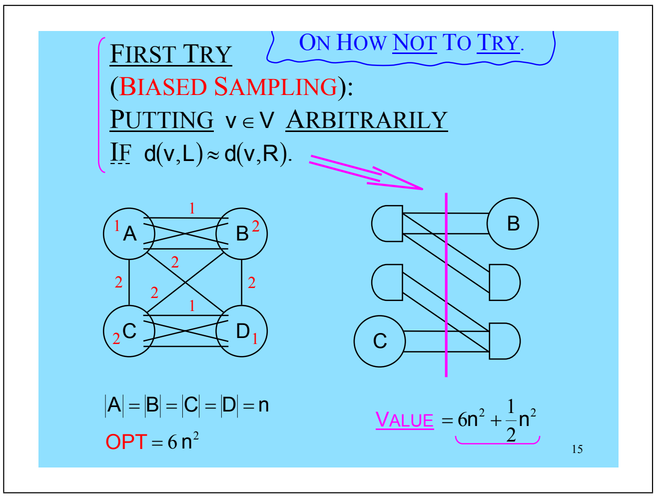

MIN-(k, n−k)-PART ((V,d)) $\geq \frac{W}{2\left(1+\frac{k}{n-k}+\frac{n-k}{k}\right)}$
L
R
|L| = k
|R| = n − k
14


FIRST TRY
(BIASED SAMPLING):
PUTTING v ∈ V ARBITRARILY
IF d(v,L) ≈ d(v,R).
A
B
C
D
|A| = |B| = |C| = |D| = n
OPT = 6 n²
15


FIRST TRY
ON HOW NOT TO TRY.
(BIASED SAMPLING):
PUTTING v ∈ V ARBITRARILY
IF d(v,L) ≈ d(v,R).
A
B
C
D
|A| = |B| = |C| = |D| = n
OPT = 6 n²
15


FIRST TRY
ON HOW NOT TO TRY.
(BIASED SAMPLING):
PUTTING v ∈ V ARBITRARILY
IF d(v,L) ≈ d(v,R).
v
d(v, L) = 3n ,
d(v, R) = 4n − 2
L
R
A
B
C
D
|A| = |B| = |C| = |D| = n
OPT = 6 n²
15


FIRST TRY
ON HOW NOT TO TRY.
(BIASED SAMPLING):
PUTTING v ∈ V ARBITRARILY
IF d(v,L) ≈ d(v,R).
v
d(v, L) = 3n − 1 ≈
3n = d(v, R)
L
R
d(w, R) = 3n − 1 ≈
3n = d(w, L)
w
A
B
C
D
|A| = |B| = |C| = |D| = n
OPT = 6 n²
15


FIRST TRY
ON HOW NOT TO TRY.
(BIASED SAMPLING):
PUTTING v ∈ V ARBITRARILY
IF d(v,L) ≈ d(v,R).
A
B
C
D
B
C
|A| = |B| = |C| = |D| = n
OPT = 6 n²
VALUE = $6n^2 + \frac{1}{2}n^2$
15

IS

METRIC BISECTION

HARD TO APPROX. ?

16

A PTAS FOR GENERAL METRIC BISECTION:

(VIA LINEARIZED QUADRATIC PROGRAMS)

$$P_{(V,d)} = \frac{1}{2}\sum_{i \neq j} w_{ij}\left(x_i\overbrace{(1-x_j)}+x_j\overbrace{(1-x_i)}\right),$$

$$\sum_i x_i = n/2$$

17

A PTAS FOR GENERAL METRIC BISECTION:

(VIA LINEARIZED QUADRATIC PROGRAMS)

$$P_{(V,d)} = \frac{1}{2}\sum_{i \neq j} w_{ij}\left(x_i\overbrace{(1-x_j)}+x_j\overbrace{(1-x_i)}\right),$$

([AKK95], [FK00]) $\sum_i x_i = n/2$

17

A PTAS FOR GENERAL METRIC BISECTION:

(VIA LINEARIZED QUADRATIC PROGRAMS)

$$P_{(V,d)} = \frac{1}{2}\sum_{i \neq j} w_{ij}\left(x_i\overbrace{(1-x_j)}+x_j\overbrace{(1-x_i)}\right),$$

([AKK95], [FK00]) $\sum_i x_i = n/2$

(SMOOTH, BUT UNBOUND. WEIGHT PR.)

17

A PTAS FOR GENERAL METRIC BISECTION:

(VIA LINEARIZED QUADRATIC PROGRAMS) $r_i = \sum_{j \in N(i)} (1-x_j)$

$$P_{(V,d)} = \frac{1}{2}\sum_{i \neq j} w_{ij}\left(x_i\overbrace{(1-x_j)}+x_j\overbrace{(1-x_i)}\right),$$

([AKK95], [FK00]) $\sum_i x_i = n/2$

(SMOOTH, BUT UNBOUND. WEIGHT PR.)

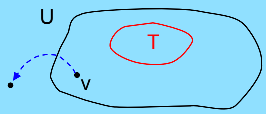
17

BASIC METHOD:

METRIC SAMPLER FOR ESTIMATING

$\underbrace{d(v,U)}$, $\underbrace{U \subseteq V}$,

$\boxed{u \in_R U}$ WITH PR. $= \frac{w_U}{W_U}$,



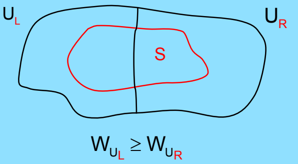
$T = \{U_1, \ldots, U_t\}$

$$\left| d(v,U) - \frac{W_U}{t}\sum_{u \in T}\underbrace{\frac{d(v,u)}{w_U}} \right| \leq \varepsilon d(v,U)$$

WITH PR. $\geq 1 - 2e^{-t\varepsilon^2/8}$.

18

PARTITIONING (U_L, U_R) OF U:



$|S| = 3/\varepsilon^2$

WE HAVE: $|S \cap U_L| \geq 1/\varepsilon^2$

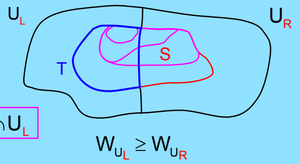
WITH PR. $\geq 1 - \varepsilon$

U_L IS UNKNOWN, SO WE

TRY ALL POSSIBLE SUBSETS T OF S.

19

PARTITIONING (U_L, U_R) OF U:



$\boxed{T = S \cap U_L}$

$|S| = 3/\varepsilon^2$

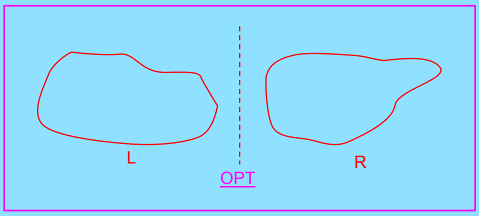
WE HAVE: $|S \cap U_L| \geq 1/\varepsilon^2$

WITH PR. $\geq 1 - \varepsilon$

U_L IS UNKNOWN, SO WE

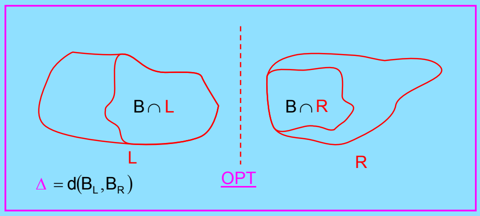
TRY ALL POSSIBLE SUBSETS T OF S.

19



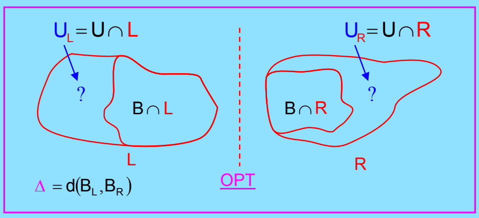
1. CONSTRUCT A SET B OF 'LARGE' WEIGHT VERTICES $v \in V$ WITH $w_v \geq \varepsilon^2 W/100$; $U = V \setminus B$.
2. TAKE A RANDOM SAMPLE S OF SIZE $3/\varepsilon^2$ (OF U) WITH THE BIAS w_u / W_U.
3. GUESS $B_L = B \cap L$, $B_R = B \cap R$.

20



1. CONSTRUCT A SET B OF 'LARGE' WEIGHT VERTICES $v \in V$ WITH $w_v \geq \varepsilon^2 W/100$; $U = V \setminus B$.
2. TAKE A RANDOM SAMPLE S OF SIZE $3/\varepsilon^2$ (OF U) WITH THE BIAS w_u / W_U.
3. GUESS $B_L = B \cap L$, $B_R = B \cap R$.

20



1. CONSTRUCT A SET B OF 'LARGE' WEIGHT VERTICES $v \in V$ WITH $w_v \geq \varepsilon^2 W/100$; $U = V \setminus B$.
2. TAKE A RANDOM SAMPLE S OF SIZE $3/\varepsilon^2$ (OF U) WITH THE BIAS w_u / W_U.
3. GUESS $B_L = B \cap L$, $B_R = B \cap R$.

20

RANDOM SAMPLE S ON U_L :

GUESS $T = S \cap U_L$.

(ASSUME, $W_{U_L} \geq W_{U_R}$).

CONSTRUCT ESTIMATORS:

$$e_v = \frac{\widehat{W}_{U_L}}{t} \sum_{u \in T} \frac{\mathbf{d(v,u)}}{\mathbf{w_u}} + d(v, B_L)$$

21

[RANDOM SAMPLE S ON U_L :

$(|S| = 3/\varepsilon^2)$ ON U.

GUESS $T = S \cap U_L$.

(ASSUME, $W_{U_L} \geq W_{U_R}$).

CONSTRUCT ESTIMATORS:

$$e_v = \frac{\widehat{W}_{U_L}}{t} \sum_{u \in T} \frac{\mathbf{d(v,u)}}{\mathbf{w_u}} + d(v, B_L)$$

→ LINEAR PROGRAM:

4. CONSTRUCT

LP(U)::

$$\text{MIN} \quad \sum_{v \in U} x_v e_v + \sum_{v \in U} (1 - x_v)\, d(v, B_R) + \Delta$$

WITH CONSTRAINTS:

$$\sum_{u \in U} (1 - x_u)\, d(u,v) \leq e_v (1 + \varepsilon)$$

$$\sum_{v \in U} x_v = \frac{n}{2} - |B_L| \quad , \quad 0 \leq x_v, x_u \leq 1$$

RR

COMPUTE OPTIMAL FRACTIONAL SOLUTION (x_v^*), $v \in U$ OF LP(U).

USE RANDOMIZED ROUNDING TO OBTAIN A PARTITION (V_1, V_2) OF U (A SOLUTION (y_v)).

"BALANCE" A PARTITION $(V_1 \cup B_L, V_2 \cup B_R)$ BY MOVING THE SMALLEST WEIGHT VERTICES.

[MAIN STEP IN ANALYSIS OF RR:

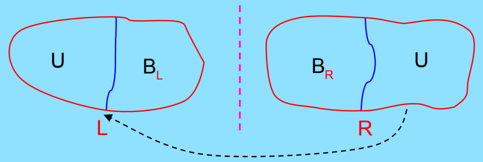
$$E\left[\left|\sum_{v \in U} (x_v^* - y_v)\right|\right] \leq \sqrt{n}/2$$

AND

$\text{OPT} \geq c(x_v^*) - 30\,\varepsilon W$

W.H.P. (AT LEAST $89/100$)

(PROVED BY METRIC SAMPLER ESTIMATIONS.)



MOVING VERTICES

⇒ RESULTING BISECTION

(A,B) $\quad ((y'_v), v \in V)$

CLAIM: (A,B) APPROX. OPT TO WITHIN εW.

RUNNING TIME: $LP(n) \cdot 2^{O(1/\varepsilon^2)}$

COMBINATORIAL VERSION: $n^2 \cdot 2^{O(1/\varepsilon^2)}$

EXTENSION TO METRIC $(n_1, n_2, \ldots, n_K)$ MIN-PARTITIONING

(EXISTENCE OF PTASs)

RUNNING TIME: $LP(n)\cdot 2^{O(1/\varepsilon^2)}$

COMBINATORIAL VERSION: $n^2 \cdot 2^{O(1/\varepsilon^2)}$

EXTENSION TO
METRIC $(n_1,n_2,\dots,n_K)$ MIN-PARTITIONING
(EXISTENCE OF PTASS)

26

EXTENSION TO
QUASIMETRIC PARTITIONING
AND CLUSTERING

($d(x,z) \le C(d(x,y)+d(y,z))$
FOR SOME POSITIVE CONSTANT C)

→ INCLUDES ALL "POWERS" OF
METRIC FUNCTIONS LIKE $\ell_2^2, \ell_p^q, \dots$
USED IN CLUSTERING APPLICATIONS.

27

IF $d(x,y)$ IS
METRIC AND $\alpha > 0$,
THEN $(d(x,y))^\alpha$ IS
A QUASIMETRIC.

28

FURTHER RESEARCH:

1. DESIGN THE PTASS (?)
FOR "BALANCED"
METRIC $(n_1,n_2,\dots,n_K)$ CLUSTERING
(EXTENSION OF
[FKKR03]-TECHNIQUE ?)

2. IMPROVE EFFICIENCY

29

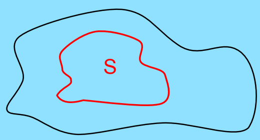
FURTHER RESEARCH:

1. DESIGN THE PTASS (?)
FOR "BALANCED"
METRIC $(n_1,n_2,\dots,n_K)$ CLUSTERING
(EXTENSION OF
[FKKR03]-TECHNIQUE ?)

2. IMPROVE EFFICIENCY
(CONSTANT TIME ALGS ?)

29

RANDOM CORE-SAMPLE COMPLEXITY :



$|S| = \tilde{O}(1/\varepsilon^4)$ IS ENOUGH FOR
MAX-KCSP, K ARBITRARY.
([AFKK02] , [FKKV05])

30

Approximation Algorithms for Facility Location

Jens Vygen

University of Bonn

Outline

Facility Location: Applications

- manufacturing plants
- storage facilities, depots
- warehouses, retail stores
- libraries, fire stations, hospitals
- servers in the internet
- base stations for wireless services
- buffers distributing signals on a chip
- ...

Goal: Optimum service for clients at minimum cost

Common features of facility location problems

- Two sets: clients and potential facilities
  - Each client must be served.
  - A potential facility can be opened or not.
  - Clients can only be served by open facilities.
- Two cost components: facility cost and service cost.
  - Opening a facility involves a certain cost.
  - Serving a client from a facility involves a certain cost.
- The total cost is to be minimized.

But there are many variants

- Can a client's demand be satisfied by more than one facility?
- Are there constraints on the total demand, or total service cost, that a facility can handle?
- Do the service costs satisfy the triangle inequality?
- Are there finitely or infinitely many potential facilities?
- Do the facility costs depend on the total demand served?
- Is it allowed to serve only a subset of clients, and pay for those that are not served?
- Is there a bound on the number of facilities that we can open?
- Does the total service cost of a facility depend on the sum of the distances to its clients, or the length of a shortest tour, or the length of an optimal Steiner tree?
- Are we interested in the sum of all service costs, or rather in the maximum service cost?
- Do we need to serve facilities by second-stage facilities (etc.)?

Example 1: Fermat-Weber Problem

The most prominent example for continuous facility location

Locating a single facility in $\mathbb{R}^n$: Given $a_1, \dots, a_m \in \mathbb{R}^n$ and weights $w_1, \dots, w_m \in \mathbb{R}_+$, find $p \in \mathbb{R}^n$ minimizing

$$\sum_{i=1}^{m} w_i ||p - a_i||.$$

- For ℓ_1-norm solvable in linear time (Blum et al. 1973)
- ℓ_2-norm, $n = 2$, $m = 3$: Simple geometric solution (Fermat, Torricelli, Cavalieri, Simpson, Heinen)
- For ℓ_2-norm: construction by ruler and compasses impossible (Bajaj 1988)
- Approximate solution for ℓ_2-norm: Weiszfeld's algorithm (Weiszfeld 1937, Kuhn 1973, Vardi and Zhang 2001, Rautenbach et al. 2004)

Example 2: Uncapacitated Facility Location (UFL)

The most prominent example for discrete facility location

Instance:

- a finite set $\mathcal{D}$ of clients;
- a finite set $\mathcal{F}$ of potential facilities;
- a fixed cost $f_i \in \mathbb{R}_+$ for opening each facility $i \in \mathcal{F}$;
- a service cost $c_{ij} \in \mathbb{R}_+$ for each $i \in \mathcal{F}$ and $j \in \mathcal{D}$.

We look for:

- a subset S of facilities (called *open*) and
- an assignment $\sigma : \mathcal{D} \to S$ of clients to open facilities,
- such that the sum of facility costs and service costs

$$\sum_{i \in S} f_i + \sum_{j \in \mathcal{D}} c_{\sigma(j)j}$$

is minimum.

More examples discussed later

- Capacitated Facility Location
- Universal Facility Location
- Facility Location and Network Design with Service Capacities

These are more general and more realistic in many applications.

Approximation Algorithms: Definition

Let f be a function assigning a real number to each instance. An f-**approximation algorithm** is an algorithm for which a polynomial p exists such that for each instance I:

- the algorithm terminates after at most $p(size(I))$ steps,
- the algorithm computes a feasible solution, and
- the cost of this solution is at most $f(I)$ times the optimum cost of instance I.

f is called the **approximation ratio** or **performance guarantee**. If f is a constant, we have a **(constant-factor) approximation algorithm**.

Uncapacitated Facility Location is as hard as Set Covering

Set Covering: Given a finite set U, a family $\mathcal{S}$ of subsets of U with $\bigcup_{S \in \mathcal{S}} S = U$, and weights $w : \mathcal{S} \to \mathbb{R}_+$, find a set $\mathcal{R} \subseteq \mathcal{S}$ with $\bigcup_{R \in \mathcal{R}} R = U$ with minimum total weight $\sum_{R \in \mathcal{R}} w(R)$.

- No $o(\log |U|)$-approximation algorithm exists unless $P = NP$. (Raz, Safra 1997)
- Greedy algorithm has performance ratio $1 + \ln |U|$. (Chvátal 1979)
- Set Covering is a special case of Uncapacitated Facility Location: define $\mathcal{D} := U$, $\mathcal{F} := \mathcal{S}$, $f_S = w(S)$ for $S \in \mathcal{S}$, $c_{Sj} := 0$ for $j \in S \in \mathcal{S}$ and $c_{Sj} := \infty$ for $j \in U \setminus S$.
- Conversely, the greedy algorithm for Set Covering can be applied to Uncapacitated Facility Location: Set $U := \mathcal{D}$, $\mathcal{S} = \mathcal{F} \times 2^{\mathcal{D}}$, and $w(i, D) := f_i + \sum_{j \in D} c_{ij}$. (Hochbaum 1982)

A natural assumption: metric service costs

Therefore we assume henceforth **metric service costs**:

$$c_{ij} \geq 0$$

and

$$c_{ij} + c_{i'j} + c_{i'j'} \geq c_{ij'}$$

for all $i, i' \in \mathcal{F}$ and $j, j' \in \mathcal{D}$.

Equivalently, we assume c to be a (semi)metric on $\mathcal{D} \cup \mathcal{F}$.

Motivation:

- The general problem is as hard as Set Covering.
- In many practical problems service costs are proportional to geometric distances, or to travel times, and hence are metric.

But: Greedy algorithm has performance guarantee $\Omega(\log n / \log \log n)$ even for metric instances. (Jain et al. 2003)

Integer Linear Programming Formulation

$$
\begin{array}{llcll}
\text{minimize} & \displaystyle\sum_{i \in \mathcal{F}} f_i y_i + \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{D}} c_{ij} x_{ij} & & & \\
\text{subject to} & & & & \\
& x_{ij} & \leq & y_i & (i \in \mathcal{F}, j \in \mathcal{D}) \\
& \displaystyle\sum_{i \in \mathcal{F}} x_{ij} & = & 1 & (j \in \mathcal{D}) \\
& x_{ij} & \in & \{0,1\} & (i \in \mathcal{F}, j \in \mathcal{D}) \\
& y_i & \in & \{0,1\} & (i \in \mathcal{F})
\end{array}
$$

(Balinski 1965)

Linear Programming Relaxation

$$\begin{array}{llcll} \text{minimize} & \sum_{i\in\mathcal{F}} f_i y_i + \sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{D}} c_{ij}x_{ij} & & & \\ \text{subject to} & & & & \\ & x_{ij} & \leq & y_i & (i\in\mathcal{F}, j\in\mathcal{D}) \\ & \sum_{i\in\mathcal{F}} x_{ij} & = & 1 & (j\in\mathcal{D}) \\ & x_{ij} & \geq & 0 & (i\in\mathcal{F}, j\in\mathcal{D}) \\ & y_i & \geq & 0 & (i\in\mathcal{F}) \end{array}$$

The Dual LP

$$\begin{array}{llcll} \text{maximize} & \sum_{j\in\mathcal{D}} v_j & & & \\ \text{subject to} & & & & \\ & v_j - w_{ij} & \leq & c_{ij} & (i\in\mathcal{F}, j\in\mathcal{D}) \\ & \sum_{j\in\mathcal{D}} w_{ij} & \leq & f_i & (i\in\mathcal{F}) \\ & w_{ij} & \geq & 0 & (i\in\mathcal{F}, j\in\mathcal{D}) \end{array}$$

First Approximation Algorithm: LP Rounding

- Compute an optimum solutions (x^*, y^*) and (v^*, w^*) to the primal and dual LP.
- By complementary slackness, $x^*_{ij} > 0$ implies $v^*_j - w^*_{ij} = c_{ij}$, and thus $c_{ij} \leq v^*_j$.
- Let G be the bipartite graph with vertex set $\mathcal{F} \cup \mathcal{D}$ containing an edge $\{i,j\}$ iff $x^*_{ij} > 0$.
- Assign clients to clusters iteratively as follows.
  - In iteration k, let j_k be a client $j \in \mathcal{D}$ not assigned yet and with v^*_j smallest.
  - Create a new cluster containing j_k and those vertices of G that have distance 2 from j_k and are not assigned yet.
  - Continue until all clients are assigned to clusters.
- For each cluster k we choose a neighbour i_k of j_k with f_{i_k} minimum, open i_k, and assign all clients in this cluster to i_k.

Analysis of the LP Rounding Approximation Algorithm

- The service cost for client j in cluster k is at most
$$c_{i_k j} \;\leq\; c_{ij} + c_{ij_k} + c_{i_k j_k} \;\leq\; v^*_j + 2v^*_{j_k} \;\leq\; 3v^*_j,$$
where i is a common neighbour of j and j_k.
- The facility cost f_{i_k} can be bounded by
$$f_{i_k} \;\leq\; \sum_{i\in\mathcal{F}} x^*_{ij_k} f_i \;=\; \sum_{i\in\mathcal{F}:\{i,j_k\}\in E(G)} x^*_{ij_k} f_i \;\leq\; \sum_{i\in\mathcal{F}:\{i,j_k\}\in E(G)} y^*_i f_i.$$
As j_k and $j_{k'}$ cannot have a common neighbour for $k \neq k'$, the total facility cost is at most $\sum_{i\in\mathcal{F}} y^*_i f_i$.
- The total cost is at most
$$3\sum_{j\in\mathcal{D}} v^*_j + \sum_{i\in\mathcal{F}} y^*_i f_i,$$
which is at most four times the LP value. Hence we get:

Theorem
This is a 4-approximation algorithm for metric UFL.
(Shmoys, Tardos and Aardal 1997)

Better approximation ratios for metric UFL

technique	ratio	RT	authors	year
LP-Rounding	3.16	−	Shmoys, Tardos, Aardal	1997
LP-Rounding+Greedy	2.41	−	Guha, Khuller	1998
LP-Rounding	1.74	−	Chudak	1998
Local Search	5.01	∘	Korupolu, Plaxton, Rajaraman	1998
Primal-Dual	3.00	+	Jain, Vazirani	1999
Primal-Dual+Greedy	1.86	+	Charikar, Guha	1999
LP-Rounding+Primal-Dual+Greedy	1.73	−	Charikar, Guha	1999
Local Search	2.42	∘	Arya et al.	2001
Primal-Dual	1.61	+	Jain, Mahdian, Saberi	2002
LP-Rounding	1.59	−	Sviridenko	2002
Primal-Dual+Greedy	1.52	+	Mahdian, Ye, Zhang	2002

RT : running time; − : slow; ∘ : medium; + : fast

Primal-Dual Algorithm by Jain, Mahdian and Saberi (2002)

Start with $U := \mathcal{D}$ and time $t = 0$. Increase t, maintaining $v_j = t$ for all $j \in U$. Consider the following events:

- $v_j = c_{ij}$, where $j \in U$ and i is not open. Then start to increase w_{ij} at the same rate, in order to maintain $v_j - w_{ij} = c_{ij}$.
- $\sum_{j\in\mathcal{D}} w_{ij} = f_i$. Then open i. For all $j \in \mathcal{D}$ with $w_{ij} > 0$: freeze v_j and set $w_{i'j} := \max\{0, c_{ij} - c_{i'j}\}$ for all $i' \in \mathcal{F}$, and remove j from U.
- $v_j = c_{ij}$, where $j \in U$ and i is open. Then freeze v_j and set $w_{i'j} := \max\{0, c_{ij} - c_{i'j}\}$ for all $i' \in \mathcal{F}$, and remove j from U.

Improvement by Mahdian, Ye and Zhang (2002)

- Multiply all facility costs by 1.504.
- Apply the Jain-Mahdian-Saberi algorithm.
- Now consider the original facility costs.
- Apply greedy augmentation (Charikar, Guha 1999): Let g_i be the service cost saving induced by adding facility i. Iteratively pick an element $i \in \mathcal{F}$ maximizing $\frac{g_i}{f_i}$ as long as this ratio is greater than 1.

Theorem
This is a 1.52-approximation algorithm for metric UFL.

Lower bound on approximation ratios

Theorem
There is no 1.463-factor approximation algorithm for metric UFL unless $P = NP$.

(Sviridenko [unpublished], based on Guha and Khuller [1999] and Feige [1998])

Local Search as a general heuristic

Basic Framework:

- Define a neighbourhood graph on the feasible solutions.
- Start with any feasible solution x.
- If there is a neighbour y of x that is (significantly) better, set $x := y$ and iterate.

Features:

- Quite successful for many practical (hard) problems
- Many variants of local search heuristics
- Typically no guarantees of running time and performance ratio.

Local Search in Combinatorial Optimization

Example: TSP

- Even simple 2-opt typically yields good solutions. Variants (chained Lin-Kernighan) with empirically less than 1% error
- Worst-case running time of k-opt is exponential for all k.
- Performance ratio $\Omega(n^{\frac{1}{2k}})$.

(Applegate et al. 2003, Chandra, Karloff, Tovey 1999)

Example: Facility Location

- Probably the first nontrivial problem where local search led to constant-factor approximation algorithms.
  (Korupolo, Plaxton and Rajamaran 2000, Arya et al. 2004)
- But: for metric UFL worse in theory (maybe also in practice)
- The only known technique to obtain a constant-factor approximation for CAPACITATED FACILITY LOCATION.

Capacitated Facility Location (CFL)

Instance:

- finite sets $\mathcal{D}$ (clients) and $\mathcal{F}$ (potential facilities);
- metric service costs $c_{ij} \in \mathbb{R}_+$ for $i \in \mathcal{F}$ and $j \in \mathcal{D}$;
- an opening cost $f_i \in \mathbb{R}_+$ for each facility $i \in \mathcal{F}$;
- a capacity $u_i \in \mathbb{R}_+$ for each facility $i \in \mathcal{F}$;
- a demand d_j for each client $j \in \mathcal{D}$.

We look for:

- a subset S of facilities (called *open*) and
- an assignment $x : S \times \mathcal{D} \to \mathbb{R}_+$ with $\sum_{i\in S} x_{ij} = d_j$ for $j \in \mathcal{D}$ and $\sum_{j\in\mathcal{D}} x_{ij} \le u_i$ for $i \in S$
- such that the sum of facility costs and service costs

$$\sum_{i\in S}\left(f_i + \sum_{j\in\mathcal{D}} c_{ij}x_{ij}\right)$$

is minimum.

Splittable or Unsplittable Demands

Assume that facilities with given capacities are open.
Task: assign the clients to these facilities, respecting capacity constraints.

- Splittable (or uniform) demand:
  Hitchcock transportation problem.
- Unsplittable non-uniform demand:
  Generalizes bin packing.

Consequence: CFL with unsplittable demands has no approximation algorithm. It is strongly *NP*-hard to distinguish between instances with optimum cost 0 and ∞.

Hence consider splittable demands only.

Universal Facility Location (UniFL)

Instance:

- finite sets $\mathcal{D}$ (clients) and $\mathcal{F}$ (potential facilities);
- metric service costs, i.e. a metric c on $\mathcal{D} \cup \mathcal{F}$;
- a demand $d_j \geq 0$ for each $j \in \mathcal{D}$;
- for each $i \in \mathcal{F}$ a cost function $f_i : \mathbb{R}_+ \to \mathbb{R}_+ \cup \{\infty\}$, left-continuous and non-decreasing.

We look for:

- a function $x : \mathcal{F} \times \mathcal{D} \to \mathbb{R}_+$ with $\sum_{i \in \mathcal{F}} x_{ij} = d_j$ for all $j \in \mathcal{D}$

(a *feasible solution*), such that $c(x) := c_F(x) + c_S(x)$ is minimum, where

$$c_F(x) := \sum_{i \in \mathcal{F}} f_i\left(\sum_{j \in \mathcal{D}} x_{ij}\right) \qquad \text{and} \qquad c_S(x) := \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{D}} c_{ij} x_{ij}.$$

UniFL: Facility cost function given by an oracle

$f_i(z)$: cost to install capacity z at facility i.

Given by an oracle that, for each $i \in \mathcal{F}$, $u, c \in \mathbb{R}_+$ and $t \in \mathbb{R}$, computes $f_i(u)$ and

$$\max\{\delta \in \mathbb{R} : u + \delta \geq 0,\ f_i(u+\delta) - f_i(u) + c|\delta| \leq t\}.$$

Proposition

There always exists an optimum solution.

(Mahdian and Pál 2003)

UniFL: important special cases

- Uncapacitated Facility Location: $d_j = 1$ ($j \in \mathcal{D}$), and $f_i(0) = 0$ and $f_i(z) = t_i$ for some $t_i \in \mathbb{R}_+$ and all $z > 0$ ($i \in \mathcal{F}$).
- Capacitated Facility Location: $f_i(0) = 0$, $f_i(z) = t_i$ for $0 < z \leq u_i$ and $f_i(z) = \infty$ for $z > u_i$, where $u_i, t_i \in \mathbb{R}_+$ ($i \in \mathcal{F}$).
- Soft-Capacitated Facility Location: $d_j = 1$ ($j \in \mathcal{D}$), and $f_i(z) = \lceil \frac{z}{u_i} \rceil t_i$ for some $u_i \in \mathbb{N}$, $t_i \in \mathbb{R}_+$ and all $z \geq 0$ ($i \in \mathcal{F}$).

Simple local search operations

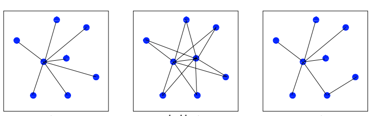
- Add: open a facility (CFL); add capacity to a facility (UniFL).
- Drop: close a facility (CFL).
- Swap: open one facility, close another one (CFL).

Even for CFL with non-uniform demands, these operations do not suffice:
When closing one facility, it may be necessary to open many other ones (and re-assign the demand along the edges of a star).

Previous approximation algorithms for CFL and UniFL

Kuehn, Hamburger 1963	add,drop,swap	CFL	—	
Korupolu, Plaxton, Rajamaran 1998	add,drop,swap	CFL	8.001	uniform capacities
Chudak, Williamson 1999	add,drop,swap	CFL	5.829	uniform capacities
Pál, Tardos, Wexler 2001	add,star	CFL	8.532	
Mahdian, Pál 2003	add,star	UniFL	7.873	
Zhang, Chen, Ye 2004	add,double-star	CFL	5.829	
Garg, Khandekar, Pandit 2005	add,double-star	UniFL	5.829	not polynomial!
Vygen 2005	add,comet	UniFL	6.702	

All based on local search.



star double star comet

Add Operation for UniFL

Let $t \in \mathcal{D}$ and $\delta > 0$. Replace current solution x by an optimum solution y of the transportation problem

$$\min\left\{ c_S(y) \;\middle|\; y : \mathcal{F} \times \mathcal{D} \to \mathbb{R}_+,\ \sum_{i \in \mathcal{F}} y_{ij} = d_j\ (j \in \mathcal{D}),\ \sum_{j \in \mathcal{D}} y_{ij} \leq \sum_{j \in \mathcal{D}} x_{ij}\ (i \in \mathcal{F} \setminus \{t\}),\ \sum_{j \in \mathcal{D}} y_{tj} \leq \sum_{j \in \mathcal{D}} x_{tj} + \delta \right\}.$$

We denote by

$$c^x(t, \delta) := c_S(y) - c_S(x) + f_t\left(\sum_{j \in \mathcal{D}} x_{tj} + \delta\right) - f_t\left(\sum_{j \in \mathcal{D}} x_{tj}\right)$$

the estimated cost (which is at least $c(y) - c(x)$).

How to find a profitable ADD operation

Lemma

Let $\epsilon > 0$ and $t \in \mathcal{F}$. Let x be a feasible solution. Then there is an algorithm with running time $O(|V|^3 \log |V| \epsilon^{-1})$ that

- *finds a $\delta \in \mathbb{R}_+$ with $c^x(t, \delta) \le -\epsilon c(x)$*
- *or decides that no $\delta \in \mathbb{R}_+$ exists for which $c^x(t, \delta) \le -2\epsilon c(x)$.*

(Mahdian, Pál 2003)

PIVOT Operation

Let x be a feasible solution. Let A be a graph with $V(A) = \mathcal{F}$ and

$$\delta \in \Delta_A^x := \left\{ \delta \in \mathbb{R}^{\mathcal{F}} \,\middle|\, \sum_{j \in \mathcal{D}} x_{ij} + \delta_i \ge 0 \text{ for all } i \in \mathcal{F}, \sum_{i \in \mathcal{F}} \delta_i = 0 \right\}.$$

Then we consider the operation $\mathrm{PIVOT}(A, \delta)$, which means:

- Compute a minimum-cost (w.r.t. c) uncapacitated δ-flow in (A, c).
- W.l.o.g., the edges carrying flow form a forest.
- Scan these edges in topological order, reassigning clients according to flow values.
- This increases the cost of the solution by at most the cost of the flow plus

$$\sum_{i \in \mathcal{F}} f_i \left(\sum_{j \in \mathcal{D}} x_{ij} + \delta_i \right) - f_i \left(\sum_{j \in \mathcal{D}} x_{ij} \right).$$

How to find a profitable PIVOT operation

But: how to choose δ?

- δ cannot be chosen almost optimally for the complete graph (unless $P = NP$).
- We show how to choose δ almost optimally if A is a forest.

Restrict attention to PIVOT on arborescences

Let A be an arborescence with $V(A) = \mathcal{F}$. Let x be a feasible solution.
For $\delta \in \Delta_A^x$ define

$$c_{A,i}^x(\delta) := f_i \left(\sum_{j \in \mathcal{D}} x_{ij} + \delta_i \right) - f_i \left(\sum_{j \in \mathcal{D}} x_{ij} \right) + \left| \sum_{j \in A_i^+} \delta_j \right| c_{ip(i)}$$

for $i \in \mathcal{F}$ and

$$c^x(A, \delta) := \sum_{i \in \mathcal{F}} c_{A,i}^x(\delta).$$

Here A_i^+ denotes the set of vertices reachable from i in A, and $p(i)$ is the predecessor of i.

How to find a profitable PIVOT for an arborescence

Lemma

Let $\epsilon > 0$. There is an algorithm with running time $O(|\mathcal{F}|^4 \epsilon^{-3})$ that

- *finds a $\delta \in \Delta_A^x$ with $c^x(A, \delta) \le -\epsilon c(x)$*
- *or decides that no $\delta \in \Delta_A^x$ exists for which $c^x(A, \delta) \le -2\epsilon c(x)$.*

(Vygen 2005)

Bounding the cost of a local optimum

Let $0 < \epsilon < 1$. Let x, x^* be feasible solutions to a given instance.

Lemma

If $c^x(t, \delta) \ge -\frac{\epsilon}{|\mathcal{F}|} c(x)$ for all $t \in \mathcal{F}$ and $\delta \in \mathbb{R}_+$, then

$$c_S(x) \le c_F(x^*) + c_S(x^*) + \epsilon c(x).$$

(Pál, Tardos and Wexler 2001)

Lemma

If $c^x(A, \delta) \ge -\frac{\epsilon}{|\mathcal{F}|} c(x)$ for all stars and comets A and $\delta \in \Delta_A^x$, then

$$c_F(x) \le 4 c_F(x^*) + 2 c_S(x^*) + 2 c_S(x) + \epsilon c(x).$$

(Vygen 2005)

The total cost of a local optimum

These two lemmata imply:

Theorem
If $c^x(t,\delta) > -\frac{\epsilon}{8|\mathcal{F}|}c(x)$ for $t \in \mathcal{F}$ and $\delta \in \mathbb{R}_+$ and $c^x(A,\delta) > -\frac{\epsilon}{8|\mathcal{F}|}c(x)$ for all stars and comets A and $\delta \in \Delta_A^x$, then

$$c(x) \leq (1+\epsilon)(7c_F(x^*) + 5c_S(x^*)).$$

By scaling facility costs by $\frac{\sqrt{41}-5}{2}$ we get a polynomial-time $(\frac{\sqrt{41}+7}{2}+\epsilon)$-approximation algorithm for UniFL.

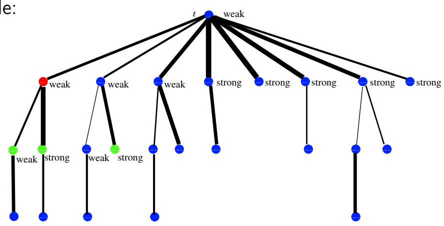
How to bound the facility cost

Let x be the current solution and x^* be an optimum solution.
Let $b(i) := \sum_{j\in\mathcal{D}}(x_{ij} - x^*_{ij})$ $(i \in \mathcal{F})$.
Let y be an optimum transshipment from $S := \{i \in \mathcal{F} : b(i) > 0\}$ to $T := \{i \in \mathcal{F} : b(i) < 0\}$.
W.l.o.g., the edges where y is positive form a forest F.
The cost of y is at most $c_S(x^*) + c_S(x)$.

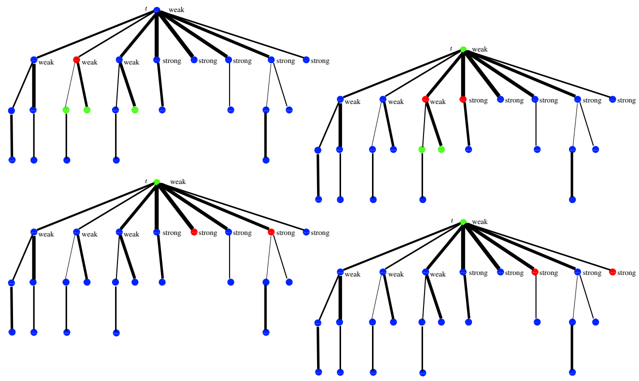
Using F and y, we will define a set of pivot operations on stars and comets, whose total estimated cost is at most $4c_F(x^*) - c_F(x) + 2c_S(x^*) + 2c_S(x)$.
An operation (A,δ) **closes** $s \in S$ if $\delta_s = -b(s) < 0$, and it **opens** $t \in T$ if $0 < \delta_t \leq -b(t)$.
Over all operations to be defined, we will close each $s \in S$ once, open each $t \in T$ at most four times, and use an estimated routing cost at most twice the cost of y.

How to define the operations (1)

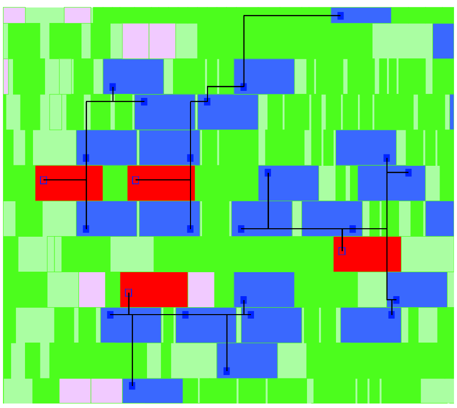
Orient F as a set of arborescences rooted at elements of T.
Call a vertex **weak** if there is more flow on downward than on upward incident arcs, otherwise **strong**. Let $t \in T$.
Open t up to twice if t is strong and up to three times if t is weak.
For each child s of t: Close s once, and open each child of s at most once (if weak) or twice (if strong).
Example:



How to define the operations (2)



VLSI Design: Distributing a signal to several terminals

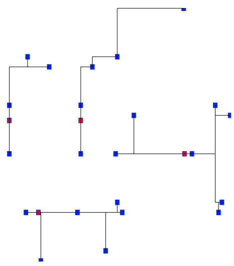


blue: terminals red: facilities

Problem Statement

Instance:

- metric space (V,c),
- finite set $\mathcal{D} \subseteq V$ (terminals/clients),
- demands $d : \mathcal{D} \to \mathbb{R}_+$,
- facility opening cost $f \in \mathbb{R}_+$,
- capacity $u \in \mathbb{R}_+$.



Find a partition $\mathcal{D} = D_1 \dot\cup \cdots \dot\cup D_k$ and Steiner trees T_i for D_i $(i = 1,\ldots,k)$ with

$$c(E(T_i)) + d(D_i) \leq u$$

for $i = 1,\ldots,k$ such that

$$\sum_{i=1}^{k} c(E(T_i)) + kf$$

is minimum.

Complexity Results

(All the following results are by Maßberg and Vygen 2005)

Proposition

- *There is no $(1.5 - \epsilon)$-approximation algorithm (for any $\epsilon > 0$) unless $P = NP$.*
- *There is no $(2 - \epsilon)$-approximation algorithm (for any $\epsilon > 0$) for any class of metrics where the Steiner tree problem cannot be solved exactly in polynomial time.*
- *There is a 2-approximation algorithm for geometric instances (similar to Arora's approximation scheme for the TSP). However, this is not practically efficient.*

Lower bound: spanning forests

Let F_1 be a minimum spanning tree for $(\mathcal{D}, c)$.
Let $e_1, \ldots, e_{n-1}$ be the edges of F_1 so that $c(e_1) \geq \ldots \geq c(e_{n-1})$.
Set $F_k := F_{k-1} \setminus \{e_{k-1}\}$ for $k = 2, \ldots, n$.

Lemma

F_k is a minimum weight spanning forest in $(\mathcal{D}, c)$ with exactly k components.

Proof.

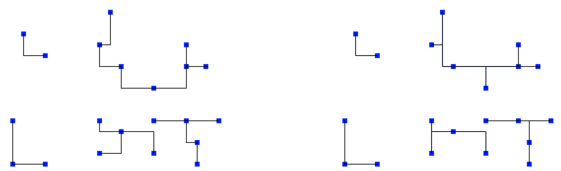
By induction on k. Trivial for $k = 1$. Let $k > 1$.
Let F^* be a minimum weight k-spanning forest.
Let $e \in F_{k-1}$ such that $F^* \cup \{e\}$ is a forest. Then

$$c(F_k) + c(e_{k-1}) = c(F_{k-1}) \leq c(F^*) + c(e) \leq c(F^*) + c(e_{k-1}).$$

□

Lower bound: Steiner forests

A k-Steiner forest is a forest F with $\mathcal{D} \subseteq V(F)$ and exactly k components.



Lemma

$\frac{1}{\alpha}c(F_k)$ is a lower bound for the cost of a minimum weight k-Steiner forest, where α is the Steiner ratio.

Lower bound: number of facilities

Let t' be the smallest integer such that

$$\frac{1}{\alpha}c(F_{t'}) + d(\mathcal{D}) \leq t' \cdot u$$

Lemma

t' is a lower bound for the number of facilities of any solution.

Let t'' be an integer in $\{t', \ldots, n\}$ minimizing

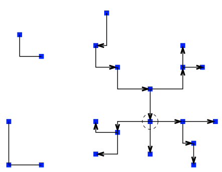
$$\frac{1}{\alpha}c(F_{t''}) + t'' \cdot f.$$

Theorem

$\frac{1}{\alpha}c(F_{t''}) + t'' \cdot f$ is a lower bound for the cost of an optimal solution.

Algorithm A

1. Compute a minimum spanning tree on $(\mathcal{D}, c)$.
2. Compute t'' and spanning forest $F_{t''}$ as above.
3. Split up overloaded components by a bin packing approach.



It can be guaranteed that for each new component at least $\frac{u}{2}$ of load will be removed from the initial forest.

Analysis of Algorithm A

Recall: $\frac{1}{\alpha}c(F_{t''}) + t'' \cdot f$ is a lower bound for the optimum.

We set $L_r := \frac{1}{\alpha}c(F_{t''})$ and $L_f := t'' \cdot f$.

Observe: $L_r + d(\mathcal{D}) \leq \frac{u}{f} L_f$.

The cost of the final solution is at most

$$c(F_{t''}) + t''f + \frac{2}{u}\Big(c(F_{t''}) + d(\mathcal{D})\Big)f$$

$$= \alpha L_r + L_f + \frac{2f}{u}(\alpha L_r + d(\mathcal{D}))$$

$$\leq \alpha L_r + L_f + 2\alpha L_f$$

Theorem

Algorithm A is a $(2\alpha + 1)$-approximation algorithm.

Algorithm B

Define metric c' by $c'(v,w) := \min\{c(v,w), \frac{uf}{u+2f}\}$.

1. Compute a Steiner tree F for $\mathcal{D}$ in (V,c') with some β-approximation algorithm.
2. Remove all edges e of F with $c(e) \geq \frac{uf}{u+2f}$.
3. Split up overloaded components of the remaining forest as in algorithm A.

Theorem
Algorithm B has perfomance ratio 3β.

Using the Robins-Zelikovsky Steiner tree approximation algorithm we get a 4.648-approximation algorithm.

With a more careful analysis of the Robins-Zelikovsky algorithm we can get a 4.099-approximation algorithm in $O(n^{2^{10000}})$ time.

Algorithm C

Define metric c'' by $c''(v,w) := \min\{c(v,w), \frac{uf}{u+f}\}$

1. Compute a tour F for $\mathcal{D}$ in (V,c'') with some γ-approximation algorithm.
2. Remove the longest edge of F.
3. Remove all edges e of F with $c(e) \geq \frac{uf}{u+f}$.
4. Split up overloaded components of the remaining forest as in algorithm A.

Theorem
Algorithm C has perfomance ratio 3γ.

Using Christofides' TSP approximation algorithm we get a 4.5-approximation algorithm in $O(n^3)$ time.

Comparison of the three approximation algorithms

- Algorithm A computes a minimum spanning tree.
- Algorithm B calls the Robins-Zelikovsky algorithm.
- Algorithm C calls Christofides' algorithm.
- Then each algorithm deletes expensive edges and splits up overloaded components.

algorithm	metric	perf.guar.	runtime
A	$(\mathbb{R}^2, \ell_1)$	4	$O(n \log n)$
A	general	5	$O(n^2)$
B	general	4.099	$O(n^{2^{10000}})$
C	general	4.5	$O(n^3)$

Experimental Results

Algorithm A on six real-world instances:

	inst1	inst2	inst3	inst4	inst5	inst6
# terminals	3675	17140	45606	54831	109224	119461
MST length	13.72	60.35	134.24	183.37	260.36	314.48
t'	117	638	1475	2051	3116	3998
L_r	8.21	31.68	63.73	102.80	135.32	181.45
$L_r + L_f$	23.07	112.70	251.06	363.28	531.05	689.19
# facilities	161	947	2171	2922	4156	5525
service cost	12.08	54.23	101.57	159.93	234.34	279.93
total cost	32.52	174.50	377.29	531.03	762.15	981.61
gap (factor)	1.41	1.55	1.59	1.46	1.44	1.42

Reduction of power consumption

Algorithm A on four chips, compared to the previously used heuristic:

chip	Jens	Katrin	Bert	Alex
technology	180nm	130nm	130nm	130nm
# clocktrees	1	3	69	195
total # sinks	3805	137265	40298	189341
largest instance	375	119461	16260	35305
power (W, old)	0.100	0.329	0.306	2.097
power (W, new)	0.088	0.287	0.283	1.946
difference	−11.1%	−12.8%	−7.5%	−7.2%

Some Open Problems

- Close the gap between 1.46 and 1.52 for the approximability of UNCAPACITATED FACILITY LOCATION.
- Find better lower bounds than 1.46 for capacitated problems (such as CFL).
- Is Universal Facility Location really harder than CFL?
- Improve the approximation ratio for the problem with service capacities (in $(\mathbb{R}^2, \ell_1)$, with a practically efficient algorithm).
- In some real-world instances, there exists an interval graph on the terminals, and we have to partition this graph into cliques. Is there an approximation algorithm for the resulting problem?
- What other interesting problems combining facility location with network design, or routing, can be approximated?
- What about multi-stage extensions?

Further Reading

- J. Vygen. Approximation Algorithms for Facility Location Problems (lecture notes, with complete proofs and references). Can be downloaded at `http://www.or.uni-bonn.de/~vygen`
- B. Korte, J. Vygen. Combinatorial Optimization: Theory and Algorithms (Chapter 22). Springer, Berlin, third edition 2006. Also available in Japanese!
- J. Maßberg, J. Vygen. Approximation Algorithms for Network Design and Facility Location with Service Capacities. Proceedings of the 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2005); LNCS 3624, Springer, Berlin 2005, pp. 158–169

Algorithms for a Networked World

Magnús M. Halldórsson

Dept. of Computer Science, University of Iceland

Abstract:

The realization of Moore's law has ensured that computing ability has increased dramatically in our times. The law has held not only for processor power and quantity of internal and external memory, but also for the ability to communicate information. The resulting impact on essentially all spheres of society and life has been nothing less than astounding.

Along with increased connectivity, we are also seeing the introduction of a wide range of compact computing entities, possibly mobile and often non-statically connected into largely wireless networks. The explosion of the web and the internet as not only a source of information but also as a resource of computational intelligence, is poised to lead to a dramatic change in the way we view computation. The traditional view of an algorithm with full random access to its input, operating serially on a single processor, is on the retreat.

In comparison, it can be said that changes in CS theory are less dramatic. Surely, each year and each conference brings new topics, new subjects, new treatments, and new directions. Yet, we can also easily detect a great deal of consistency [one that is certainly comforting at times], and a measured pace of change. Are we theoreticians then by nature reactionaries? One of the theses of this talk is that the objects of study in CS theory are inherently fundamental and long-lasting, applying also to this Panopticon world of global and ubiquitous computing.

Yet, we cannot rest on our laurels, with self-satisfied smugness. We must find ways to treat the new means, ways, possibilities and limitations of computation in a systematic framework that continues to provide applied fields with rigorous guidance. The aim of the talk is to discuss some objectives, measures, and paradigms that address the changing nature of computing in a networked world. The concrete examples discussed will mostly relate to problems of coloring and packing, the topics of main focus of the speakers research.

This will by no means be a comprehensive overview – in fact, it is unlikely to be even a balanced introduction. Instead, the hope is that by posing some questions, some members of the audience will eventually be prompted to find some of the answers.

Algorithms for Sequence Manipulation and Related Problems

D. T. Lee

Institute of Information Science
Academia Sinica
&
Department of Computer Science and Information Engineering
National Taiwan University

2006 NHC Spring School Workshop March 2-3, 2006

Combinatorial Optimization

- The combinatorial optimization typically deals with problems of *maximizing* or *minimizing* a function of one or many variables subject to a number of inequality constraints.
- Consider two categories of problems on sequences:
  - Optimization Problems
  - Range Search Problems

Optimization on Sequences

- Given a sequence $A = a_1, a_2, \ldots, a_n$, an optimization problem on sequences is to *maximize* or *minimize* some function, such as: *sum of subsequence*, *density of subsequence*, etc., with some constraints, such as: length, weight, etc.

Range Search Problems

- A *range* (*query*) *search problem* is typically to *report* the subset S' to *count* the total number of elements of the subset S' of a set S contained in a query range Q subject to certain conditions.

Range Search on Sequences

- Given a sequence $A = a_1, a_2, \ldots, a_n$, and a range query, we want to *report* or *count* some subsequences of A contained in the query range satisfying certain conditions.

5/92

Focus of Talk

- Focus on the optimization and range search problems on sequences, give a survey and present efficient problem-solving techniques for them.
- Many elegant and sometimes sophisticated techniques have been developed and applied to these problems in the past decade.
- Key: To exploit the combinatorial nature of these problems to obtain simpler and faster algorithms or algorithms with faster query time for range search problems.

6/92

Notations 2

- Let A be a **sequence** of n real numbers $a_1, a_2, \ldots, a_n$.
- **segment** of A(i, j) is a *consecutive* subsequence $a_i, a_{i+1}, \ldots, a_j$
- **width** w(i, j) of A(i, j): $j - i + 1$
- **density** d(i, j) of A(i, j):
  $(a_i + a_{i+1} + \ldots + a_j)/w(i, j)$
- **sum** s(i, j) of A(i, j): $a_i + a_{i+1} + \ldots + a_j$.

7/92

Notations 1

- Given two positive real numbers ℓ and u with $\ell \le u$, we say $A(i, j) = a_i, a_{i+1}, \ldots, a_j$ of A is **feasible** if $\ell \le w(i, j) = j-i+1 \le u$.
- **rank** r(x, S) of an element x in a set S: the number of elements in S no greater than x, i.e., $r(x, S) = |\{y \mid y \in S, y \le x\}|$.

8/92

Optimization Problems on Sequences

- Sum of subsequence,
- Density of subsequence
- Selection of subsequence
- subject to constraints on length, or weight of subsequences.

Max-Sum Segment Problem

- **Input:**
  - a sequence A of n real numbers $a_1, a_2, \ldots, a_n$
  - two nonnegative real numbers ℓ, u with $\ell \leq u$.
- **Output:**
  - a segment A(i*, j*) with maximum sum over all $O((u-\ell)\, n)$ feasible segments such that s(i*, j*) = max{ s(i, j) | A(i, j) is feasible.}

Example – Max-Sum Segment

- **Input:**
  - A = 3, -4, -2, 5, 6
  - $\ell = 2$, $u = 4$
- **Output: s(4, 5) = 11**
- Sums of all feasible segments:
  s(1, 2) = -1, s(1, 3) = -3, s(1, 4) = 2
  s(2, 3) = -6, s(2, 4) = -1, s(2, 5) = 5
  s(3, 4) = 3, s(3, 5) = 9
  s(4, 5) = 11

Application: Max-Sum Segment

- Finding the most rich GC-rich region in a DNA sequence can be cast as a maximum-sum segment problem.
- Input sequence A corresponds to a given DNA sequence, where $a_i = 1$ if the corresponding nucleotide in the DNA sequence is G or C, $a_i = 0$ otherwise.
- Output feasible segment corresponds to the GC-rich region of the given DNA sequence.

Applications in Bioinformatics

- Useful applications in bioinformatics including
  - finding tandem repeats, which are commonly used to map disease genes
  - locating DNA segments with rich CG content is a key step in gene finding and promoter prediction
  - low complexity filter, which is most commonly applied in sequence database search.

13/92

Max-Sum Segment Problem

- **Prior Results:**
  - O(n) time for the special case $\ell = 1$, $u = n$ Gries [Science of Computer Programming'82]
  - O(n) time for the special case $\ell = 1$, $u = n$ Bentely. [Commun. ACM '84]
  - O(n) time. Lin, Jiang and Chao. [Journal of Computer and System Sciences '02]
  - O(n) time. Fan, Lee, Lu, Tsou, Wang, and Yao. [CIAA '03]

14/92

Technique Used – Lin *et al.*

- Lin, Jiang and Chao gave an O(n) time algorithm based on a clever technique called *left-negative decomposition.*

  J. of Computer and System Sciences '02

15/92

Lin *et al.* Algorithm ['02]

- A sequence A= $a_1, a_2, \ldots, a_n$ is ***left-negative*** iff the sum of each proper prefix $a_1, a_2, \ldots, a_i$ is negative or zero for all $1 \le i \le n-1$.
- A partition of the sequence A= $A_1A_2 \ldots A_k$ is ***minimal left-negative*** if each A_i, $1 \le i \le k$, is l*eft-negative*, and, for each $1 \le i \le k-1$, the sum of A_i is positive.

16/92

Example – left-negative sequence

- The sequence -4, 1, -2, 3 is left-negative.
- The sequence -5, 3, 4, -1, 2, -6 is not left-negative.
- The partition (-5, 3, 4), (-1, 2), (-6) is minimal left-negative.
- For every suffix of a sequence we can find a minimal left-negative partition.
- (3)(4)(-1,2)(-6); (4)(-1,2)(-6); (-1,2)(-6), (2)(-6); (-6) are all possible minimal left-negative partitions

17/92

Example – left-negative sequence

- Lemma: Every sequence of real numbers can be uniquely partitioned in linear time into blocks of minimal left-negative segments, and the right end of a maximum sum segment must be at a block boundary.
- (-5, (3), (4)), (-1, (2)), (-6)
- Prefix sum sequence S= 0, -5, -2, 2, 1, 3, -3
- Sum(1st) = 2, Sum(2nd) = 1, Sum(3rd) = -6
- Max-Sum segment is a_2, a_3 a_4 a_5 of sum s_5 - s_1 = 8

18/92

Max-Density Segment Problem

- **Input:**
  - a sequence A of n real numbers $a_1, a_2, \ldots, a_n$
  - two nonnegative real numbers ℓ, u with $\ell \leq u$.
- **Output:**
  - a segment A(i*, j*) with maximum density over all O((u-ℓ) n) feasible segments such that d(i*, j*) = max{ d(i, j) | A(i, j) is feasible.}

19/92

Example – Max-Density Segment

- **Input:**
  - A = 3, -4, -2, 5, 6
  - ℓ = 2, u = 4
- **Output: d(4, 5) = 11/2**

- Densities of all feasible segments:
  d(1, 2) = -1/2, d(1, 3) = -1, d(1, 4) = 1/2
  d(2, 3) = -3, d(2, 4) = -1/3, d(2, 5) = 5/4
  d(3, 4) = 3/2, d(3, 5) = 3
  d(4, 5) = 11/2

20/92

Application: Max-Density Segment

- Finding the segment with the largest GC-ratio in a DNA sequence can be cast as a maximum-density segment problem.
- Input sequence A corresponds to the given DNA sequence, where $a_i = 1$ if the corresponding nucleotide G or C; and $a_i = 0$ otherwise.
- Output feasible segment corresponds to the region with the largest GC-ratio.

21/92

Application in Bioinformatics

- Maximum-Density Segment Problem arises from the investigation of non-uniformity of nucleotide composition within genomic sequences, which was first revealed through thermal melting and gradient centrifugation experiments.
- Researchers observed that the compositional heterogeneity is highly correlated to the **GC content** of the genomic sequences, and this motivates finding the segment with the largest GC-ratio.

22/92

Max-Density Segment Problem

- **Prior Results:**
  - $O(n\ell)$ time algorithm for the special case $u = n$ Huang. [Computer Appl. in the Biosciences '94]
  - $O(n \log \ell)$ time algorithm for the special case $u = n$ Lin, Jiang, and Chao. [J. Comp. and Syst. Sci. '02]
  - $O(n \log (u-\ell))$ time algorithm. Goldwasser, Kao, and Lu. [J. of Comp. and Syst. Sci. '03]
  - $O(n)$ time algorithm. Kim. [IPL '03]– has a flaw
  - $O(n)$ time algorithm. Chung and Lu. [SICOMP '04]

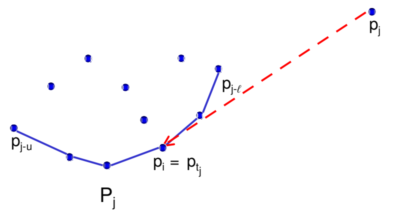
23/92

Technique Used – Kim. IPL '03 3

- Construct a point set in the plane $P = \{p_k \mid p_k = (k, s_k), k = 1, 2, \ldots, n\}$, where $s_k = a_1+a_2+\ldots+a_k$ is the prefix sum of sequence A.
- Construct lower hull of $P_j = \{p_{j-\ell}, p_{j-\ell+1}, \ldots, p_{j-u}\}$ and find tangent segment t_j from p_j to P_j.
- The tangent segment of the maximum slope is the maximum-density segment of A.

24/92

Technique-- Kim. IPL '03 2



$$\frac{p_j - p_i}{j-i} = \frac{a_j + a_{j-1} + \ldots + a_i}{j-i} = d(i+1,\ j)$$

25/92

Technique– Chung & Lu

- Based on a clever observation that d(x, y), d(y+1, z) and d(x, z) with $x \le y < z$ are related as follows:

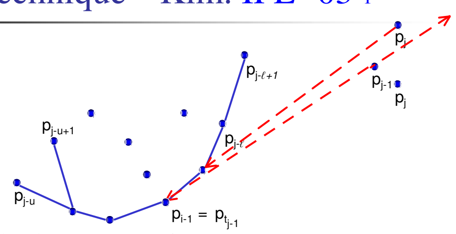
  $d(x, y) \le d(y+1, z)$

  $\Leftrightarrow d(x, y) \le d(x, z)$

  $\Leftrightarrow d(x, z) \le d(y+1, z)$
- Chung and Lu, SIAM Journal on Computing '04.

27/92

Technique-- Kim. IPL '03 1



For each j,

(1) we need to do tangent query from p_j

If p_j lower than the line $p_{t_{j-1}}p_{j-1}$ we do nothing

If p_j higher than the line $p_{t_{j-1}}p_{j-1}$ we do tangent query from p_j

(2) we then delete p_{j-u} from P_j and insert $p_{j-\ell+1}$ into P_j to obtain P_{j+1}

26/92

k-Maximum Sums Problem

- **Input:**
  - a sequence A of n real numbers $a_1, a_2, \ldots, a_n$
  - two nonnegative real numbers ℓ, u with $\ell \le u$
  - a positive integer k.
- **Output:**
  - k feasible segments such that their sums are the k largest over all $O((u-\ell)\ n)$ feasible segments.

28/92

Example – k-Max. Sums

- **Input:**
  - A = 3, -4, -2, 5, 6
  - $\ell = 2$, $u = 4$
  - k = 3
- **Output: s(4, 5) = 11, s(3, 5) = 9 , s(2, 5) = 5**

- Sums of all feasible segments:
  s(1, 2) = -1, s(1, 3) = -3, s(1, 4) = 2
  s(2, 3) = -6, s(2, 4) = -1, s(2, 5) = 5
  s(3, 4) = 3, s(3, 5) = 9
  s(4, 5) = 11

29/92

k-Maximum Sums Problem

- **Prior Results:** (special case $\ell = 1$, $u = n$)
  - O(kn) time algorithm for the Bae and Takaoka. [Int'l Symp. on Parallel Architectures, Algorithms and Networks '04]
  - O(min {k + n $\log^2$n, n$k^{1/2}$}) time algorithm Bengtsson and Chen. ISAAC 2004
  - O(n + k log(min{n, k})) time algorithm Cheng, Chen, Tien, and Chao. ISAAC 2005

30/92

Technique Used – Cheng *et al.*

- Prune-and-Search technique
  - There are n subsequences A(1, n), A(2, n), …, A(n, n) of A, where A(i, n) = a_i, a_{i+1}, …, a_n
  - At iteration i, it prunes away a fraction (½) of the n/2^{i-1} subsequences, so that no more than 2^ik/n segments are in the set of feasible k-maximum sum segments.
- Cheng, Chen, Tien, and Chao. ISAAC 2005.

31/92

Random Sampling Technique

- **Recent Result:**
  - We obtain an O(n log (u - ℓ) + h) time randomized algorithm based on random sampling technique. Lin and Lee. ISAAC 2005
- Solve the k^{th}-Sum Selection Problem using random sampling technique.
- After finding the k^{th} largest feasible segment s*, we can find k largest sum segments by sum range search algorithm, introduced later.

32/92

k-Maximum Densities Problem

- **Input:**
  - a sequence A of n real numbers $a_1, a_2, \ldots, a_n$
  - two nonnegative real numbers ℓ, u with $\ell \leq u$
  - a positive integer k.
- **Output:**
  - the k segments such that their densities are the k largest over all O((u-ℓ) n) feasible segments.

33/92

Example – k-Max Densities

- **Input:**
  - A = 3, -4, -2, 5, 6
  - ℓ = 2, u = 4
  - k = 3
- **Output: d(4, 5) = 11/2, d(3, 5) = 3, d(2, 5) = 5/4**
- Densities of all feasible segments:
  d(1, 2) = -1/2, d(1, 3) = -1, d(1, 4) = 1/2
  d(2, 3) = -3, d(2, 4) = -1/3, d(2, 5) = 5/4
  d(3, 4) = 3/2, d(3, 5) = 3
  d(4, 5) = 11/2

34/92

Random Sampling Technique

- **Recent Result:**
  - We obtain an O(n log (u - ℓ) + h) time randomized algorithm based on random sampling technique.
- Solve the k^{th}-Density Selection Problem using random sampling technique.
- After finding the k^{th} largest density d*, we can find k largest density segments by density range search algorithm, introduced later.

35/92

k^{th}-Sum Selection Problem

- **Input:**
  - a sequence A of n real numbers $a_1, a_2, \ldots, a_n$
  - two nonnegative real numbers ℓ, u with $\ell \leq u$
  - a positive integer k.
- **Output:**
  - the segment A(i*, j*) over all O((u-ℓ) n) feasible segments such that the rank of the sum s(i*, j*) in the set of sums of possible feasible segments is k.

36/92

k^{th}-Sum Selection Problem

- **Prior Result:**
  - O(n log n) time randomized algorithm for the special case $\ell = 1$, $u = n$ based on random sampling technique by Lin and Lee. ISAAC 2005
- **New results:**
  - We obtain an O(n log (u - ℓ)) time randomized algorithm based on random sampling technique.

Technique – Random Sampling

- Contract initial interval $[s_l, s_r] = (-\infty, \infty)$ into a smaller subinterval $[s_l', s_r']$ such that it contains the k^{th} largest feasible segment s* and the subinterval $[s_l', s_r']$ contains at most $O(n^2/n^{1/2}) = O(n^{3/2})$ feasible segments.
- Contract the interval $[s_l', s_r']$ into a smaller subinterval $[s_l'', s_r'']$ such that it contains not only s* but also at most $O(n^{3/2}/n^{1/2}) = O(n)$ feasible segments.
- If both steps are successful, output all the segments in $[s_l'', s_r'']$ and find the solution segment with an appropriate rank, whose sum is s*, by using any *standard selection* algorithm.

k^{th}-Density Selection Problem

- **Input:**
  - a sequence A of n real numbers $a_1, a_2, \ldots, a_n$
  - two nonnegative real numbers ℓ, u with $\ell \le u$
  - a positive integer k.
- **Output:**
  - the segment A(i*, j*) over all O((u-ℓ) n) feasible segments such that the rank of the density d(i*, j*) in the set of densities of possible feasible segments is k.

Sum Finding Problem

- **Input:**
  - a sequence A of n real numbers $a_1, a_2, \ldots, a_n$
  - two nonnegative real numbers ℓ, u with $\ell \le u$
  - a real number s*
- **Output:**
  - the segment over all O((u-ℓ) n) feasible segments such that its sum s(i*, j*) is closest to s*

Example – Sum Finding

- **Input:**
  - A = 3, -4, -2, 5, 6
  - $\ell = 2$, $u = 4$
  - s* = 1.5
- **Output: s(1, 4) = 2**
- Sums of all feasible segments:
  s(1, 2) = -1, s(1, 3) = -3, s(1, 4) = 2
  s(2, 3) = -6, s(2, 4) = -1, s(2, 5) = 5
  s(3, 4) = 3, s(3, 5) = 9
  s(4, 5) = 11

Open

41/92

Density Finding Problem

- **Input:**
  - a sequence A of n real numbers $a_1, a_2, \ldots, a_n$
  - two nonnegative real numbers ℓ, u with $\ell \leq u$
  - a real number d*
- **Output:**
  - the segment over all $O((u-\ell)\ n)$ feasible segments such that its density d(i*, j*) is closest to d*

42/92

Example – Density Finding

- **Input:**
  - A = 3, -4, -2, 5, 6
  - $\ell = 2$, $u = 4$
  - d* = 1.5
- **Output: d(3, 4) = 3/2**
- Densities of all feasible segments:
  d(1, 2) = -1/2, d(1, 3) = -1, d(1, 4) = 1/2
  d(2, 3) = -3, d(2, 4) = -1/3, d(2, 5) = 5/4
  d(3, 4) = 3/2, d(3, 5) = 3
  d(4, 5) = 11/2

43/92

Density Finding Problem

- **Result to date:**
  - We obtain an $O(n \log^2 (u-\ell))$ time and $O(n \log (u-\ell))$ space algorithm based on maintaining a data structure of the left branch of an upper hull.

44/92

Technique – maintaining a data structure of left branch of upper hull $_3$

- We first reduce this problem to a geometric slope finding problem
- Input:
  - Given a set of points $P = \{p_0, p_1, \ldots, p_n\}$ in R^2, and
  - two integers ℓ, u
- Output:
  - a feasible line segment s(i, j) that minimizes the slope m(i, j), where $m(i, j) \geq 0$.

45/92

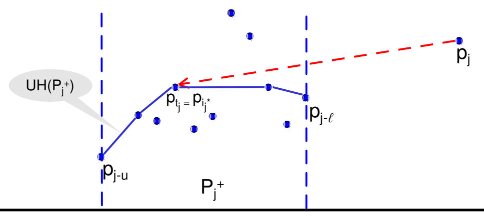
Technique – maintaining a data structure of left branch of upper hull $_2$

- Let $x_i = a_1 + \ldots + a_i$, i=1, 2, …, n, and let $x_0=0$.
- Let $P = \{x_0, x_1, \ldots, x_n\}$.
- Let $P_j = \{x_{j-u}, x_{j-u+1}, \ldots, x_{j-\ell}\}$.
- Let $P_j^+ = \{p_i \in P_j \mid m(i, j) \geq 0\}$.
- Let p_{t_j} be the tangent point of the upper hull $UH(P_j^+)$ from p_j.
- Lemma: p_{t_j} must be a hull point on the left branch of the upper hull $UH(P_j^+)$.

46/92

Technique – maintaining a data structure of left branch of upper hull $_1$

- $m(t_j, j) = \min\{m(i, j) \mid p_i \in P_j^+\} = m(i_j^*, j)$.
- Find $m(t_j, j)$ for each j and take the minimum.



47/92

Generalization to Higher Dimen.

- We can generalize the optimization problems on sequences to higher dimensional space.
- Given an m×n matrix, the *maximum subarray problem* is to find a rectangular subarray with the largest possible sum among all such subarrays.
- For example, the 2D maximum-sum segment problem with $\ell = 1$, $u = n$, referred to as *maximum sum subarray* problem, has many applications in graphics and data mining.

48/92

Range Search Problems

- Single-shot mode query: O(n) query time is optimal for this query mode.
- Repetitive mode query: Preprocessing allowed, and obtaining an o(n) time query time is the goal. A trade-off between storage and query time is expected.

49/92

Sum Range Search Problem single-shot mode

- **Input:**
  - a sequence A of n real numbers $a_1, a_2, \ldots, a_n$
  - two nonnegative real numbers ℓ, u with $\ell \leq u$
  - two real numbers s_l, s_r with $s_l \leq s_r$.
- **Output:**
  - the segments over all O((u-ℓ) n) feasible segments such that their sums are between s_l and s_r.

50/92

Sum Range Search Problem single-shot mode

- **Related work :**
  - O(n log (u-ℓ) + h) time algorithm, where h is the output size. Lin and Lee. ISAAC 2005
  - Straightforward
    - Ordinary binary search trees (order-statistic tree)

51/92

Technique – maintaining an order-statistic tree $_2$

- **prefix sum** x_i of the sequence A, $x_i = a_1 + \ldots + a_i$, i=1, 2, …, n, and let $x_0 = 0$.
- Let P = $\{x_0, x_1, \ldots, x_n\}$.
- Let $P_j = \{x_{j-u}, x_{j-u+1}, \ldots, x_{j-\ell}\}$.
- Maintain an **order-statistic tree** T(P_j) on P_j. by scanning the sequence of prefix sums.

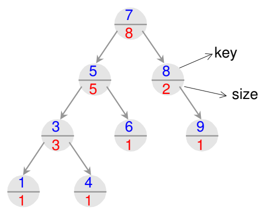
52/92

Order-Statistic Tree

- An order-statistic tree is a balanced binary search tree with size information, size[z], stored in each node z of the tree, and containing the total number of nodes in the subtree rooted at z.
- For an internal node z, size[z] = size[left[z]] + size[right[z]] + 1
- For a leaf node z, size[z]= 1.

53/92

Order-Statistic Tree: Example



54/92

Technique – maintaining an order-statistic tree $_1$

- To solve this problem, it suffices to iterate on each j finding all $x_i \in P_j$ such that $s_l \le x_j - x_i \le s_r$.
- At each iteration j,
  - maintain $T(P_j)$ dynamically such that we can find all the numbers $x_i \in [x_j - s_r, x_j - s_l]$ by binary search in $O(\log (u\text{-}\ell) + h_j)$ time
  - delete $x_{j\text{-}u}$ and insert $x_{j-\ell+1}$ into $T(P_j)$ to obtain $T(P_{j+1})$ in $O(\log (u\text{-}\ell)$) time.

55/92

Basic Lemmas

- The rank $r(x, P_j) = |\{y \mid y \in P_j, y \le x\}|$ of any x not necessarily in P_j can be determined using $T(P_j)$ in O(log n) time.
- An element in $T(P_j)$ of a given rank can be found in O(log n) time.
- Both insertion and deletion operations in $T(P_j)$ can be done in O(log n) time.

56/92

Density Range Search Problem single-shot mode

- **Input:**
  - a sequence A of n real numbers $a_1, a_2, \ldots, a_n$
  - two nonnegative real numbers ℓ, u with $\ell \le u$
  - two real numbers d_l, d_r with $d_l \le d_r$.
- **Output:**
  - the segments over all $O((u-\ell)\,n)$ feasible segments such that their densities are between d_l and d_r

57/92

Density Range Search Problem single-shot mode

- **Recent result:**
  - We obtain an $O(n \log (u-\ell) + h)$ time algorithm, where h is the output size.
  - Use *problem transformation*
  - Maintain a ***priority search tree*** $T(P_j)$ on P_j.

58/92

Problem Transformation

- Transform the **Density Range** Search Problem into a geometric slope range query problem as follows.
- Consider the point set $P = \{p_0, p_2, ..., p_n\}$ in R^2 based on the prefix sums of the sequence A, where $p_i = (x_i, y_i) = (i, a_1 + a_2 + \ldots + a_i)$, i=1, 2, ..., n and $p_0 = (0, 0)$.
- Slope of segment connecting p_i and p_j corresponds to density d(i+1,j)

59/92

Geometric **Slope Range** Search

- **Input:**
  - A point set $P = \{p_0, p_2, ..., p_n\}$ in R^2,
  - two integers ℓ, u
  - two real numbers d_l, d_r,
- **Output:**
  - find all feasible line segments s(i, j) such that their densities m(i, j) satisfying $d_l \le m(i, j) \le d_r$.

60/92

Point-Line Duality Transform

- We can further transform this *geometric **slope range** search problem* into its dual problem, by mapping a *point* $p_i = (x_i, y_i)$ into its *dual line* ℓ_i: $y = x_i x - y_i$.
- For any two points p_i, and p_j, their corresponding dual lines ℓ_i, ℓ_j will intersect at the x-coordinate $x_{ij} = (y_j - y_i)/(x_j - x_i)$ which equals m(i, j).

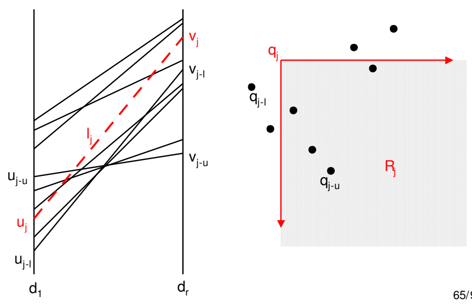
Dual of **Geometric Slope** Range Search– Intersection Search

- **Input:**
  - a set of lines $L = \{\ell_0, \ell_1, ..., \ell_n\}$ in R^2, where each line ℓ_i: $y = x_i x - y_i$,
  - two integers ℓ, u
  - two real numbers d_l, d_r,
- **Output:**
  - find all intersection points $p_{ij}=(x_{ij}, y_{ij})$ of two feasible *dual lines* ℓ_i, ℓ_j such that their abscissae x_{ij} lie in $[d_l, d_r]$.

Intercept to Point Transform

- Let $u_k = x_k d_l - y_k$ denote the *y*-intercept of line ℓ_k at d_l and $v_k = x_k d_r - y_k$ .the *y*-intercept of line ℓ_k at d_r
- Denote $Q = \{q_0, q_1, ..., q_n\}$, where $q_k = (u_k, v_k) = (x_k d_l - y_k, x_k d_r - y_k)$.
- Let $Q_j = \{q_{j-u}, q_{j-u+1}, ..., q_{j-l}\} = Q_{j-u, j-l}$

- By the monotonically increasing property of the slopes of lines in L, i.e., the slope of ℓ_j is larger than each $\ell_i \in L_j$ for $i < j$, any dual line ℓ_i in L_j intersects ℓ_j in $[d_1, d_r]$ iff $u_i \geq u_j$ and $v_i \leq v_j$.



Algorithm for Intersection Search

- The *Intersection Search* problem is now equivalent to an orthogonal range search of the form $R_j = [u_j, \infty) \times (-\infty, v_j]$ to report all the points of Q_j which lie in R_j for each j = 1, 2, …, n.
- We use a data structure called *priority search tree* to support the above orthogonal range queries in logarithmic time.

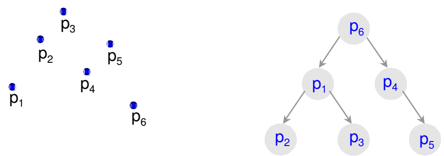
Priority Search Tree - Definition

- A *priority search tree* for a set of ordered pairs $\{(x_i, y_i) \mid i = 1, 2, \ldots, n\}$ is a hybrid of a heap (say in y) and a balanced binary search tree (say in x) used for orthogonal range query where at least one of sides of the query range is unbounded.

Priority Search Tree - Construction

- The priority search tree $T(\{P_j\})$ can be constructed in O(n log n) time using any balanced binary search tree as follows:
  - For a given set S of points P_j, if S is empty, then the priority search tree T(S) is null.
  - Otherwise, let point P_i be the point in S with the smallest y-coordinate and x_m be the median of the x-coordinates of the remaining point set $S\backslash\{P_i\}$. Let $S_l = \{ P_k (x_k, y_k) \in S\backslash\{P_i\} \mid x_k \le x_m \}$ and $S_r = \{ P_k (x_k, y_k) \in S\backslash\{P_i\} \mid x_k > x_m \}$.
  - The root of T(S) contains P_j (and key x_m) and its left child is $T(S_l)$ and right child is $T(S_r)$.

Priority Search Tree: Example



69/92

Sum Range Search Problem repetitive mode -- definition

- **Input:**
  - a sequence A of n real numbers $a_1, a_2, \ldots, a_n$.
  - two nonnegative real numbers ℓ, u with $\ell \le u$
  - two real numbers s_l, s_r with $s_l \le s_r$
- **Online Query:**
  - for an intervals $[s_l, s_r]$, reports the segments over all $O((u-\ell)\ n)$ feasible segments such that their sums are between s_l and s_r.

70/92

Sum Range Search Problem repetitive mode

- **Work to Do:**
  - Try to preprocess A into an appropriate data structure such that the query time is $o(n + h)$.

71/92

Density Range Search Problem repetitive mode -- definition

- **Input:**
  - a sequence A of n real numbers $a_1, a_2, \ldots, a_n$.
  - two nonnegative real numbers ℓ, u with $\ell \le u$
  - two real numbers d_l, d_r with $d_l \le d_r$.
- **Online Query:**
  - for an intervals $[d_l, d_r]$, reports the segments over all $O((u-\ell)\ n)$ feasible segments such that their densities are between d_l and d_r.

72/92

Density Range Search Problem repetitive mode

- **Work to Do:**
  - Try to preprocess A into a nice appropriate structure such that the query time is $o(n + h)$.

Aggregate Range Search Problem -- definition

- Given S and a range query Q, compute an *aggregate function* on the subset $S'=S \cap Q$ An aggregation function can be Min, Max, Sum, Count, Mean, Median (of S'), etc.

Range **Minimum Search** Problem -- definition

- **Input:**
  - a sequence A of n real numbers $a_1, a_2, \ldots, a_{n.}$
  - two real numbers i, j, $i \leq j$.
- **Online Query:**
  - for each query interval [i, j], reports the index k with $i \leq k \leq j$ such that a_k achieves **minimum**.

Range **Minimum Search** Problem -- example

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9
Sequence A	3	8	5	34	7	19	10	12	13	16

$RMS_A(3, 7) = 4$

$RMS_A(1, 6) = 2$

Range **Minimum Search** Problem

- **Related work :**
  - O(n) preprocessing time and O(1) query time under the unit-cost RAM model. -- Gabow, Bentley, and Tarjan. STOC 1984
  - O(n) preprocessing time and O(1) query time under the unit-cost RAM model. -- Bender and Colton. In Proc. the 4th Latin American Symposium on Theoretical Informatics 2000

Technique Used

- ±1RMS problem: adjacent elements of the input sequence A differ by +1 or -1
- Least Common Ancestor (LCA) problem reduces to ±1RMS problem by depth first search traversal of input tree T of LCA.
- An (O(n log n), O(1))-time table-lookup algorithm for RMS
- Using the above algorithm on a smaller array A' obtained by partitioning A into 2n/logn blocks, each of size (log n)/2 and making use of the ±1 property, we can solve ±1RMS in (O(n), O(1))-time.
- RMS reduces to LCA building the Cartesian tree of A.

Problem Reduction

- LCA problem reduces to ±1RMS problem
- Observation: The LCA of nodes u and v in T is the lowest node encountered between the visits to u and to v during a depth first traversal of T, where the depths of the nodes in T differ by exactly one.

(O(n log n), O(1))-time Algorithm for RMS

- Table-lookup algorithm
- Pre-compute the n by log n matrix M[i, j], $1 \le i \le n$, $0 \le j \le \log n$
  $M[i, j] = RMS(i, i + 2^j)$
- For any RMS(i, j) query:
  - Let $k = \max \{r | 2^r < j-i+1\}$
  - $RMS(i, j) = \min\{M(i, k), M(j-2^k+1, k)\}$

Problem Reduction

- *RMS can be reduced to LCA* by building a Cartesian tree of A.
- The root of a Cartesian tree is the minimum element of the array. The root is labeled with the position *k* of this minimum element.
- The left and right children of the root are the roots of recursively constructed Cartesian trees of the left and right subarrays respectively.

Range **Maximum-Sum Segment Search** Problem -- definition

- **Input:**
  - a sequence A of n real numbers $a_1, a_2, \ldots, a_n$ to be preprocessed.
  - real numbers i, j, $i \le j$ and k, *l*, $k \le l$.
- **Online Query:**
  - for two query intervals [i, j] and [k, *l*], reports A(x, y) with $i \le x \le j$ and $k \le y \le l$ that maximizes s(x, y).

Range **Maximum-Sum Segment Search** Problem

- **Related work :**
  - O(n) preprocessing time and O(1) query time under unit-cost RAM model by Chen and Chao. ISAAC 2004

Technique Used

- Let $S = s_1, s_2, \ldots, s_n$ be the sequence where $s_k = a_1+a_2+\ldots+a_k$ is the prefix sum of sequence A.
- Disjoint case: by min= RMinS(S, i, j), max= RMaxS(S, k, l), Ans.= $s_{max} - s_{min}$
- Overlapping case: Divide into 3 possible cases and take minimum of the outputs of these three cases.
  - [i, k] and [k, l]: by RMinS(S, i, k), RMaxS(S, k, l)
  - [k, j] and [j, l]: by RMinS(S, k, j), RMaxA(S, j, l)
  - [k, j] and [k, j]: by RMSSS(k, j), a special case!

Range **Maximum-Sum Segment Search** Problem- A Special Case

- **Input:**
  - a sequence A of n real numbers $a_1, a_2, \ldots, a_n$.
  - two real numbers i, j, $i \le j$.
- **Online Query:**
  - for any query interval [i, j], reports A(x, y) with $i \le x \le y \le j$ such that A(x, y) is the maximum-sum segment of A(i, j).

Range **Maximum-Sum Segment Search** Problem- A Special Case

- **Related work :**
  - O(n) preprocessing time and O(1) query time under the unit-cost RAM model by Chen and Chao. ISAAC 2004
- They solved RMSSS by using RMinS and RMaxS.

Algorithm

- Let $C[\cdot]$ be the array of prefix sum of A.
- Define: left bound L[k] of A at index k to be the largest index *l* with $1 \le l \le k-1$ such that $C[l] \ge C[k]$, and L[k]=0, if no such *l* exists.
- Define: partner P[k] of A at index k to be the largest index p with $L[k]+1 \le p \le k$ that minimizes C[p-1].
- A(P[k], k) is a candidate segment of A at index k with sum M[k] = s (P[k], k), for $1 \le k \le n$.

RMax-SumSegSearch

- Algorithm of RMSSS(A, i, j)

  r $\leftarrow$ RMaxS(M, i, j), i.e. M[r] is maximum
  if P[r] < i **then**
      p $\leftarrow$ RMinS(C, i-1, r-1)+1
      s $\leftarrow$ RMaxS(C, i-1, r-1)+1
      if C[r]-C[p-1] < M(s)
      then output (P[s], s)
      else output (p, r)
  else output (P[r], r)

Range **Maximum-Density Segment Search** Problem

- **Input:**
  - a sequence A of n real numbers $a_1, a_2, \ldots, a_n$
- **Online Query:**
  - for two intervals [i, j] and [k, l], reports A(x, y) with $i \leq x \leq j$ and $k \leq y \leq l$ that maximizes d(x, y).

89/92

Range **Maximum-Density Segment Search** Problem- A Special Case

- **Input:**
  - a sequence A of n real numbers $a_1, a_2, \ldots, a_n$.
  - two real numbers i, j, $i \leq j$.
- **Online Query:**
  - for any query interval [i, j], reports A(x, y) with $i \leq x \leq y \leq j$ such that A(x, y) is the maximum-density segment of A(i, j).

90/92

Generalization

- We can also generalize the above aggregation range search problems to other aggregation functions or consider the dynamic range query problems which support insertion, deletion, concatenation operations.

91/92

Conclusion

- We considered optimization and range search problems on sequences, gave a survey and presented recent results for these problems.
- Open problems
  - **Sum Range** Search Problem (repetitive mode)
  - **Density Range** Search Problem (repetitive mode)
  - Range **Maximum-Density Segment Search** Problem
  - Problems for which the elements are each chosen from an (error) interval

92/92

Dynamic Data Structures in Computational Geometry

Timothy M. Chan
School of Computer Science
University of Waterloo

Every student in computer science knows that we can maintain a set of n real numbers (in 1-d) to support insertions and deletions in $O(\log n)$ time and searches in $O(\log n)$ time, by his/her favorite kind of balanced search trees. The underlying question in this talk is: to what extent can this basic result hold in dimension beyond one? In other words, for what problems in computational geometry can we maintain solutions efficiently under insertions and deletions of data?

In the first part of the talk, we give a rough introduction to this fundamental and important topic called "dynamic computational geometry", which has been extensively studied for over three decades. We obviously do not have time to give a comprehensive survey (despite what the title says), but the purpose is to provide some examples to illustrate what types of problems have been studied, what types of results are possible, and what general techniques are available. Our focus is on fully dynamic algorithms that have guaranteed time bounds for arbitrary update sequences. We mention the following results:

- For *orthogonal range searching* or 2-d *point location*, dynamization with polylogarithmic update time is relatively easy, by directly applying standard balancing tricks to geometric search trees (such as range trees or segment trees); for these problems, research is thus all about removing logarithmic factors [16, 14, 10, 15]. The well-known dynamic 2-d *convex hull* problem [17, 5, 3] also fits in this category.

- Simple dynamization strategies (the "logarithmic method" and the "square-root method") are applicable to a general class of *decomposable search* problems [2].

- For some non-decomposable problems, like 2-d *smallest enclosing circle* and 3-d *linear programming*, dynamization can still be obtained by reducing them to decomposable problems through parametric search [13].

- A general technique by Eppstein [12] can solve dynamic max-max- or min-min-type problems like *bichromatic closest pair*. A key idea here is to handle deletions by re-insertions.

- For some other non-decomposable problems, like 2-d *width* [7] and *rectangle connectivity* [6], polylogarithmic dynamization is not known, but sublinear update time is possible.

- For even more challenging problems (like 2-d *smallest enclosing rectangle*), min-max-type problems (like 2-d *discrete 1-center*), and non-convex optimization problems (like 2-d *largest empty circle*), nothing is known at all, except in special cases (such as the insertion-only and offline settings) [8].

In the second part of the talk, we focus specifically on the dynamic 2-d *post office* problem, and discuss in more detail a new data structure recently discovered by this speaker [9], which is the first polylogarithmic result known. The problem is to maintain a set of points in the Euclidean plane, to support exact nearest neighbor queries. (By a standard lifting transformation, it can also be viewed as a dynamic 3-d convex hull problem.) The previous solution by Agarwal and Matoušek [1] more than a decade ago has $O(n^{\varepsilon})$ amortized update time and $O(\log n)$ query time, or $O(\log^2 n)$ amortized update time and $O(n^{\varepsilon})$ query time, for any fixed $\varepsilon > 0$. The new solution has $O(\log^3 n)$ amortized insertion time, $O(\log^6 n)$ amortized deletion time, and $O(\log^2 n)$ query time. The techniques involved include removing "high-degree" elements (following Agarwal and Matoušek [1]), handling deletions by re-insertions (as in Eppstein [12]), and using geometric tools like "sampling" and "conflict lists" (as in [11, 4, 18]).

We conclude by mentioning applications of dynamic data structures in the design of geometric algorithms, and by listing some open problems in the area.

References

[1] P. K. Agarwal and J. Matoušek. Dynamic half-space range reporting and its applications. *Algorithmica*, 13:325–345, 1995. (Also in FOCS 1992.)

[2] J. Bentley and J. Saxe. Decomposable searching problems I: static-to-dynamic transformation. *J. Algorithms*, 1:301–358, 1980.

[3] G. S. Brodal and R. Jacob. Dynamic planar convex hull. In FOCS 2002.

[4] T. M. Chan. Random sampling, halfspace range reporting, and construction of ($\leq k$)-levels in three dimensions. *SIAM J. Comput.*, 30:561–575, 2001. (Also in FOCS 1998.)

[5] T. M. Chan. Dynamic planar convex hull operations in near-logarithmic amortized time. *J. ACM*, 48:1–12, 2001. (Also in FOCS 1999.)

[6] T. M. Chan. Dynamic subgraph connectivity with geometric applications. In STOC 2002.

[7] T. M. Chan. A fully dynamic algorithm for planar width. *Discrete Comput. Geom.*, 30:17–24, 2003. (Also in SoCG 2001.)

[8] T. M. Chan. Semi-online maintenance of geometric optima and measures. *SIAM J. Comput.*, 32:700–716, 2003. (Also in SODA 2002.)

[9] T. M. Chan. Dynamic data structures for 3-d convex hulls and 2-d nearest neighbor queries. In SODA 2006.

[10] Y.-J. Chiang and R. Tamassia. Dynamic algorithms in computational geometry. *Proc. IEEE*, 80:1412–1434, 1992.

[11] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom*, 4:387–421, 1989.

[12] D. Eppstein. Dynamic Euclidean minimum spanning trees and extrema of binary functions. *Discrete Comput. Geom.*, 13:111–122, 1995. (Also in FOCS 1992.)

[13] J. Matoušek. Linear optimization queries. *J. Algorithms*, 14:432–448, 1993.

[14] K. Mehlhorn. *Data Structures and Algorithms 3: Multi-Dimensional Searching and Computational Geometry.* Springer-Verlag, 1984.

[15] C. W. Mortensen. Fully-dynamic two dimensional orthogonal range and line segment intersection reporting in logarithmic time. In SODA 2003.

[16] M. H. Overmars. *The Design of Dynamic Data Structures.* Springer-Verlag, 1983.

[17] M. H. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *J. Comput. Sys. Sci.*, 23:166–204, 1981. (Also in STOC 1980.)

[18] E. Ramos. On range reporting, ray shooting, and k-level construction. In SoCG 1999.

Dynamic Data Structures in Computational Geometry

Timothy M. Chan
School of CS
U of Waterloo

Theme

Dynamic data structures for searching in 1D:

balanced search trees

What about dynamic data structures in 2D (& beyond) ???

"Dynamic Computational Geometry"

Solve geometric problems under
insertion of an object &
deletion " " "
w. guaranteed worst-case/amortized time

Settings:

- fully dynamic ⇐
- insert-only
- delete-only
- offline (all updates known)
- semi-online (delete times known)
- random updates
- kinetic

This Talk

Part I:
Survey

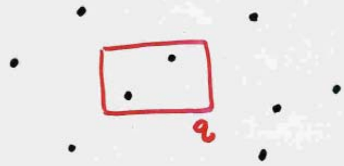
Part II:
A new result for dynamic 3D convex hull/ 2D post office [SODA'06]

PART I:

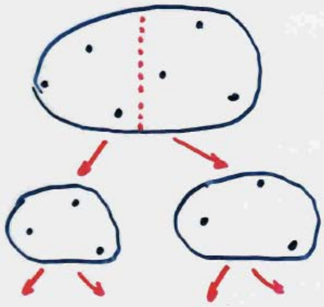
SURVEY

Ex 1: 2D Orthogonal Range Search

Maintain n pts in 2D s.t.
given rectangle q,
report/count all pts in q



General idea: just dynamize a geometric search tree, here, a "range tree"



node stores y-coords in a 1D tree

(balance by weight)

$\Rightarrow$ $O(\log^2 n)$ update & query

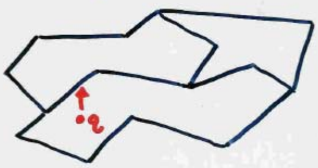
Various improvements:

$O(\log n \log\log n)$ update & query [Mehlhorn, Näher '86]

$O(\log n)$ update & query [Mortensen, SODA '03]

Ex 2: 2D Point Location

Maintain n disjoint line segments s.t.
given pt q,
find segment immediately above q



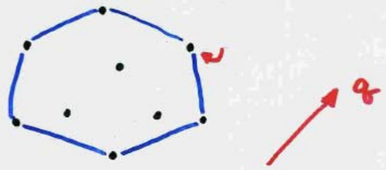
Similar idea: dynamize a "segment tree"

$\Rightarrow$ $O(\log^2 n)$ update & query [Overmars '85]

Various improvements on the logs ...

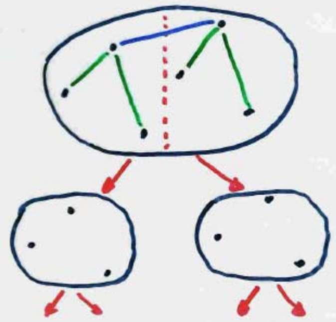
Ex 3 : 2D Convex Hull

Maintain n pts in 2D s.t.
given direction q,
find vertex of CH extreme along q



Rmk: CH may change a lot!

Similar idea: dynamize a "hull tree"



$\Rightarrow$ $O(\log^2 n)$ update, $O(\log n)$ query
[Overmars, van Leeuwen, STOC'80]

Various improvements:
$O(\log^{1+\varepsilon} n)$ update, $O(\log n)$ query
[Chan, FOCS'99]
$O(\log n)$ update & query
[Brodal, Jacob, FOCS'02]

Ex 4 : 2D Post Office Problem

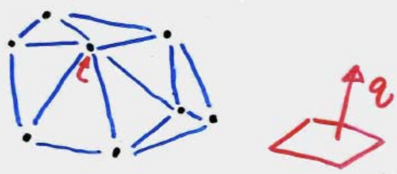
Maintain n pts in 2D s.t.
given pt q, find nearest neighbor to q

i.e. given (x,y), find
$\min_i \, (x-a_i)^2 + (y-b_i)^2$
i.e. $\min_i \, -2a_i x - 2b_i y + a_i^2 + b_i^2$

Rmk: viewing $(-2a_i, -2b_i, a_i^2+b_i^2)$ as a pt in 3D, this reduces to ...

Ex 5 : 3D Convex Hull

Maintain n pts in 3D s.t.
given direction q,
find vertex of CH extreme along q



Static sol'n: [classical CG]
$O(n \log n)$ preproc, $O(\log n)$ query

Rmk: but CH may change a lot!

Simple general idea for insert-only:

decompose into $O(\log n)$ subsets

sizes distinct powers of 2

$\Rightarrow$ $O(\log^2 n)$ insert & query

[Bentley, Saxe '80: the "logarithmic method" for "decomposable search problems"]

Simple general idea for fully dynamic:

decompose into $O(\sqrt{n})$ subsets

$\Rightarrow$ $O(\sqrt{n}\log n)$ insert/delete & query

A different idea for 3D CH:

dynamize a slower static sol'n, namely, a "partition tree" OR "cutting tree" [Matoušek '91]

$\Rightarrow$ $O(\log^2 n)$ update, $O(n^{\varepsilon})$ query

OR $O(n^{\varepsilon})$ update, $O(\log n)$ query

[Agarwal, Matoušek, FOCS '92]

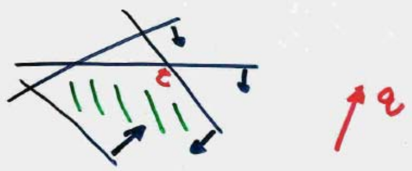
A new result for 3D CH:

$O(\text{polylog } n)$ update & query

[wait for 2^{nd} half of talk!]

Ex 6: 3D Linear Programming

Maintain n halfspaces in 3D s.t. given direction q, find vertex in intersection extreme along q

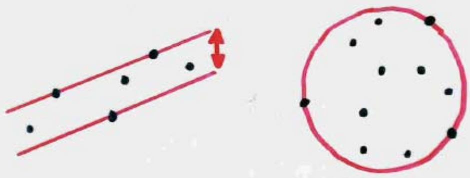


Rmk: "dual" to 3D CH but not decomposable!

Idea: [Megiddo '83 / Matoušek '92]
parametric search reduces LP to membership queries (which are decomposable)

⇒ $O(\text{polylog}\, n)$ update & query
[by new result]

Appl'ns: 2D vertical width,
2D min enclos. circle



Ex 7: 2D Bichromatic Closest Pair

Maintain n red/blue pts in 2D to find $\min_{p\ \text{red}} \min_{q\ \text{blue}} d(p,q)$

Appl'n: 2D MST

Static sol'n: $O(n \log n)$ time

Rmk: but many nearest neighbors may change!

Eppstein's general technique: [FOCS '92]

Preprocessing: ("conga line")
for $i = 1, 2, \ldots$
p_i = nearest neighbor of p_{i-1}
delete p_i

Obs: closest pair is in conga line
each pt has const in-degree

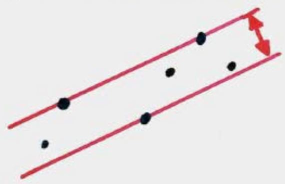
Old idea: handle insert by the logarithmic method

Extra idea: handle delete by re-insert!

⇒ $O(\text{polylog}\, n)$ update
[using new result]

Ex 8: 2D Width

Maintain n pts in 2D to find min-width enclos. strip



Static sol'n: $O(n \log n)$

Rmks: CH may change a lot!
not decomposable!
not linear/convex programming

Delete-only sol'n: [Eppstein, SODA'99/ Chan, SoCG'01]

$O(\log^3 n)$ delete time (amort.)

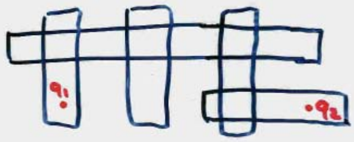
Fully dynamic idea: [Chan, SoCG'01]

decompose into 2 subsets

$\Rightarrow$ $O(\sqrt{n}\log^3 n)$ update

Ex 9: 2D Orthogonal Connectivity

Maintain n rectangles in 2D s.t.
given q_1, q_2,
decide if q_1 and q_2 connected



Similar idea: [Chan, STOC'02]

$\Rightarrow$ $O(n^{0.94})$ update &
$O(n^{1/3})$ query

Ex 10: Even Tougher Problems

- 2D min enclos. rectangle
- 2D max empty circle
- 2D discrete 1-center
  i.e. $\min_q \max_p d(p,q)$
- 3D convex hull volume

Similar idea for insert-only (or semi-online) [Chan, SODA'02]

$\Rightarrow$ $O(n^{1-\alpha})$

Rmk: but nothing for fully dynamic !!

PART II:

THE NEW RESULT

Dynamic 3D Convex Hull/ 2D Post Office

$O(\log^3 n)$ insert time (amort, rand)

$O(\log^6 n)$ delete time (amort, rand)

$O(\log^2 n)$ query time

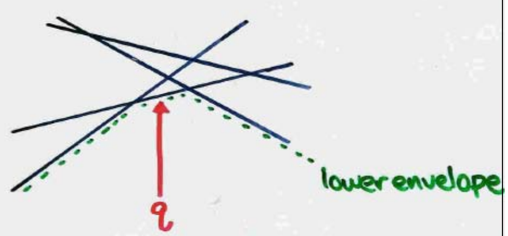
Overview of Ideas

- handle insert by the logarithmic method ($O(\log n)$ subsets)
- move "high-degree stuff" to new subsets during preprocessing [also done by Agarwal, Matoušek, FOCS'92]
- handle delete by re-inserts [inspired by Eppstein, FOCS'92]
- use $O(\log n)$ samples & "conflict lists" [inspired by skip list & a static structure by Chan, FOCS'98]

Dual Problem

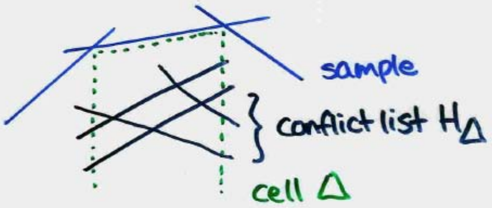
Maintain n halfspaces in 3D s.t.
given vertical line q,
find intersection at q

i.e. maintain n planes in 3D s.t.
given vertical line q,
find lowest plane at q



Sampling Lemma

For a random sample of r planes, triangulation of its lower envelope has $\sim r$ cells, each w. $\sim n/r$ conflicting planes



[Clarkson,Shor'88, Matoušek'91, Ramos'99: "shallow cuttings"]

Preprocessing a Subset

For $i = 1 .. \log n$
  take sample of 2^i planes

$\sim 2^i$ cells, each w.
$\sim n/2^i$ conflicts

move planes that have $\geq b$ conflicts to new subset

Analysis:

planes moved $\approx \frac{n}{b} \cdot \log n$

$\Rightarrow$ only a fraction moved, for $b \approx \log n$

$\Rightarrow$ preproc. time $O(n \log^2 n)$

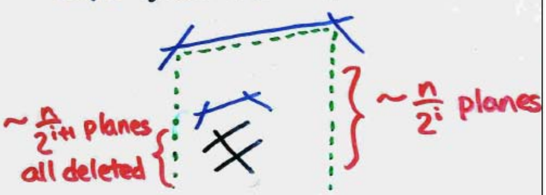
$\Rightarrow$ insert time $O(\log^3 n)$

Deleting h from a Subset

For $i = 1 .. \log n$
  for each cell Δ in conflict with h
    if a fraction of H_Δ was deleted
      re-insert all planes in H_Δ

Correctness:

suppose answer is in a cell at layer i, but not $i+1$



Analysis:

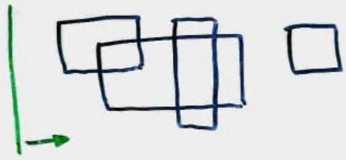
re-inserts $\approx b \log n \approx \log^2 n$

$\Rightarrow$ delete time $O(\log^2 n \cdot \log^3 n \cdot \log n)$

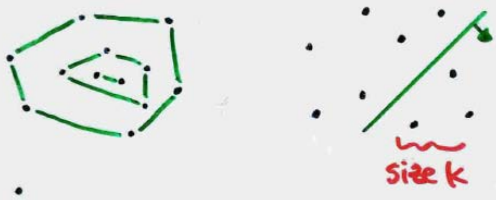
CONCLUSION

Exs of Appl'ns

- Klee's measure problem (compute volume of union of n boxes)



- 3D convex layers, k-sets



⋮

Open Problems

- dynamic 3D CH / 2D post office:
  $O(\log n)$ update & query ????
  & $O(n)$ space ??
- dynamic 2D point location:
  $O(\log n)$ update & query ??
- dynamic 2D width:
  $o(\sqrt{n})$??
- dynamic 3D approx. width (or "core-sets"
  $O(\log n)$??
- dynamic 2D connectivity for line segments
  $o(n)$??
- dynamic 2D centerpoint
  $o(n)$??

  ⋮

Geometric Networks:

Integer Linear Programming and Combinatorial Algorithms

Alexander Wolff

Department of Computer Science
Karlsruhe University

Outline

Geometric Networks

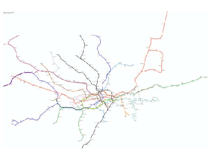
modeled by graphs $G(V, E)$ with
– $V \subset \mathbb{R}^2$
– edge weights:
distances, travel time, . . .

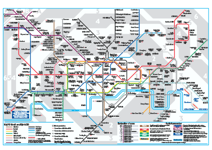
Applications:

- backbone of transportation networks
- model for communication networks
- . . .

Outline

What is a Metro Map?





- schematic diagram for public transport
- visualizes lines and stations
- goal: ease navigation for passengers
  - "How do I get from A to B?"
  - "Where to get off and change trains?"
- distorts geometry and scale
- improves readability
- compromise between schematic road map $\leftrightarrow$ abstract graph

More Formally

The Metro Map Problem

Given: planar embedded graph $G = (V, E)$, $V \subset \mathbb{R}^2$,
line cover $\mathcal{L}$ of paths or cycles in G (the metro lines),
Goal: draw G and $\mathcal{L}$ nicely.

- What is a nice drawing?
- Look at real-world metro maps drawn by graphic designers and model their design principles as
  - *hard* constraints – must be fulfilled,
  - *soft* constraints – should hold as tightly as possible.

Hard Constraints

(H1) preserve embedding of G

(H2) draw all edges as octilinear line segments, i.e. parallel to a coordinate axes or at 45° degrees

(H3) draw each edge e with length $\geq \ell_e$

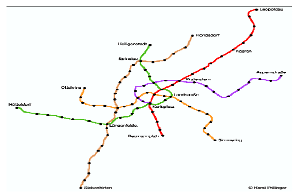
(H4) keep vertices $d_{\min}$ away from non-incident edges

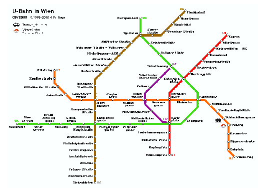
Soft Constraints

(S1) draw metro lines with few bends

(S2) keep total edge length small

(S3) draw each octilinear edge similar to its geographical orientation: keep its relative position





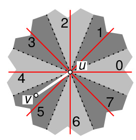
Mixed-Integer Programming

- Linear Programming: efficient optimization method for
  - linear constraints and objective function,
  - real-valued variables (domain $\mathbb{R}$).
- Mixed-Integer Programming (MIP):
  - allows also integer variables (domain $\mathbb{Z}$),
  - solution NP-hard in general.
- Still a practical method for many hard optimizat. problems.

Theorem (Nöllenburg & Wolff GD'05)

The problem MetroMapLayout can be formulated as a MIP s.th.
linear constraints → hard constraints,
objective function → soft constraints.

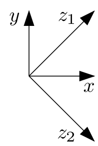
Example: Octilinearity and Relative Position



Sectors

– for each vtx. u partition plane into sectors 0–7
– number octilinear edge directions accordingly

- $\mathrm{sec}_u(v) = 5$ (input)
- $\mathrm{dir}(u, v) = 5$ (output)



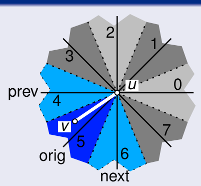
Coordinates

assign z_1- and z_2-coordinates to each vertex v:

- $z_1(v) = x(v) + y(v)$
- $z_2(v) = x(v) - y(v)$

Example: Octilinearity and Relative Position

Goal



Draw edge uv

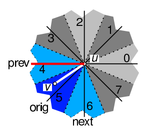
- with minimum length ℓ_{uv}
- restricted to 3 directions

How to model this using linear constraints?

Binary Variables

$$\alpha_{\text{prev}}(u, v) + \alpha_{\text{orig}}(u, v) + \alpha_{\text{next}}(u, v) = 1$$

Example: Octilinearity and Relative Position



Previous Sector

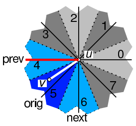
$$\begin{aligned} y(u) - y(v) &\leq M(1 - \alpha_{\text{prev}}(u, v)) \\ -y(u) + y(v) &\leq M(1 - \alpha_{\text{prev}}(u, v)) \\ x(u) - x(v) &\geq -M(1 - \alpha_{\text{prev}}(u, v)) + \ell_{uv} \end{aligned}$$

How does this work?

Case 1: $\alpha_{\text{prev}}(u, v) = 0$

$$\begin{aligned} y(u) - y(v) &\leq M \\ -y(u) + y(v) &\leq M \\ x(u) - x(v) &\geq \ell_{uv} - M \end{aligned}$$

Example: Octilinearity and Relative Position



Previous Sector

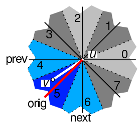
$$\begin{aligned} y(u)-y(v) &\le M(1-\alpha_{\text{prev}}(u,v)) \\ -y(u)+y(v) &\le M(1-\alpha_{\text{prev}}(u,v)) \\ x(u)-x(v) &\ge -M(1-\alpha_{\text{prev}}(u,v))+\ell_{uv} \end{aligned}$$

How does this work?

Case 2: $\alpha_{\text{prev}}(u,v)=1$

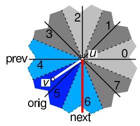
$$\begin{aligned} y(u)-y(v) &\le 0 \\ -y(u)+y(v) &\le 0 \\ x(u)-x(v) &\ge \ell_{uv} \end{aligned}$$

Example: Octilinearity and Relative Position



Original Sector

$$\begin{aligned} z_2(u)-z_2(v) &\le M(1-\alpha_{\text{orig}}(u,v)) \\ -z_2(u)+z_2(v) &\le M(1-\alpha_{\text{orig}}(u,v)) \\ z_1(u)-z_1(v) &\ge -M(1-\alpha_{\text{orig}}(u,v))+2\ell_{uv} \end{aligned}$$



Next Sector

$$\begin{aligned} x(u)-x(v) &\le M(1-\alpha_{\text{next}}(u,v)) \\ -x(u)+x(v) &\le M(1-\alpha_{\text{next}}(u,v)) \\ y(u)-y(v) &\ge -M(1-\alpha_{\text{next}}(u,v))+\ell_{uv} \end{aligned}$$

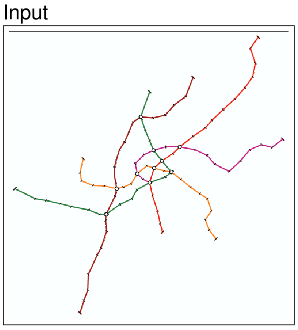
Summary of the Model

- The above constraints enforce
  - octilinearity,
  - minimum edge length,
  - (partially) relative position
- Similarly:
  - preservation of embedding
  - planarity
- Soft constraints: weighted sum in objective function

  minimize $\lambda_{\text{bends}}\,\text{cost}_{\text{bends}} + \lambda_{\text{length}}\,\text{cost}_{\text{length}} + \lambda_{\text{dir}}\,\text{cost}_{\text{dir}}$
- In total $O(|V|^2)$ constraints and variables

Results – Vienna

Input



Input	$\lvert V\rvert$	$\lvert E\rvert$	lines
normal	90	96	5
reduced	44	50	

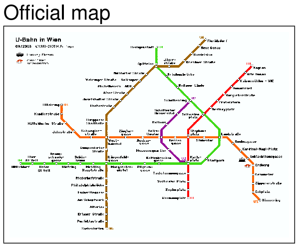
↓

MIP	constr.	var.
normal	39363	9960
improved	23226	6048
heuristic 1	5703	1800
heuristic 2*	1875	872

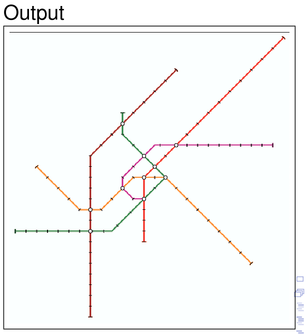
*) 29 seconds w/o proof of opt.

Results – Vienna

Official map

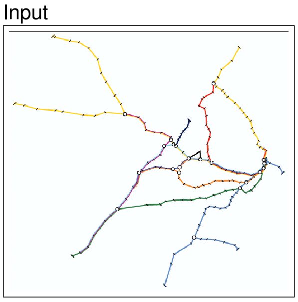


Output



Results – Sydney

Input



Input	$\lvert V\rvert$	$\lvert E\rvert$	lines
normal	174	183	10
reduced	62	71	

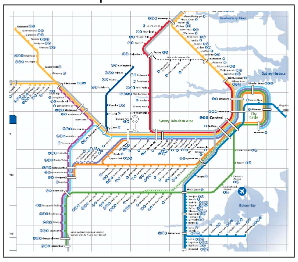
↓

MIP	constr.	var.
normal	81416	20329
improved	45182	11545
heuristic 1*	6242	2105
heuristic 2	3041	1329

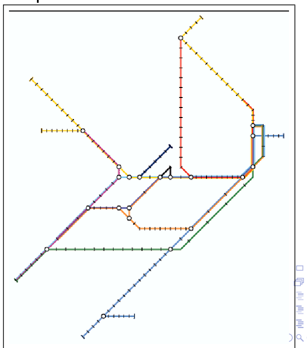
*) 22 minutes w/o proof of opt.

Results – Sydney

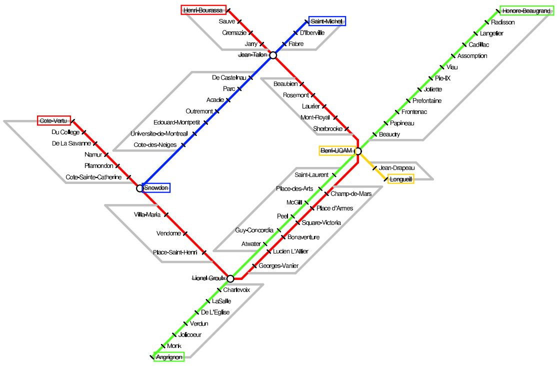
Official map



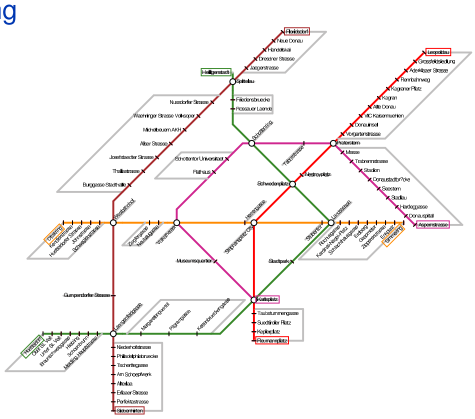
Output



Labeling



Labeling



A Similar Problem

RECTILINEARGRAPHDRAWING Decision Problem

Given a planar embedded graph G with max degree 4.
Is there a drawing of G that

- preserves the embedding,
- uses straight-line edges,
- is rectilinear?

Theorem (Tamassia SIAMJComp'87)

RECTILINEARGRAPHDRAWING *can be solved efficiently.*

Proof.

By reduction to a flow problem. □

Our Problem

METROMAPLAYOUT Decision Problem

Given a planar embedded graph G with max degree 8.
Is there a drawing of G that

- preserves the embedding,
- uses straight-line edges,
- is octilinear?

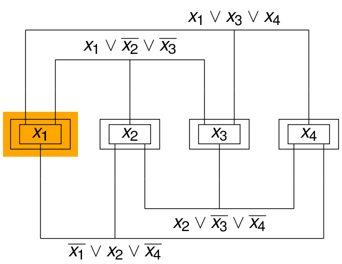
Theorem (Nöllenburg MSc'05)

METROMAPLAYOUT *is NP-hard.*

Proof.

By reduction from PLANAR 3-SAT to METROMAPLAYOUT. □

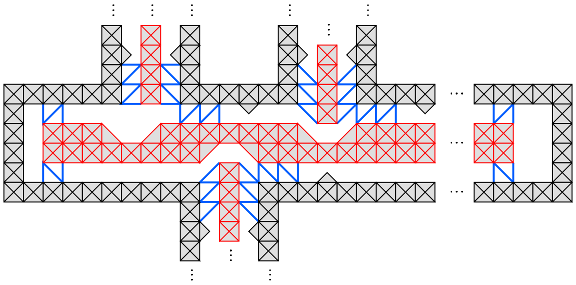
Outline of the Reduction



Input: planar 3-SAT formula $\varphi =$
$(x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge \dots$

Goal: planar embedded graph G_φ with:
G_φ has a metro map drawing $\Leftrightarrow$ φ satisfiable.

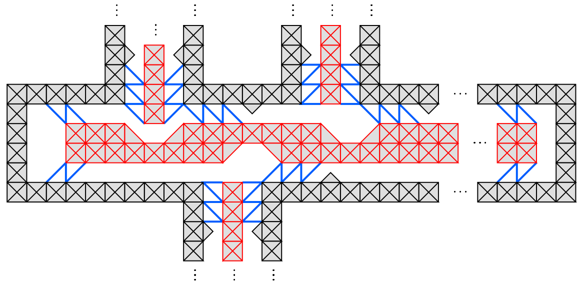
Variable Gadget



x = true

Introduction | Model
Drawing Metro Maps | MIP & Experiments
Spanning trees with few crossings | NP-Hardness

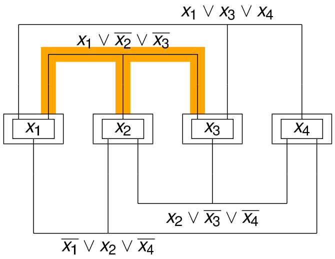
Variable Gadget



x = false

Introduction | Model
Drawing Metro Maps | MIP & Experiments
Spanning trees with few crossings | NP-Hardness

Outline of the Reduction

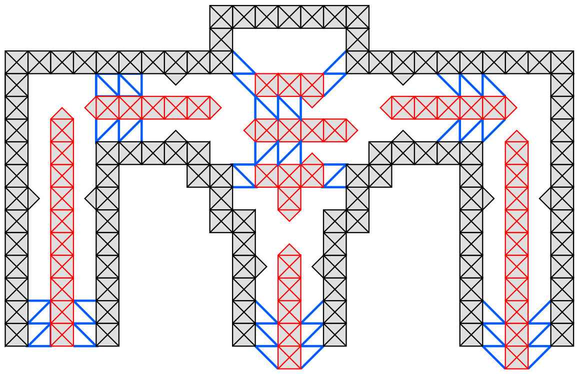


Input: planar 3-SAT formula $\varphi =$
$(x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge \ldots$

Goal: planar embedded graph G_φ with:
G_φ has a metro map drawing $\Leftrightarrow$ φ satisfiable.

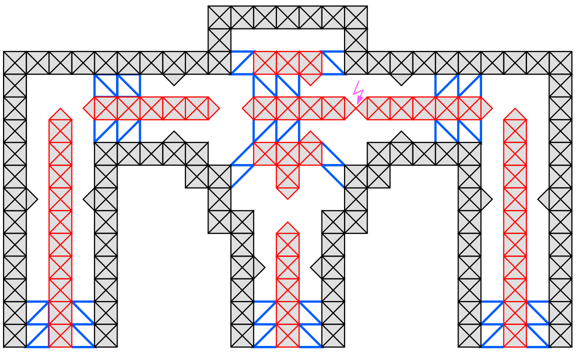
Introduction | Model
Drawing Metro Maps | MIP & Experiments
Spanning trees with few crossings | NP-Hardness

Clause Gadget



Introduction | Model
Drawing Metro Maps | MIP & Experiments
Spanning trees with few crossings | NP-Hardness

Clause Gadget



Introduction | Model
Drawing Metro Maps | MIP & Experiments
Spanning trees with few crossings | NP-Hardness

Summary (metro maps)

- METROMAPLAYOUT is NP-hard.
- Formulated and implemented MIP.
- Our MIP can draw *any* kind of sketch "nicely".
- Results comparable to manually designed maps.
- Reduced MIP size & runtime drastically.

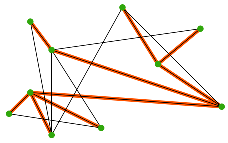
To Do

- rectangular stations
- multi-edges
- user interaction (e.g. fixing certain edge directions)

Outline

1. Introduction
   - Geometric Networks
2. Drawing Metro Maps
   - Modeling the Metro Map Problem
   - MIP Formulation & Experiments
   - NP-Hardness
3. Spanning trees with few crossings
   - Non-Approximability
   - Fixed-Parameter Algorithms
   - MIP Formulation

Spanning trees with few crossings



Given: geometric graph $G(V, E)$, i.e. $V \subset \mathbb{R}^2$.
Find: spanning tree in G with minimum number of crossings.
Known: Deciding "does G have a crossing-free spanning tree?" is NP-hard. [Jansen & Woeginger BIT'93]
We show: The problem is even hard to approximate.
But it is fixed-parameter tractable
and can be formulated as a mixed-integer program.

Crossing-free spanning trees

Let k = #crossings in G (crossing = pair of crossing edges).
$\varphi(G) = 1+$ minimum #crossings in spanning tree of G.

Theorem (Knauer, Schramm, Spillner, Wolff ISAAC'05)

If there is a poly-time $k^{1-\varepsilon}$-approximation algo $A_{1-\varepsilon}$ of $\varphi(G)$ for some $\varepsilon \in (0, 1]$, then $\mathcal{P} = \mathcal{NP}$.

Let $G' = \underbrace{(G) \bullet\!-\!\bullet (G) \bullet\!-\!\bullet (G) \bullet\!-\!\bullet \cdots \bullet\!-\!\bullet (G)}_{}$ $= k^{1/\varepsilon - 1}$ copies

$\Rightarrow$ G' has $K = k \cdot k^{1/\varepsilon-1} = k^{1/\varepsilon}$ crossings.

$\boxed{\varphi(G) = 1 \Leftrightarrow \varphi(G') = 1}$ vs. $\boxed{\varphi(G) \geq 2 \Leftrightarrow \varphi(G') \geq 1 + k^{1/\varepsilon-1}}$

$\downarrow A_{1-\varepsilon}(G')$ $\qquad$ $\downarrow A_{1-\varepsilon}(G')$

$\leq K^{1-\varepsilon} = k^{1/\varepsilon-1}$ cross. $\qquad$ $\geq 1 + k^{1/\varepsilon-1}$ cross. in sp.-tree

Fixed-parameter tractability

Aim

Solve (NP-) hard problem by restricting combinatorial explosion to parameter independent from input size.

Definition

A decision problem Π is *FPT w.r.t. a parameter* k if there is an algorithm that decides in $O(f(k) \cdot \mathrm{poly}(|I|))$ time, whether an instance I of Π is a yes- or a no-instance. ($f : \mathbb{N} \to \mathbb{N}$ arbitrary!)

Example

k-VertexCover (Can we cover all edges of G with k vertices?) is FPT.

VertexCover

Example

k-VertexCover is FPT.

Proof: search-tree method.

Take a node u with an uncovered edge $\{u, v\}$.
Either u is in a VC or the complete neighborhood of u.
Check both possibilities!
$\Rightarrow$ search tree T of size $S(k) \leq 2S(k-1) \leq 2^k$.
If in no leaf of T all edges of G are covered,
then G has no k-VC. Runtime: $O(2^k \cdot n)$ □

Example

k-DominatingSet is *not* FPT unless $\mathcal{P} = \mathcal{NP}$.

CrossingFreeSpanningTree

Theorem

CrossingFreeSpanningTree is FPT w.r.t. #crossings in the input graph.

i.e., can decide in $O(f(k) \cdot \mathrm{poly}(n))$ time, whether a graph with n vertices and k crossings has a crossing-free spanning tree.

Proof by enumeration of edges.

$X :=$ set of crossings $\Rightarrow$ $E_X := \bigcup X$ contains $\leq 2k$ edges.
$G' := G \setminus E_X$
For each subset H of E_X check:
is H crossing-free? does $G' \cup H$ span G?
Runtime: $O(2^{2k} \cdot \mathrm{poly}) = O(4^k \cdot \mathrm{poly})$. □

Note: G has c-f spanning tree $\Leftrightarrow$ G has c-f spanning subgraph.

A faster algorithm

Theorem

CrossingFreeSpanningTree can be decided in $O(3^k \cdot \text{poly})$ time.

Proof by enumeration of "crossing vectors".

$X := \{\{e_1, f_1\}, \dots, \{e_k, f_k\}\}$ = set of crossings.
For each vector $x \in \{0, 1, 2\}^k$

$$H := \bigcup_{i=1}^{k} \begin{cases} e_i & \text{if } x_i = 1, \\ f_i & \text{if } x_i = 2, \\ \emptyset & \text{else.} \end{cases}$$

is H crossing-free? does $G' \cup H$ span G? □

A yet faster algorithm

Theorem

CrossingFreeSpanningTree can be decided in $O(2^k \cdot \text{poly})$ time.

Proof by enumeration of *maximal* crossing-free edge sets.

Let $e \in E_X \quad \Rightarrow \quad e$ crosses $c \geq 1$ other edges.
If we put e into a crossing-free subset, we discard c crossings.
If we don't put e into a c-f subset, we discard c crossings.

$\Rightarrow E_X$ has $S(k) \leq 2S(k - c) \leq 2S(k - 1) \leq 2^k$ max. c-f subsets
For each such subset H check:
does $G' \cup H$ span G? □

Our results

Theorem (KSSW'05)

CFST can be decided in $O(1.9999992^k \cdot \text{poly})$ time.

Proof.

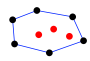
Exploit that edge sets H must make G' connected:
(1) if G' has many components, one must have low degree:
$\Rightarrow$ use search tree based on components.
(2) if G' has few components, there are few ways to connect G':
$\Rightarrow$ enumerate all possibilities. □

Theorem (KSSW'05)

There is a Monte-Carlo algo. (1-sided error, success prob. $> \frac{1}{2}$) for CFST that runs in $O(1.968^k \cdot \text{poly})$ time.

Another parameter

Let ι = number of inner points:



Theorem (Deineko, Hoffmann, Okamoto, Woeginger ORL'06)

TravelingSalesPerson (TSP) is FPT w.r.t. ι.

Theorem (H., O. IWPEC'04; Grantson, Levcopoulos EWCG'06)

MinimumWeightTriangulation (MWT) is FPT w.r.t. ι.

Theorem (G. & L. JCDCG'04; Spillner CCCG'05)

MinimumConvexPartition (MCP) is FPT w.r.t. ι.

Theorem (Knauer & Spillner Manu'06)

CrossingFreeSpanningTree (CFST) is FPT w.r.t. ι.

A MIP for MinimumCrossingSpanningSubgraph

For each edge e of G let

$$y_e = \begin{cases} 1 & \text{if } e \text{ is in the desired spanning subgraph of } G, \\ 0 & \text{otherwise.} \end{cases}$$

We would like to have for each crossing $\{e, e'\} \in X$ a variable

$$x_{ee'} = \begin{cases} 1 & \text{if } y_e = y_{e'} = 1, \\ 0 & \text{otherwise.} \end{cases}$$

Then we could state as objective function

$$\text{minimize} \sum_{\{e,e'\} \in X} x_{ee'}.$$

Clearly $x_{ee'} = y_e \cdot y_{e'}$

A MIP for MinimumCrossingSpanningSubgraph

For each edge e of G let

$$y_e = \begin{cases} 1 & \text{if } e \text{ is in the desired spanning subgraph of } G, \\ 0 & \text{otherwise.} \end{cases}$$

We would like to have for each crossing $\{e, e'\} \in X$ a variable

$$x_{ee'} = \begin{cases} 1 & \text{if } y_e = y_{e'} = 1, \\ 0 & \text{otherwise.} \end{cases}$$

Then we could state as objective function

$$\text{minimize} \sum_{\{e,e'\} \in X} x_{ee'}.$$

Clearly $x_{ee'} = y_e \cdot y_{e'}$, but this is not linear!

Expressing $x_{ee'}$

Reminder: we want $x_{ee'} = \begin{cases} 1 & \text{if } y_e = y_{e'} = 1, \\ 0 & \text{otherwise.} \end{cases}$

We can also express this as $x_{ee'} = \min\{y_e, y_{e'}\}$.

So what about $x_{ee'} \le y_e$ and $x_{ee'} \le y_{e'}$ (and $x_{ee'} \ge 0$)?

Recall our objective function: minimize $\sum_{\{e,e'\} \in X} x_{ee'}$:-(

Now here's the trick: $x_{ee'} \ge 0$ and $x_{ee'} \ge y_e + y_{e'} - 1$:-)

To do: must make sure that edges e with $y_e = 1$ span G!

Idea: use flow to achieve connectivity!

skip one! skip two!

From flow to connectivity [Williams'01, Shirabe'04]

Make two arcs (u, v) and (v, u) for each edge $\{u, v\}$ in G.
Let A = arc set. Let $f_a \in \mathbb{N}$ be the amount of flow on arc a.
Choose arbitrary *source* vertex s.
Make s "pour" $n - 1$ buckets of water into the graph G:

$$\sum_{(s,u) \in A} f_{(s,u)} = (n-1) + \sum_{(u,s) \in A} f_{(u,s)}$$

Make each other vertex v "drink" one bucket:

$$\sum_{(v,u) \in A} f_{(v,u)} = -1 + \sum_{(u,v) \in A} f_{(u,v)}$$

$\Rightarrow$ the set of arcs with non-zero flow is connected!

From paths to connectivity

Choose arbitrary *center* vertex s. Go through all $t \neq s$.

Idea: establish simple s–t path π^t. Let $p_e^t = \begin{cases} 1 & \text{if } e \in \pi^t \\ 0 & \text{else.} \end{cases}$

Make sure π^t leaves s and reaches t:

$$\sum_{e \text{ incident to } s} p_e^t = \sum_{e \text{ incident to } t} p_e^t = 1.$$

Make sure π^t does not stop in any vertex v other than s and t:

$$\{0, 2\} \ni \sum_{e \text{ incident to } v} p_e^t = 2d_e^t \quad \text{with } d_e^t \in \{0, 1\}.$$

Synchronize y_e with all variables of type p_e^t:

Summary (spanning trees)

- MinimumCrossingSpanningSubgraph:
  - hard to approximate,
  - MIP of size $O(mn + k)$.
- Fixed-parameter tractability:
  - interesting concept.
- CFST is FPT w.r.t.
  - #crossings,
  - #inner points.

To Do

- Beat running time of $O(1.9999992^k \mathrm{poly})$ for CFST.
- Reduce size of MIP.
- Find other parameters – and motivate them!

Geometric embeddings and graph expansion

James R. Lee
jrl@ias.edu
Institute for Advanced Study (Princeton)
and University of Washington (Seattle)

Abstract

Beginning in the early-90's, it became gradually apparent that techniques from high-dimensional geometry were highly relevant for a variety of computational tasks. At the center of this connection is the study of *graph partitioning problems* which involve breaking a graph into two or more large parts while minimizing the size of the "interface" between them. These problems are of central importance in numerous computational settings like data clustering, divide and conquer algorithms, and packet routing in networks. Additionally, the techniques involved have applications to areas like Markov chains, nearest-neighbor search, and learning theory.

We will discuss an approach to these problems based on *geometric embeddings.* The basic idea is to represent a graph $G = (V, E)$ as a geometric object by mapping the vertices of G to points in a high-dimensional Euclidean space, and then to use geometric techniques to find good cuts in G—these are cuts which separate G into "large" pieces while minimizing the number of edges that are cut.

In the *Sparsest Cut problem*, the input is given as a graph $G = (V, E)$ along with a subset of pairs of vertices $D \subseteq V \times V$ called *demands.* For a subset $S \subseteq V$, let $E(S, \bar{S})$ be the set of edges crossing from S to its complement $\bar{S}$, and let $D(S, \bar{S})$ be the set of demands crossing from S to $\bar{S}$. The *sparsity of a cut* $(S, \bar{S})$ is defined as the ratio

$$\Phi(S) = \frac{|E(S, \bar{S})|}{|D(S, \bar{S})|},$$

and the goal of the Sparsest Cut problem is to find the cut $(S, \bar{S})$ which minimizes $\Phi(S)$. Since solving this problem exactly is NP-hard, we will instead search for a set $S \subseteq V$ for which $\Phi(S)$ is *approximately optimal,* i.e. is within some factor C of the sparsest cut.

During the talk, I will first explain the relationship between geometric embeddings and the Sparsest Cut problem, and then I will discuss the current state-of-the-art techniques in constructing such embeddings. These employ a beautiful mix of semi-definite programming, high-dimensional convex geometry, probability theory, and combinatorics. A brief outline of the talk (with references) now follows.

The relation of geometric embeddings to the Sparsest Cut problem was first realized in the two papers [LLR95, AR98]. The basic approach follows.

1. Associate a *metric* to the graph G. This is a value $d(u,v)$ for every pair of vertices $u,v \in V$ which is symmetric, i.e. $d(u,v) = d(v,u)$ and satisfies the triangle inequality

$$d(u,v) \le d(u,w) + d(w,v)$$

for every triple of points $u,v,w \in V$. This metric is closely related to the structure of *multicommodity flows in G,* and is the solution to a linear programming relaxation of the Sparsest Cut problem.

2. Given the metric d, we can think of (V,d) as a metric space. The second step is to *embed* this space into $\mathbb{R}^n$ (where $n = |V|$) such that the embedding preserves the structure of (V,d). Such a mapping is called a *low-distortion embedding.* This is a map $f : V \to \mathbb{R}^n$ which satisfies

$$d(u,v) \le \|f(u) - f(v)\| \le C \cdot d(u,v) \qquad \text{for all } u,v \in V.$$

The factor C is called the *distortion* of the map f, and our goal is to make C as small as possible. (Note that $\|f(u) - f(v)\|$ is the Euclidean distance between $f(u)$ and $f(v)$.)

3. The final step is to use the geometric embedding to find a cut in G. One way to do this is by projecting the image of G (under the embedding f) onto a randomly oriented line. If we number the points of G from left to right on the line $1, 2, \ldots, n$ then there are $n-1$ cuts of the form $(\{1, 2, \ldots, i\}, \{i+1, i+2, \ldots, n\})$. The final step of the algorithm returns the cut $(S, \bar{S})$ among these which has the smallest sparsity $\Phi(S)$.

Remarkably, it turns out that with high probability, the cut we return after step (3) will be within a factor C of optimal, where C is the distortion of the embedding from step (2)!

After discussing this general approach, we will examine the new techniques that achieve the best-known approximation factor for Sparsest Cut. These are based on semi-definite programming and novel high-dimensional arguments from [ARV04] (see also [Lee05, CGR05]), as well as new techniques for constructing low-distortion embeddings [KLMN05, Lee05, ALN05].

References

[ALN05] S. Arora, J. R. Lee, and A. Naor. Euclidean distortion and the Sparsest Cut. In *37th Annual Symposium on the Theory of Computing*, pages 553–562, 2005. Full version available at `http://www.cs.berkeley.edu/~jrl`.

[AR98] Yonatan Aumann and Yuval Rabani. An $O(\log k)$ approximate min-cut max-flow theorem and approximation algorithm. *SIAM J. Comput.*, 27(1):291–301 (electronic), 1998.

[ARV04] Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings, and graph partitionings. In *36th Annual Symposium on the Theory of Computing*, pages 222–231, 2004.

[CGR05] S. Chawla, A. Gupta, and H. Räcke. Embeddings of negative-type metrics and an improved approximation to generalized sparsest cut. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 102–111, Vancouver, 2005.

[KLMN05] R. Krauthgamer, J. R. Lee, M. Mendel, and A. Naor. Measured descent: A new embedding method for finite metrics. *Geom. Funct. Anal.*, 15(4):839–858, 2005.

[Lee05] James R. Lee. On distance scales, embeddings, and efficient relaxations of the cut cone. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 92–101, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[LLR95] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

Distance Trisector curve and Voronoi diagram with neutral zone

Takeshi Tokuyama (Tohoku U)

joint work with Tetsuo Asano (JAIST)

and Jiri Matousek (Charles U)

Grand Challenge of NHC project

- Throw right on **Computational Barriers**
  - Identifying barriers (something **looking impossible**)
  - Breaking/proving/avoiding barriers
  - Making properties of barriers clearer
  - Well-known barriers in TCS
    - Computability
    - P vs NP, NP vs PSPACE etc
    - Approximation hardness, approximation ratio
    - Randomness (P vs PP, primarity checking, Yao's minmax principle)
    - Lower bounds for online algorithms
    - Cryptographic barriers (Discrete log etc)
  - Hopefully, find some more new barriers, because they will help progress of TCS

Classical barriers in math history

Let no one ignorant of Mathematics enter here. (plato)

- Incomputable geometric/arithmetic problems
  - Compute diagonal length of a unit square
  - Find a cube of volume 2
    - Dilemma of Pythagoreans
  - Draw a circle with the unit area
    - *It is easier to square the circle than to get round a mathematician.* *August de Morgan*
  - Draw regular n-gon
  - Trisect a given angle

Progress in math history

God created the integers, all else is the work of man.
(*Leopold Kronecker*)

- *Necessity is the mother of invention.*
  - Diagonal length computation — Irrational numbers
  - Cube with volume 2 — Radicals
  - Unit area circle — Transcendental number π
  - Draw regular n-gon
  - Trisect a given angle

  → "Solvability" of algebraic equations → Complex numbers, Groups, Elliptic functions

Inventing tools to measure what we cannot directly measure.

Same in CS: NP-complete theory, Proof checking,….

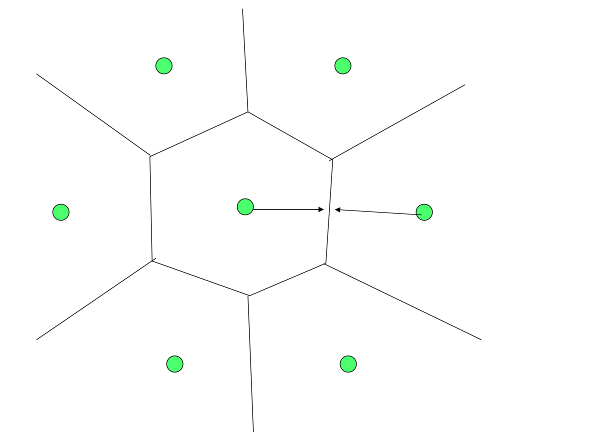
Tool to resolve a small but curious barrier.

Story of this talk

- In a computational geometric problem , we find a simple and natural geometric tool named *distance trisector curve*.
- A new transcendental curve??
- Some initial results have been obtained
  - Existence and uniqueness of the trisector curves
  - An algorithm to compute it.
  - Use of the trisector curves.
- A new possibility in computational geometry
  - Relaxed "computability" of geometric problems.

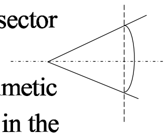
Voronoi diagram

- Subdivision of plane (space) into cells
  - $S = \{p_1, p_2, \ldots p_n\}$ points in the plane
  - $V(p_i) = \{ x : d(x, p_i) < d(x, p_j) \text{ for all } j \neq i\}$
    - Voronoi cell: dominating region of p_i
- Great geometric structure with many applications
  - Mesh generation , Graphics,
  - Simulating economic/political equilibrium
  - Simulating biological cells / crystallization
- Efficient algorithm : O(n log n) time
- Many variants: VD of lines/discs/regions, higher dimension, non-Euclidean metric, power diagrams.



Voronoi diagram is easy

- Drawn by using "perpendicular bisectors"
  - Anciently known how to draw the bisector
    - Tool to bisect an angle
  - Can be drawn by using rational arithmetic
  - It is trivial that a Voronoi diagram is in the arrangement of bisectors (size is polynomial)
    - Naïve polynomial time algorithm
    - Only need to speed up the computation
  - Known variants are also mathematically conventional



Motivation from nature

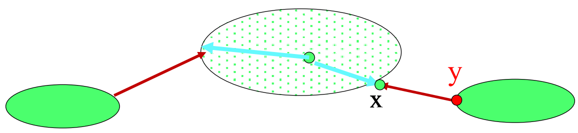
- We often see beautiful patterns with "objects" and "neutral zone" in natural structures
- Can we have a variation of Voronoi diagram to draw such a picture?

Voronoi diagram with neutral zone

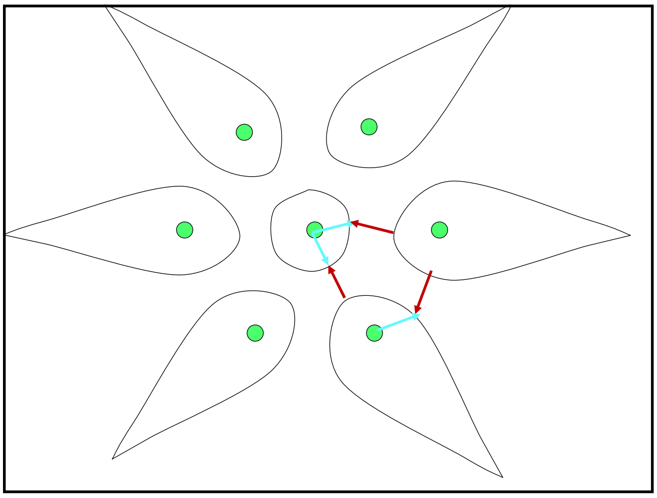
- Classical Voronoi diagram: Partition the plane (space) into domination regions of input points.
- Idea: Partition the space (plane) into domination regions and a *neutral zone*
  - There may be many mathematical formulations
  - We adopt a natural formulation

Definition of N-Voronoi diagram

- $S = \{p_1, p_2, \ldots p_n\}$ points in the plane
- $NV(p_i) = \{ x : d(x, p_i) < d(x, y)$ for every $y \in NV(p_j)$ for $j \neq i\}$

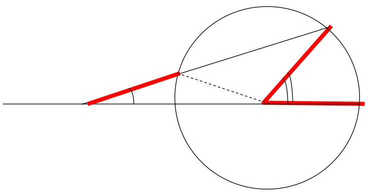


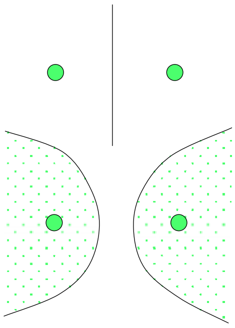
N-Voronoi diagram on 7 points



Questions

- Bisector ??
- **Distance trisector curves:**
  - Known or unknown?
  - As easy as angle trisection?

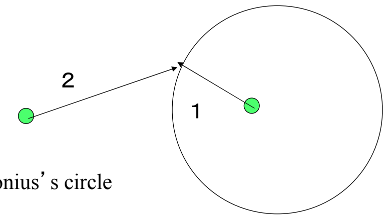




Ruler, Compass, and red-ink (Archimedes)

Distance trisector

- Different from Apollonius's circle
- Natural and simple definition (see next slide)
- Surprisingly, seems to be a new curve



Apollonius' s circle

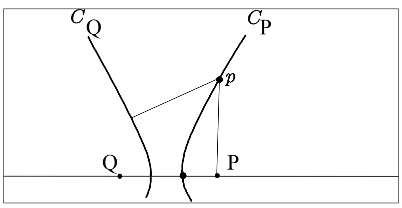
Distance Trisector Curves

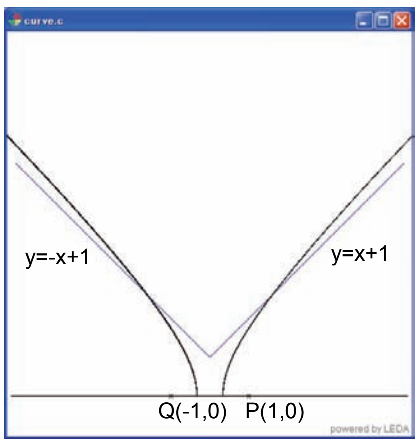
Equally-spaced curves C_P and C_Q:

for any point p on C_P, ***dist(p, P) = dist(p, C_Q)*** and

for any point q on C_Q, ***dist(q, Q) = dist(q, C_P)***.

where ***dist(p, C)*** is the distance from ***p*** to a point on ***C*** that is closest to p.

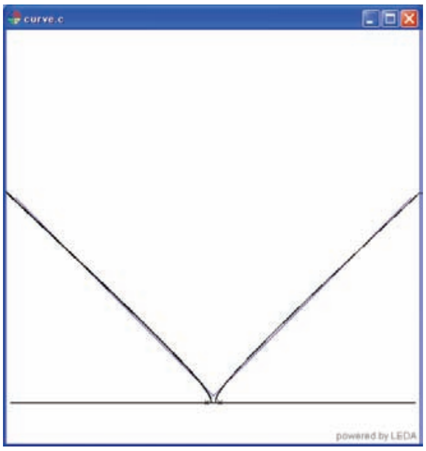




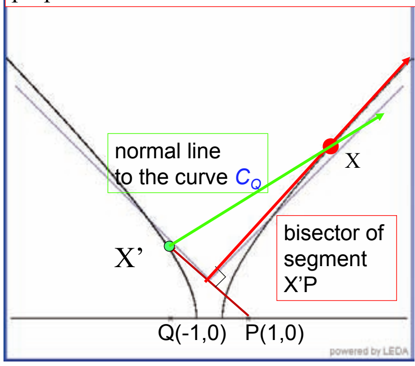
Note:

Drawing curves was not easy.

Only possible after revealing some theoretical results

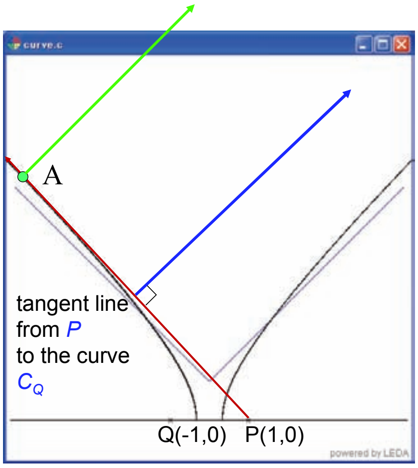


Observation: Each point X on C_P is the intersection of the normal line of C_Q at its partner point X' and the perpendicular bisector of X'P



Corollary.
If we exactly determine C_P at a small neighbor of P, we can determine whole C_P

We have a system of differential equation



Implication

•No point beyond the tangent point A can be a partner point of a point of NV(P)

Indeed, it suffices to compute C_P for

x<5.65

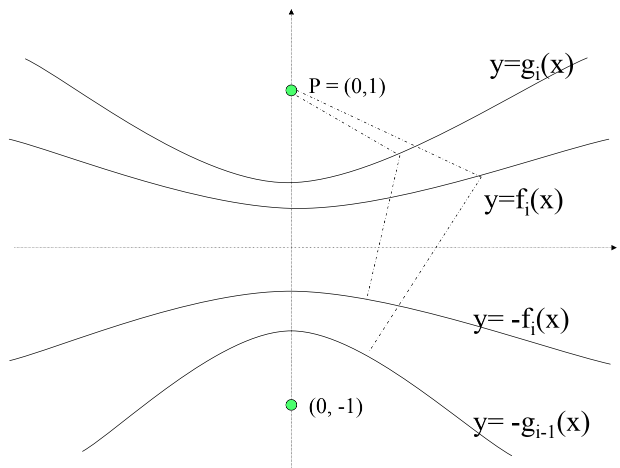
Nice nature of the curve

- Convex and smooth (revealed to be analytic)
- Satisfies a system of differential equations
  - Specialists did not know how to solve it
- It suffices to compute in the range [0, 5.65]
  - Important curves satisfy this
    - sin (x) $[0, 2\pi]$, log x [1,10]
- Problems: Existence, uniqueness, computation

Existence and uniqueness

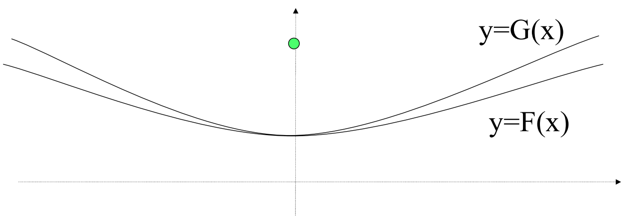
- P=(0,1), Q = (0,-1)
- $y=f_1(x)$ is x-axis (bisector of PQ)
- $y = g_1(x)$:bisector parabola of P and x-axis
- $y=f_j(x)$: bisector curve of P and $y=-g_{j-1}(x)$
- $y=g_j(x)$:bisector curve of P and $y= -f_j(x)$

Lemma. The trisecting curve C_P must be above $f_j(x)$ and below $g_j(x)$



Lemma: The series $f_j(x)$ and $g_j(x)$ converges to continuous functions F(x) and G(x),respectively

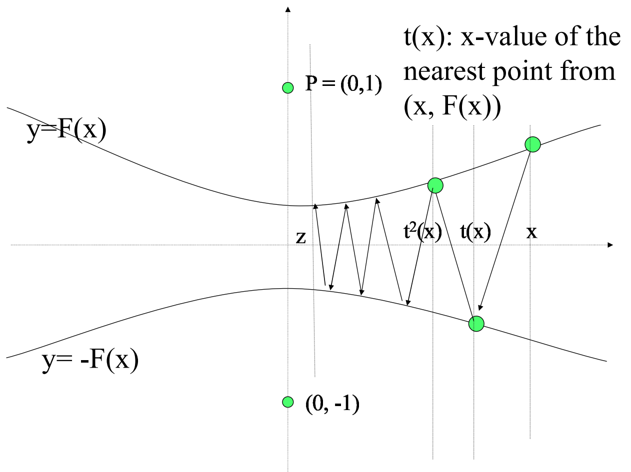
Theorem (**Existence and Uniqueness**) . G(x) ≡ F(x), and y= F(x) gives the trisector curve C_P

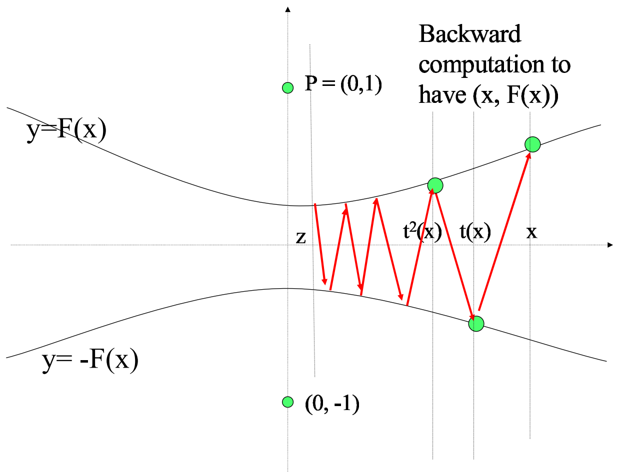


Computation of F(x)

- Given a value of x >0 , compute F(x) to any precision (error is smaller than given ε)

Theorem. We can compute F(x) to the accuracy ε in O(log ((1+x) / ε)) time





Idea for computing F(x)

- If -t(x) is the x-value of the partner point, $t(x) < \beta x$ for a constant $\beta < \sqrt{3} - 1 < 1$
- We can compute the Taylor expansion of F(x) and t(x) around zero to any precision.
  - F(x) and t(x) are computable if x < h for a small h.
- F(x) and x are computable from t(x), t(t(x)), F(t(x)), and F(t(t(x))).
- Guess $z = t^k(x) < h$, compute F(z) and F(t(z)), and recursively compute $t^i(x)$ for i=k, k-1, k-2, …, 0
- Determine correct value of z via binary search

Formulas to compute F(x)

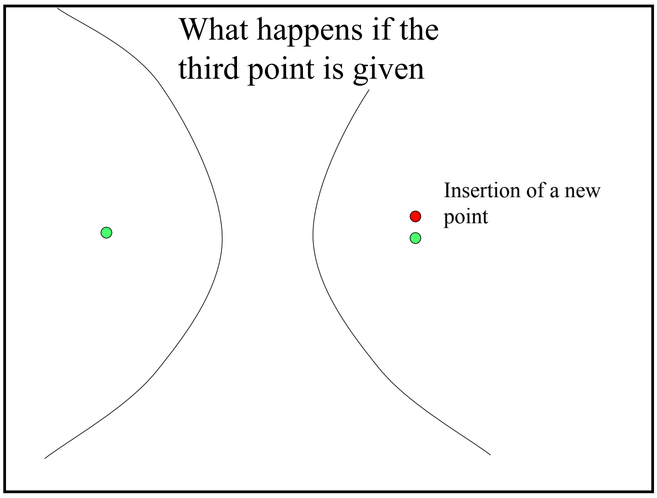
- x = Φ(t(x), t(t(x)),F(t(x)), F(t(t(x)))
- F(x) = ψ(t(x), t(t(x)),F(t(x)), F(t(t(x)))

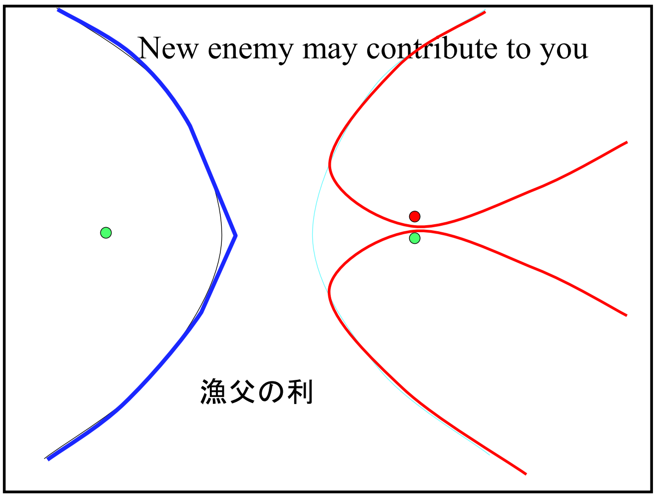
- $\Phi(x,y,u,w) = x + y(x^2+(1+u)^2)/2Q(x,y,u,w)$
- $\psi(x,y,u,w) = (2xyu + (1+u)(1+x^2-u^2)) / 2Q(x,y,u,w)$
- $Q(x,y,u,w) = (1+u)(1+v)-xy$

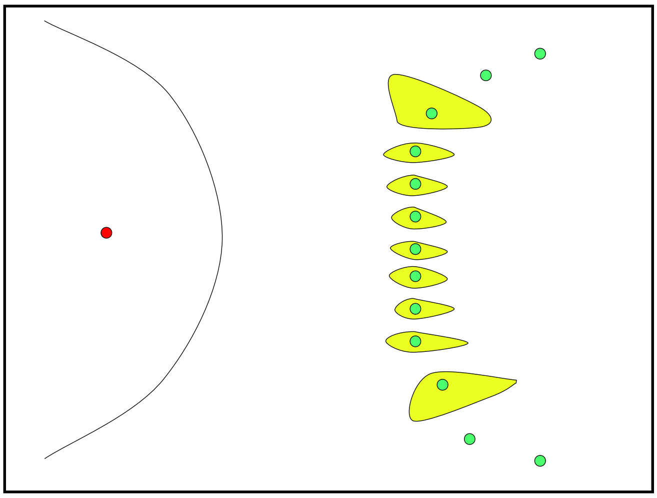
The above mentioned: STOC06, to appear

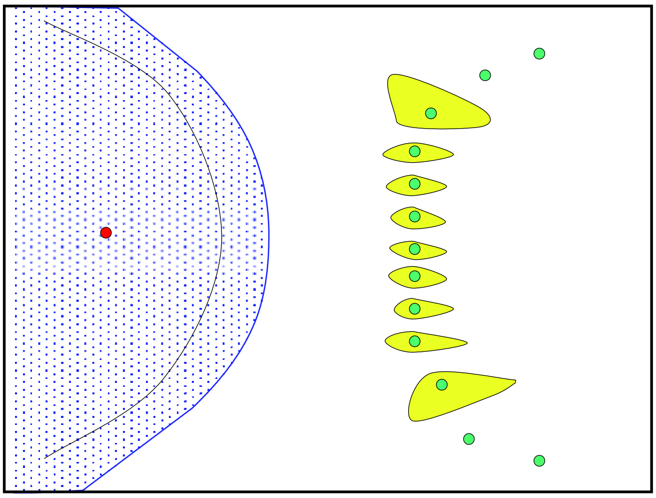
N-Voronoi diagram on many points

Now we have trisector curves.
Can we draw N-Voronoi diagram of more than two points by using them??







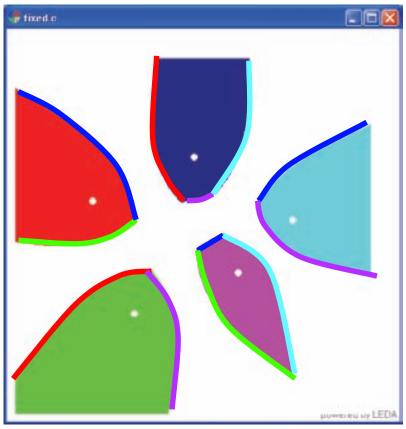


Basic properties

- Boundary curve of NV(p_i) is the **bisector** between p_i and (the union of) boundary curves of other regions
- Each region is **convex** and nonempty
- It may happen that there is no unbounded region (different from an ordinary Voronoi diagram)

Voronoi edges

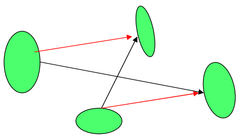
- Definition: If p is a boundary point on $NV(p_i)$ and q is its nearest point among other region boundaries, we call q the partner point of p.
- Definition: The boundary of $NV(p_i)$ is decomposed into curve segments each of which consists of points whose partner points are in a same region. The connected components are called Voronoi edges.



Combinatorial complexity

- Theorem. Number of Voronoi edges is O(n) in an N-Voronoi diagram on n points.

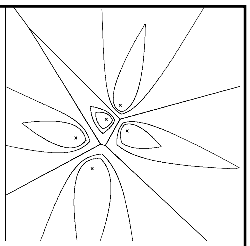
Proof: Rays towards nearest enemy's-boundary do not cross each other (cheating a little) reduced to the linearity of number of edges in a planar graph +Davenport-Schinzel sequence argument.



Questions

- N-Voronoi diagram always exists?
  - Yes
- Unique for a given point set?
  - Yes
- Efficient algorithm exist?
  - Yes, if we are given some oracles
- Really efficient in practice?
  - No……

Existence



Given a set of regions $R_1, R_2, \ldots R_n$ such that

R_j contains p_j, consider an operator $\boldsymbol{F}$

$\boldsymbol{F}(R_1, R_2, \ldots R_n) = (Q_1, Q_2, \ldots Q_n)$

where $Q_j = \{ x\colon\ d(x, p_j) < d(x, y)$ for $\forall y \in \cup_{i \neq j} R_i \}$

Theorem. F has at least one fixed point

From Schauder-Kakutani's fixed point theorem

Corollary. N-Voronoi diagram is given as a fixed point of F

Note: Fixed point theorem

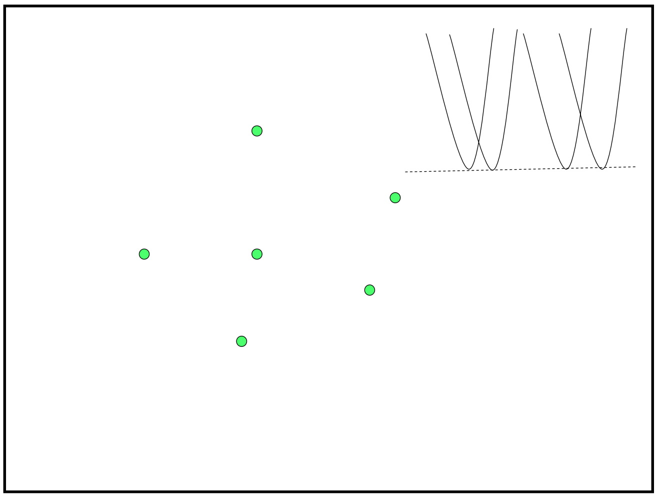
Brower(1910),Schauder(1930),Kakutani(1941)

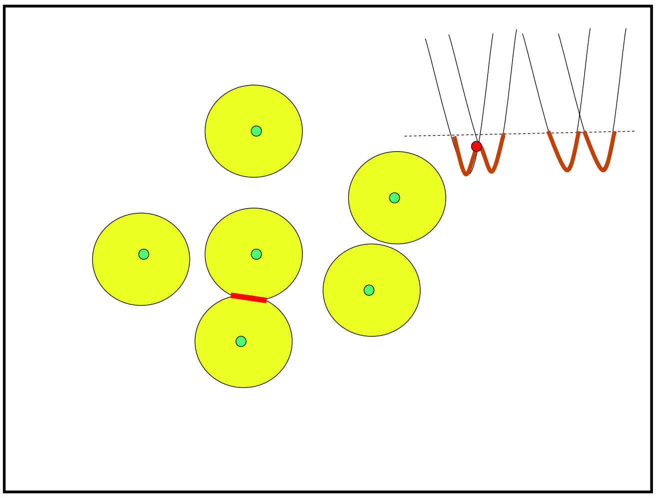
Z: Banach space,
K: Compact convex subset of Z (nonempty)
F: K K continuous map
Then, F has a fixed point

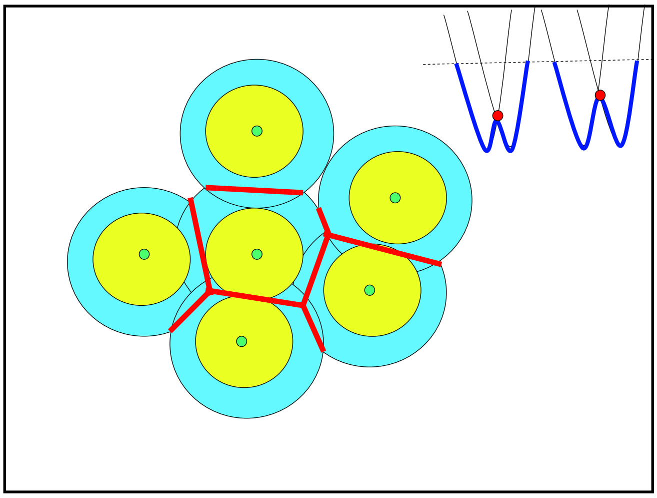
- Z: space of n-tupple of convex regions $(R_1, R_2, .. R_n)$
- We need to introduce norm, and define convexity etc…
- Continuous dimensional space (Shauder-Kakutani's version)

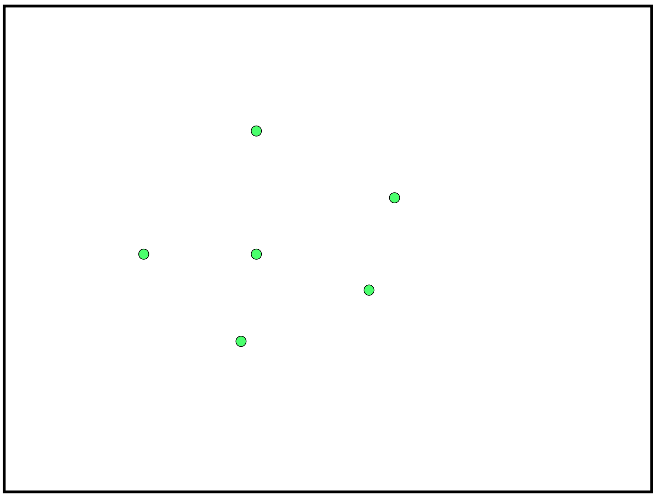
Uniquness

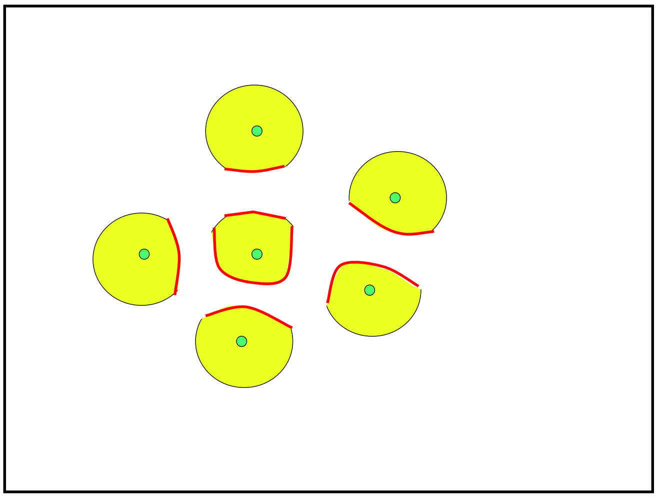
- Fixed point theorem does not assure uniqueness.
- Uniqueness is given in a constructive fashion
- Crystallization algorithm:
  - Growing radius of disks
  - Analogous to the space-sweep algorithm for computing a (classical) Voronoi diagram as a lower envelope of parabolic cylinders.

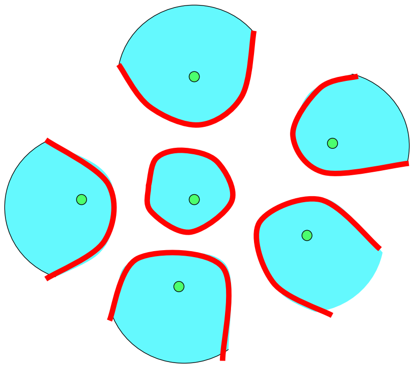












Curves in a N-Vonoroi diagram

Property 1: N-Voronoi diagram is a subset of an arrangement of curves in a curve family F

F consists of :

•Distance trisectors of pairs of input points are in F.

•Bisector curves of an input point and a curve (or point) C in F.

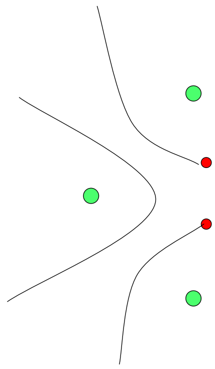
• Intersection points of curves in F.

Property 2: The number of applications of bisector operations is finite to obtain F

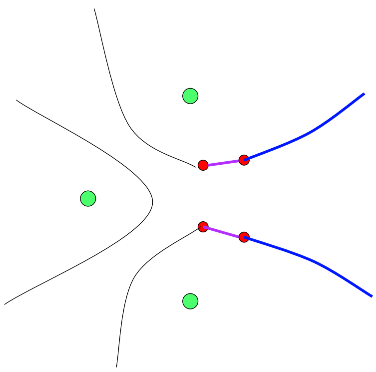
Analysis of crystallization algorithm

- For planar case, the algorithm shows uniqueness of N-Voronoi diagram
- Number of "structural changes" is O(n)
- Terminates in finite steps if we can draw
  1. bisector curve of a point and given point
  2. distance trisector curve
- Terminates in polynomial time if we assume that we can draw an "generalized" trisector curve

Generalized trisector curves



Generalized trisector curves



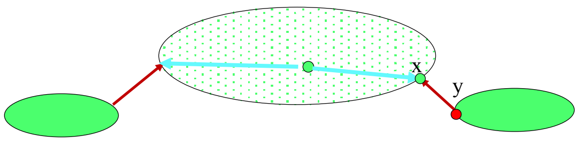
Some more ideas and problems

- Computability of geometric problems
  - Is the N-Voronoi diagram computable?
  - Bisector of a point and a curve not easy
    - Not difficult in digital (pixel) geometry
  - Especially, the trisector curves are only given by point-wise enumeration
    - Same situation as sin and cos curves.
  - Current algorithm needs iterative computation of bisectors accumulating error!
  - Is there fundamental difference from some "paper algorithms" in computational geometry?

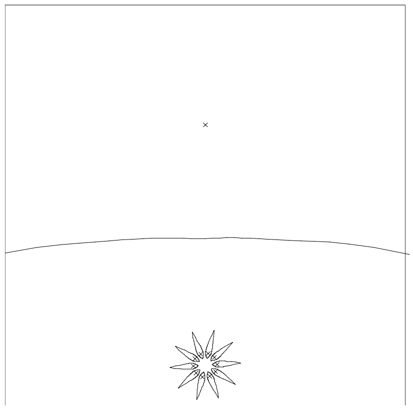
Generalized N-Voronoi diagram

- $S = \{p_1, p_2, \ldots p_n\}$ points in the plane
- $GNV(p_i) = \{ x : d(x, p_i) < \alpha\ d(x, y)$ for every $y \in GNV(p_j)$ for $j \neq i\}$

if α ∞、Voronoi diagram.



Convexity fails in an N- Voronoi diagram for $\alpha = 1.5$

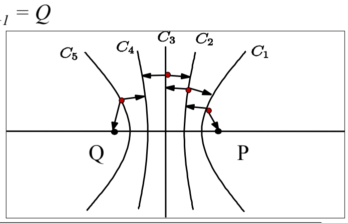


Multi-sector curves

- Given two points ***P*** and ***Q*** in the plane and an integer ***n***, draw *n* equally-spaced curves $C_1, C_2, \ldots, C_n$ between them s.t. for any point p on a curve C_i,

  $dist(p, C_{i-1}) = dist(p, C_{i+1})$

  $C_0 = P$, and $C_{n+1} = Q$



Trisector is the case where n=2

Conclusion

- N-Voronoi diagram: Interesting generalization of Voronoi diagram that admits a neutral zone.
- Many unsolved questions:
  - Is the trisector curve really transcendental?
  - Concrete analysis of N-Voronoi diagram algorithms.
  - Uniqueness of N-Voronoi diagram of a given point is not clear for high-dimensional cases
  - Generalization for $\alpha > 1$ (convexity fails)
  - N-sector curves