Seminar *Optimierungsprobleme in der Bioinformatik*

Protein Threading

Hans Joachim Ferreau

`ferreau@urz.uni-heidelberg.de`

20.04.2005

Inhalt

- Biologischer Hintergrund
- Modellierung
- MIP-Formulierungen
- Zusätzliche Lösungsstrategien
- Zusammenfassung
- Literatur

Biologischer Hintergrund

- **Biologischer Hintergrund**
- Modellierung
- MIP-Formulierungen
- Zusätzliche Lösungsstrategien
- Zusammenfassung
- Literatur

Proteine

- **Biologischer Hintergrund**
  - **Proteine**

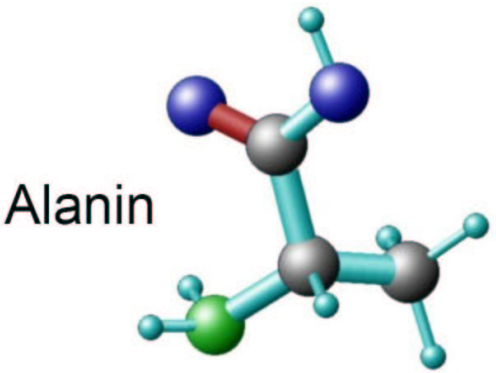
    Protein Threading

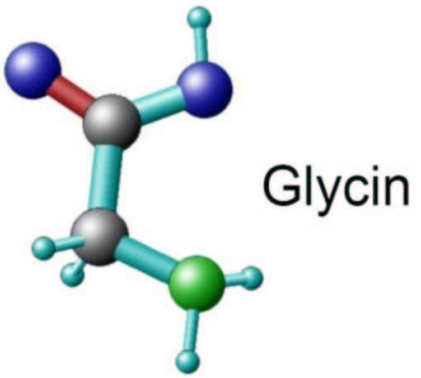
Proteine · Motivation

- Proteine sind die wichtigsten Substanzen in allen lebenden Organismen
- Ihre Funktionen sind beispielsweise:
  - Aufbau von Geweben und Organen
  - Immunabwehr
  - Steuern von Nervenimpulsen
- Verständnis ihrer Funktionsweisen verspricht daher weitreichende medizinische Anwendungen
- Dazu Auswertung der Daten des Human-Genom-Projekts

Proteine · Aufbau

- Ein Protein ist eine unverzweigte Kette von etwa 100 bis 5 000 *Aminosäuren*
- Jede Aminosäure ist ein Molekül aus ca. 10-30 Atomen:





- An der Proteinbildung sind 20 verschiedene Aminosäuren beteiligt; jeder ordnet man einen Buchstaben zu

Proteine · Amino-Alphabet

- Das Amino-Alphabet:

A	Alanin	M	Methionin
C	Cystein	N	Asparagin
D	Asparaginsäure	P	Prolin
E	Glutaminsäure	Q	Glutamin
F	Phenylalanin	R	Arginin
G	Glycin	S	Serin
H	Histidin	T	Threonin
I	Isoleucin	V	Valin
K	Lysin	W	Tryptophan
L	Leucin	Y	Tyrosin

Proteine · Primärstruktur

- Anzahl, Art und Reihenfolge der Aminosäuren sind in der DNA als Gen kodiert
- Identifizierung eines Gens ergibt Primärstruktur eines Proteins:

```
KDIQLLNVSYDPTRELYEQYNKAFSAHWKQETGDNVVIDQSHGGSGKQATSV
INGIEADTVTLALAYDVNAIAERGRIDKNWIKRLPDDSAPYTSTIVFLVRKG
NPKQIHDWNDLIKPGVSVITPNPKSSGGARWNYLAAWGYALHHNNNDQAKAE
DFVKALFKNVEVLDSGARGSTNTFVERGIGDVLIAWENEALLATNELGKDKF
EIVTPSESILAEPTVSVVDKVVEKKDTKAVAEAYLKYLYSPEGQEIAAKNFY
RPRDADVAKKYDDAFPKLKLFTIDEVFGGWAKAQKDHFADGGTFDQISKR
```

- Entscheidend für Funktionsweise ist *dreidimensionale* Struktur ($\rightarrow$ *Proteinfaltung*)

Protein Threading

- **Biologischer Hintergrund**
  - Proteine
  - **Protein Threading**

Protein Threading · Alternativen

- Physikalische Bestimmung der dreidimensionalen Struktur durch:
  - Röntgen-Strukturanalyse
  - NMR-Spektroskopie
- Beide Verfahren sind sehr zeitaufwendig, daher Simulation mit Computern
- Direkte Ansätze simulieren Gesetze der (Quanten-) Mechanik; sind jedoch nur für kleine Proteine praktikabel

Protein Threading · Grundlagen

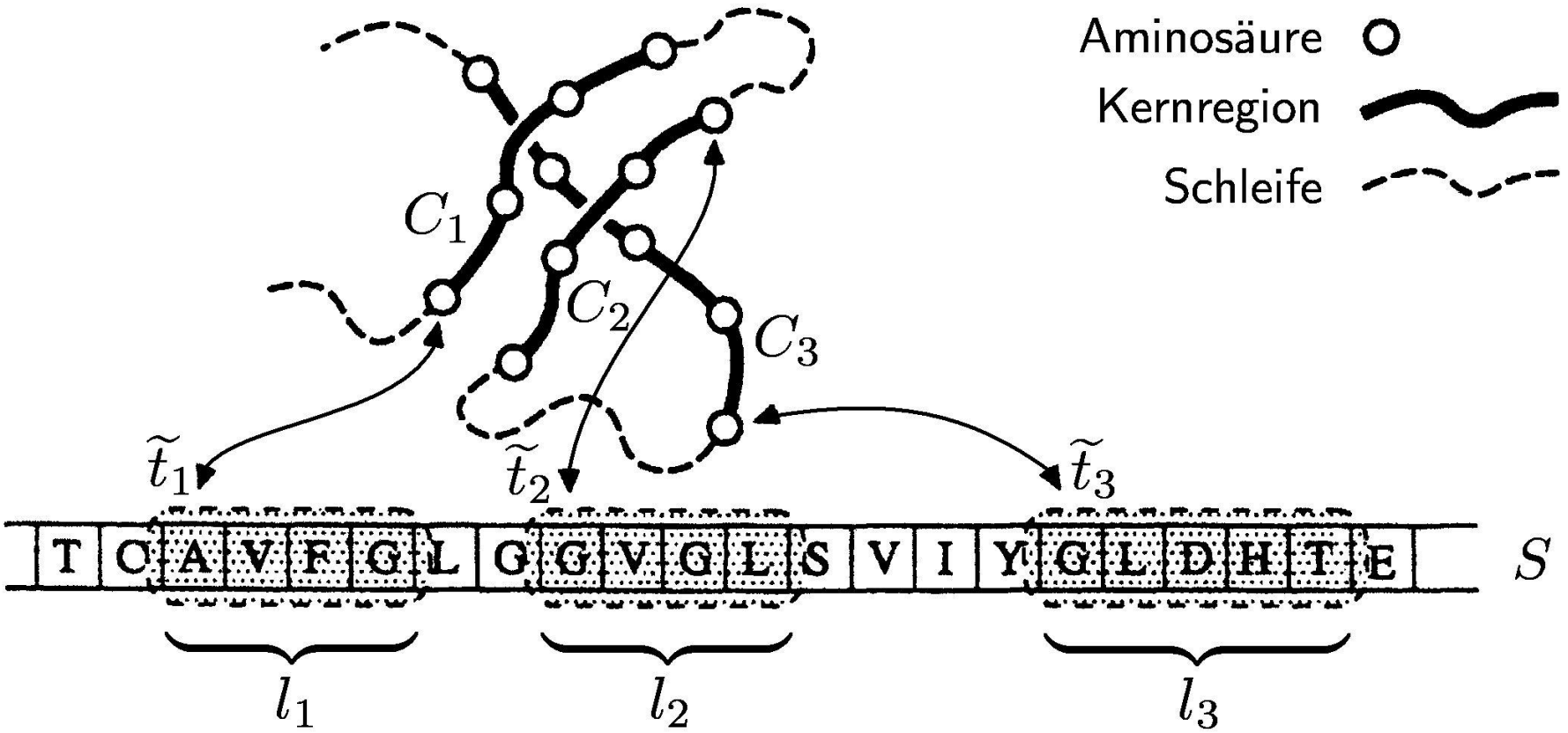
- Innerhalb der Amino-Sequenzen haben sich zwei Arten von *Kernregionen* herausgebildet:
- α-*Helices*: 5 bis 40 benachbarte Aminosäuren verbinden sich zu Spiralstruktur
- β-*Blätter*: Mehrere entfernte β-*Stränge* verknüpfen sich zu einer Faltblattstruktur

- Kernregionen sind durch *Schleifen* verbunden
- Gesamtheit aller Kernregionen eines Proteins heißt *Kern*

Protein Threading · Idee

- Aus bekannten Strukturen können statistische Daten über das Vorkommen der 20 verschiedenen Aminosäuren in den beiden Kernregionen abgeleitet werden
- Auch chemische Eigenschaften (z. B. Polarität) werden berücksichtigt
- Ist der Kern bekannt, so kann man eine *Ausrichtung* der Amino-Sequenz finden, die diese Vorgaben *am wahrscheinlichsten* erfüllt.
- Da die Sequenz nicht zerrissen werden darf, spricht man vom *Einfädeln* eines Proteins in eine *Kernschablone* – *Protein Threading*

Protein Threading · Visualisierung



Protein Threading · Idee

- Problem: Wie bestimmt man den Kern eines Proteins?
- Es gibt mehrere Millionen Proteine, aber nur 1 000 bis 8 000 verschiedene Kerne
- Wenn man alle diese Kernschablonen kennt, kann man wie folgt vorgehen:
  1. Bestimme zu jeder Schablone die wahrscheinlichste Ausrichtung der Amino-Sequenz
  2. Wähle aus diesen Ausrichtungen die „beste“ aus; diese entspricht der dreidimensionalen Struktur des Proteins

Modellierung

- Biologischer Hintergrund
- ► **Modellierung**
- MIP-Formulierungen
- Zusätzliche Lösungsstrategien
- Zusammenfassung
- Literatur

Formale Problembeschreibung

- **Modellierung**
  - **Formale Problembeschreibung**
    - Netzwerk-Fluss-Modell
    - Bemerkungen zur Zielfunktion
    - Komplexitätsbetrachtungen

Formale Problembeschreibung · Bezeichnungen

- $C := (C_1, C_2, \ldots, C_m)$ sei eine Kernschablone mit $m \in \mathbb{N}$ Kernregionen (α-Helix oder β-Strang); jede Kernregion hat dabei die Länge $l_i \in \mathbb{N}$, $i = 1, \ldots, m$
- Σ bezeichne das 20 Zeichen umfassende Amino-Alphabet, also $|\Sigma| = 20$
- $S \in \Sigma^{\widetilde{n}}$ stehe für die einzufädelnde, $\widetilde{n} \in \mathbb{N}$ Säuren lange Amino-Sequenz
- $\widetilde{t}_i \in \mathbb{N}$, $\widetilde{t}_i \leq \widetilde{n}$, für $i = 1, \ldots, m$ sei die absolute Position innerhalb der Amino-Sequenz, an der für eine bestimmte Ausrichtung die Kernregion C_i beginnt.

Formale Problembeschreibung · Gültige Ausrichtung

- Kernregion C_i belegt dabei die Positionen $\widetilde{t}_i, \ldots, \widetilde{t}_{i+l_i-1}$; Lücken oder Einfügungen innerhalb der Kernregionen sind also ausgeschlossen
- Eine Ausrichtung der Amino-Sequenz S an einer Kernschablone C heißt *gültig*, falls
  1. $1 \leq t_1, t_m \leq \widetilde{n} + 1 - l_m$ und
  2. $t_i + l_i \leq t_{i+1}$ für $i = 1, \ldots, m-1$
- Für eine *gültige Ausrichtung* wird also gefordert, dass sich die Kernregionen nicht überlappen und ihre Reihenfolge erhalten bleibt

Formale Problembeschreibung · Relative Positionen

- Wir führen *relative* Positionen ein:

$$t_i := \widetilde{t}_i - \sum_{j=1}^{i-1} l_i$$

- Weiter bezeichne

$$n := \widetilde{n} + 1 - \sum_{i=1}^{m} l_i$$

- Eine Ausrichtung ist genau dann gültig, falls gilt

$$1 \leq t_1 \leq t_2 \leq \ldots \leq t_m \leq n$$

Formale Problembeschreibung · Zielfunktion

- Die *Zielfunktion* soll zwei Informationen liefern:
  1. $c_{ik} \in \mathbb{R}$ für $i = 1, \ldots, m$ und $k = 1, \ldots, n$ sei eine Bewertung (je kleiner umso besser) dafür, dass Kernregion C_i an die k. relative Position platziert wird
  2. c_{ikjl} für $(i, j) \in L$ und $1 \leq k \leq l \leq n$ bewerte die *paarweisen Interaktionen* zweier Kernregionen, falls Kernregion C_i an der k. und Kernregion C_j an der l. relativen Stelle positioniert wird

     $L \subseteq \{(i, j) \mid 1 \leq i < j \leq m\}$ sei die Menge aller Kernregion-Paare, zwischen denen eine Interaktion besteht – etwa auf Grund räumlicher Nähe

Formale Problembeschreibung · PTP

- Das *Protein Threading Problem (PTP)* lautet nun:

$$\min_{t_1,t_2,\ldots,t_m} \left\{ \sum_{i=1}^{m} c_{it_i} + \sum_{(i,j)\in L} c_{it_ijt_j} \;\middle|\; 1 \leq t_1 \leq t_2 \leq \ldots \leq t_m \leq n \right\}$$

- D. h. finde eine gültige Ausrichtung mit minimalen Kosten

Netzwerk-Fluss-Modell

- **Modellierung**
  - Formale Problembeschreibung
  - **Netzwerk-Fluss-Modell**
  - Bemerkungen zur Zielfunktion
  - Komplexitätsbetrachtungen

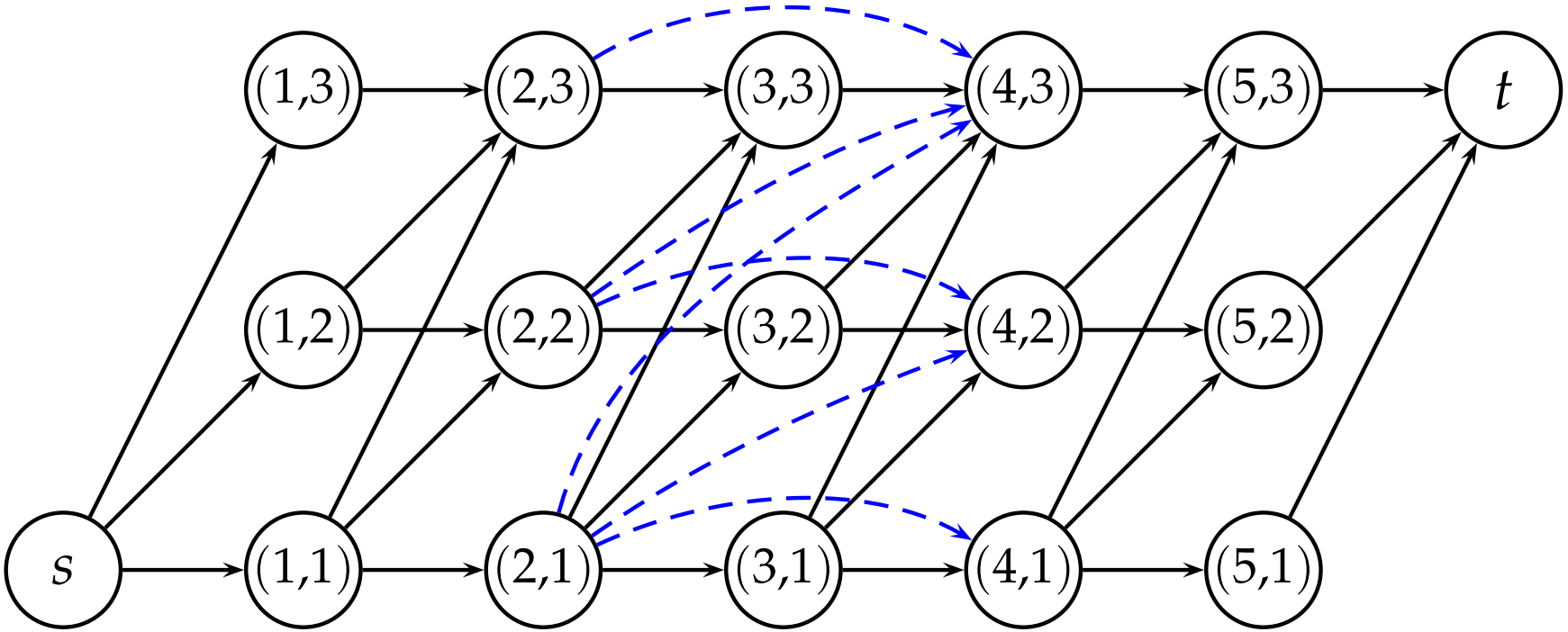
Netzwerk-Fluss-Modell · Netzwerk

- Wir konstruieren den gerichteten Graphen $G = (V, E)$:

$$
\begin{aligned}
V &:= \left\{(i,k) \;\middle|\; i = 1,\ldots,m \;\wedge\; k = 1,\ldots,n\right\} \cup \{s,t\} \\
E_L &:= \left\{\big((i,k),(j,l)\big) \;\middle|\; (i,j) \in L \;\wedge\; 1 \le k \le l \le n\right\} \\
E_x &:= \bigcup_{1\le k\le l\le n} \left\{\big((i,k),(i+1,l)\big) \;\middle|\; i = 1,\ldots,m-1\right\} \\
&\quad \cup \left\{\big(s,(1,k)\big) \;\middle|\; k = 1,\ldots,n\right\} \cup \left\{\big((m,k),t\big) \;\middle|\; k = 1,\ldots,n\right\} \\
E &:= E_L \cup E_x
\end{aligned}
$$

- Knoten (i,k) repräsentiert das Positionieren der Kernregion C_i an die k. relative Position
- Die Kanten E_L stehen für die paarweisen Interaktionen

Netzwerk-Fluss-Modell · Beispielnetzwerk



- E_x durchgehend, $E_z := E_L \setminus E_x$ gestrichelt

Netzwerk-Fluss-Modell · (s,t)-Wege

- (s,t)-Wege im Graphen $G_x := (V, E_x)$ entsprechen genau den gültigen Ausrichtungen: Dem Weg $\big(s,(1,t_1),(2,t_2),\ldots,(m,t_m),t\big)$ entspricht die Ausrichtung der Kernregionen an den relativen Positionen $t_1, t_2, \ldots, t_m$ und umgekehrt
- Weiter stehe $E_z := E_L \setminus E_x$ für die Interaktionen *nichtbenachbarter* Kernregionen
- Ein (s,t)-Weg *aktiviert* die Kante $\big((i,k),(j,l)\big) \in E_z$, falls sowohl (i,k) als auch (j,l) auf diesem Weg liegen
- Ein (s,t)-Weg zusammen mit den von ihm aktivierten Kanten wird *augmentierter Weg* genannt

Netzwerk-Fluss-Modell · Kantengewichte

- Das PTP besteht im Auffinden des kürzesten augmentierten Weges
- Die Kanten $\big(s,(1,k)\big) \in E_x$ erhalten dabei die Kosten 0
- Die Kanten $\big((m,k),t\big) \in E_x$ erhalten die Kosten c_{mk}
- Die Kosten der Kanten $\big((i,k),(j,l)\big) \in E_x$ sind jeweils die Summe aus
  - der Bewertung c_{ik} und
  - der Bewertung c_{ikjl} (0 falls $(i,j) \notin L$)
- Die Kosten der Kanten $\big((i,k),(j,l)\big) \in E_z$ entsprechen den Bewertungen c_{ijkl}

Bemerkungen zur Zielfunktion

- **Modellierung**
  - Formale Problembeschreibung
  - Netzwerk-Fluss-Modell
  - **Bemerkungen zur Zielfunktion**
  - Komplexitätsbetrachtungen

Bemerkungen zur Zielfunktion · Koeffizienten

- Zum Ermitteln der Zielfunktion werden (statistische) Informationen über einzelne Aminosäuren benutzt
- Für eine *feste* Sequenz erhält man die Zielfunktions-Koeffizienten, indem man für jede Kernregion die entsprechenden Werte der beteiligten Aminosäuren aufsummiert
- Dies wird *einmal* erledigt und die Daten für alle möglichen Positionen der Kernregionen in einem großen Array gespeichert
- Nur so kann in konstanter Zeit auf sie zugegriffen werden

Bemerkungen zur Zielfunktion · Normalisierung

- Unsere Verfahren liefern nur eine optimale Ausrichtung einer Amino-Sequenz an *einer* Kernschablone
- Um die tatsächliche Struktur des Proteins zu erhalten, werden die Verfahren auf *jede* bekannte Kernschablone angewendet
- Vergleich der Zielfunktionswerte liefert dann die wahrscheinlichste Kernstruktur (die mit niedrigster Bewertung)
- Aber allein die Anzahl der Kernregionen einer Kernschablone verändert den Zielfunktionswert

Bemerkungen zur Zielfunktion · Selbsteinfädeln

- Daher müssen die Bewertungen vorher *normalisiert* werden
- Eine Möglichkeit zum Eichen von Zielfunktion und Normalisierung ist das *Selbsteinfädeln (self-threading)*
- Dazu nimmt man den Kern einer bekannten Proteinstruktur als Schablone und richtet die zugehörige Amino-Sequenz an ihm aus
- Dabei sollte sich natürlich (normalisiert) ein niedrigerer Zielfunktionswert als bei allen anderen Schablonen ergeben

Komplexitätsbetrachtungen

- **Modellierung**
  - Formale Problembeschreibung
  - Netzwerk-Fluss-Modell
  - Bemerkungen zur Zielfunktion
  - **Komplexitätsbetrachtungen**

Komplexitätsbetrachtungen · Varianten des PTP

- Wir lassen die paarweisen Interaktionen nichtbenachbarter Kernregioen außer Acht:
  - Dann besteht das PTP im Auffinden eines kürzesten (s,t)-Weges in G_x
  - Dies kann mittels topologischer Sortierung der Knoten in $\mathcal{O}(E_x) = \mathcal{O}(mn^2)$ erfolgen
- *Polynomiale* Laufzeit erhält man auch bei festen Schleifenlängen
- Diese Vereinfachungen sind biologisch wenig gerechtfertigt

Komplexitätsbetrachtungen · Unser PTP

- Das PTP in der von uns betrachteten Form ist $\mathcal{NP}$-schwer (selbst bei Außerachtlassung der Codierungslänge der verwendeten Zahlen)
- Das zugehörige Entscheidungsproblem kann zum Lösen von ONE-IN-THREE 3SAT – einer Variante des Erfüllbarkeitsproblems – benutzt werden und ist daher $\mathcal{NP}$-vollständig
- Außerdem ist das PTP $\mathcal{MAX}$-$\mathcal{SNP}$-schwer
- D. h. die Lösung kann in polynomialer Zeit nicht beliebig genau approximiert werden (falls $\mathcal{P} \neq \mathcal{NP}$)

MIP-Formulierungen

- Biologischer Hintergrund
- Modellierung
- ► **MIP-Formulierungen**
- Zusätzliche Lösungsstrategien
- Zusammenfassung
- Literatur

Kürzeste-Wege-Formulierungen

- **MIP-Formulierungen**
  - **Kürzeste-Wege-Formulierungen**
  - Nichtlineare Formulierung
  - MXZ-Formulierung
  - MXYZ-Formulierung
  - MYZ-Formulierung
  - Praktischer Vergleich

Kürzeste-Wege-Formulierungen · Polytop X

- **Polytop X:**

$$\begin{aligned}
\sum_{e \in \Gamma(s)} x_e &= 1 \\
\sum_{e \in \Gamma^{-1}(t)} x_e &= 1 \\
\sum_{e \in \Gamma(v)} x_e - \sum_{e \in \Gamma^{-1}(v)} x_e &= 0 \quad \forall\, v \in V \setminus \{s,t\} \\
x_e &\geq 0 \quad \forall\, x_e \in E_x
\end{aligned}$$

- Ecken von X sind ganzzahlig und entsprechen genau den (s,t)-Wegen

Kürzeste-Wege-Formulierungen · Menge Υ

- **Menge ganzzahliger Punkte Υ:**

$$\begin{aligned}
\sum_{k=1}^{n} y_{ik} &= 1 \quad \forall\, i = 1, \ldots, m \\
\sum_{l=1}^{k} y_{il} - \sum_{l=1}^{k} y_{i+1,l} &\geq 0 \quad \forall\, i = 1, \ldots, m-1 \;\wedge \\
& \qquad\qquad k = 1, \ldots, n-1 \\
y_{ik} &\in \{0,1\} \quad \forall\, i = 1, \ldots, m \;\wedge \\
& \qquad\qquad k = 1, \ldots, n
\end{aligned}$$

- $y \in \Upsilon$ entsprechen genau den (s,t)-Wegen

Nichtlineare Formulierung

- **MIP-Formulierungen**
  - Kürzeste-Wege-Formulierungen
  - **Nichtlineare Formulierung**
  - MXZ-Formulierung
  - MXYZ-Formulierung
  - MYZ-Formulierung
  - Praktischer Vergleich

Nichtlineare Formulierung · und Linearisierung

- Eine naheliegende PTP-Formulierung ist

$$\begin{aligned} \min_{y} \quad & \sum_{i=1}^{m}\sum_{k=1}^{n} c_{ik}y_{ik} + \sum_{(i,j)\in L}\sum_{k=1}^{n}\sum_{l=k}^{n} c_{ik}y_{ik}y_{jl} \\ s.t. \quad & y \in Y \end{aligned}$$

- Linearisierung:
  - Ersetze $y_{ik}y_{jl} = \min\{y_{ik}, y_{jl}\}$ durch z_{ikjl}
  - $z_{ikjl} \leq y_{ik}$, $z_{ikjl} \leq y_{jl}$, $z_{ikjl} \geq y_{ik} + y_{jl} - 1$, $0 \leq z_{ikjl} \leq 1$
- In der Praxis ungeeignet, da LP-Relaxierung nur schwache Schranken liefert

MXZ-Formulierung

- **MIP-Formulierungen**
  - Kürzeste-Wege-Formulierungen
  - Nichtlineare Formulierung
  - **MXZ-Formulierung**
  - MXYZ-Formulierung
  - MYZ-Formulierung
  - Praktischer Vergleich

MXZ-Formulierung · Bezeichnungen

- $$A := \{(i,j) \in L \mid j = i+1\}$$

  sei die Menge von *benachbarten* Kernregionen, zwischen denen eine paarweise Interaktion besteht

- $$R := L \setminus A$$

  bezeichne entsprechend die Menge aller *nicht-benachbarten* Kernregionen, zwischen denen eine paarweise Interaktion besteht

- $$R_C := \{i \mid (i,j) \in R\} \cup \{j \mid (i,j) \in R\}$$

  stehe für die Kernregionen, die an *nichtbenachbarten*, paarweisen Interaktionen beteiligt sind

MXZ-Formulierung · mittels Polytop X

- Direkte Umsetzung des Netzwerk-Fluss-Models:

$$\min_{x,z} \quad \sum_{e \in E_x} c_e x_e + \sum_{e \in E_z} c_e z_e$$

$$s.t. \quad \sum_{1 \le k \le l \le n} z_{ikjl} = 1 \qquad \forall (i,j) \in R$$

$$z_{ikjl} \le \sum_{e \in \Gamma(i,k)} x_e \qquad \forall \big((i,k),(j,l)\big) \in E_z$$

$$z_{ikjl} \le \sum_{e \in \Gamma^{-1}(j,l)} x_e \qquad \forall \big((i,k),(j,l)\big) \in E_z$$

$$x \in X$$

$$z_e \in \{0,1\} \qquad \forall e \in E_z$$

MXZ-Formulierung · Ganzzahligkeit

- Aus der Ganzzahligkeit der x-Variablen folgt die der z-Variablen und umgekehrt
- Seien N_x bzw. N_z die Anzahl der möglichen Festlegungen von x bzw. z, dann gilt

$$N_z = \binom{|R_C| + n - 1}{|R_C|} \leq \binom{m + n - 1}{m} = N_x$$

- Der Suchraum der z-Variablen ist folglich nicht größer als der der x-Variablen
- Daher fordert man besser die Ganzzahligkeit von z

MXYZ-Formulierung

- **MIP-Formulierungen**
  - Kürzeste-Wege-Formulierungen
  - Nichtlineare Formulierung
  - MXZ-Formulierung
  - **MXYZ-Formulierung**
  - MYZ-Formulierung
  - Praktischer Vergleich

MXYZ-Formulierung · mittels Polytop X

- Hinzufügen von y-Variablen für die Knoten:

$$\min_{x,y,z} \quad \sum_{e \in E_x} c_e x_e + \sum_{e \in E_z} c_e z_e$$

$$s.t. \quad \sum_{k=1}^{n} y_{ik} = 1 \qquad \forall\, i \in R_C$$

$$y_{ik} = \sum_{e \in \Gamma(i,k)} x_e \qquad \forall\, i \in R_C \wedge k = 1, \ldots, n$$

$$y_{ik} = \sum_{l=k}^{n} z_{ikjl} \qquad \forall\, (i,j) \in R \wedge k = 1, \ldots, n$$

$$y_{jl} = \sum_{k=1}^{l} z_{ikjl} \qquad \forall\, (i,j) \in R \wedge l = 1, \ldots, n$$

MXYZ-Formulierung · Verbesserungen

$$y_{ik} \in \{0,1\} \quad \forall\, i \in R_C \wedge k = 1,\ldots,n$$

$$x \in X$$

$$z_e \geq 0 \quad \forall\, e \in E_z$$

- Verbesserungen gegenüber der MXZ-Formulierung:
  - Die LP-Relaxierung liegt näher an der konvexen Hülle aller ganzzahligen Punkte
  - Bei gleicher Größe des Suchraums verringert sich die Zahl der Binärvariablen von $|R|\, n(n+1)$ auf $|R_C|\, n$ (wobei $|R_C| \leq 2\,|R|$)

MYZ-Formulierung

- **MIP-Formulierungen**
  - Kürzeste-Wege-Formulierungen
  - Nichtlineare Formulierung
  - MXZ-Formulierung
  - MXYZ-Formulierung
  - **MYZ-Formulierung**
  - Praktischer Vergleich

MYZ-Formulierung · mittels Menge Υ

- Wir führen nun die y-Variablen für *alle* Knoten ein und verzichten auf die x-Variablen:

$$\min_{y,z} \quad \sum_{i=1}^{m}\sum_{k=1}^{n} c_{ik}y_{ik} + \sum_{e\in E_L} c_e z_e$$

$$s.t. \quad y_{ik} = \sum_{l=k}^{n} z_{ikjl} \quad \forall\,(i,j) \in L \wedge k = 1,\ldots,n$$

$$y_{jl} = \sum_{k=1}^{l} z_{ikjl} \quad \forall\,(i,j) \in L \wedge l = 1,\ldots,n$$

$$y \in \Upsilon$$

$$z_e \geq 0 \quad \forall\, e \in E_L$$

Praktischer Vergleich

- **MIP-Formulierungen**
  - Kürzeste-Wege-Formulierungen
  - Nichtlineare Formulierung
  - MXZ-Formulierung
  - MXYZ-Formulierung
  - MYZ-Formulierung
  - **Praktischer Vergleich**

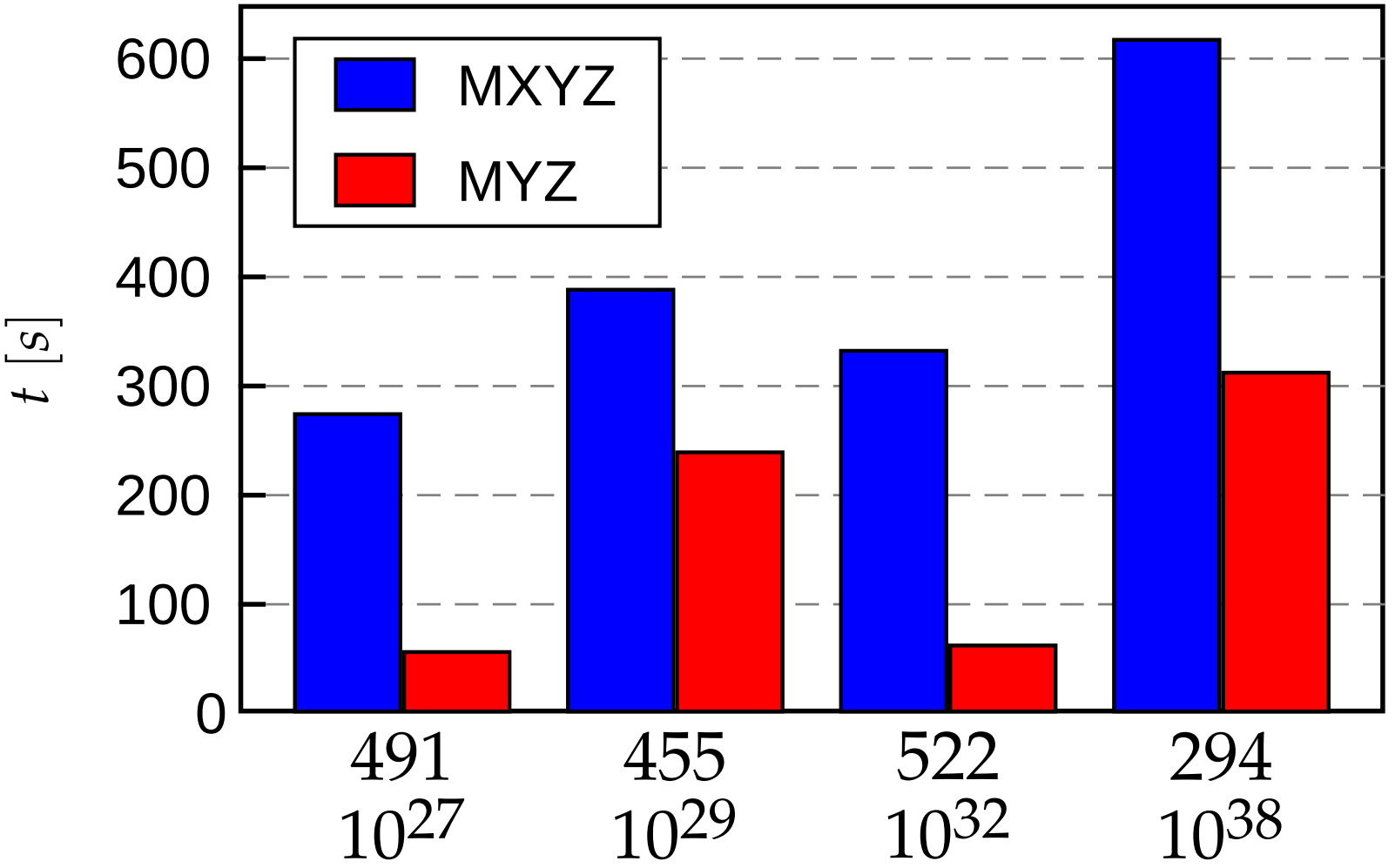
Praktischer Vergleich · MXYZ und MYZ

- Nichtlineare und MXZ-Formulierung bei kleinen Testproblemen teilweise drei bis vier Größenordnungen langsamer als die MXYZ-Formulierung
- Daher werden nur die MXYZ- und die MYZ-Formulierung verglichen
- Vorweg: MYZ ist deutlich schneller
- Bei beiden Formulierung zeigt sich ein überraschendes Resultat: In ca. 95% der Fälle liefert die LP-Relaxierung eine *ganzzahlige* Lösung!

Praktischer Vergleich · Ganzzahligkeit des LP

- Mit anderen Worten: Sehr oft liefert bereits die LP-Relaxierung eine *optimale* Lösung
- Dieser Effekt tritt nur bei realen Zielfunktionen auf
- Werden die Koeffizienten der Zielfunktion zufällig gewählt, zeigt sich die $\mathcal{NP}$-Härte des PTP
- Auch beim Zusammenfassen von Gleichungen geht die Ganzzahligkeit verloren
- Ähnliches Verhalten wie unkapazitiertes Standortproblem (UFLP)

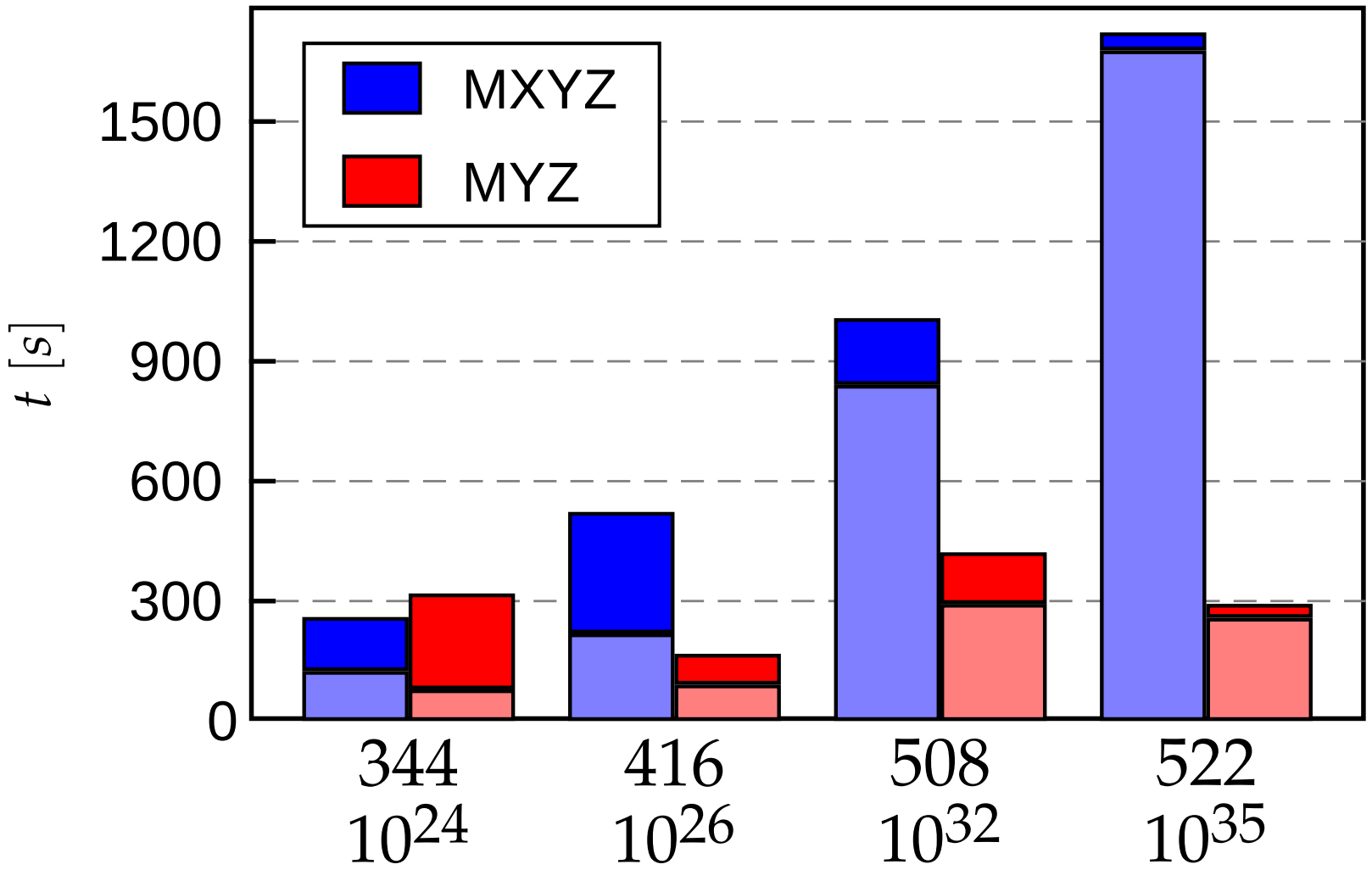
Praktischer Vergleich · LPs mit ganzzahliger Lösung



Praktischer Vergleich · LPs mit gebrochener Lösung

- Im Falle einer gebrochenen Lösung der LP-Relaxierung wurde das MIP mit einem Branch&Bound-Ansatz gelöst:
  - Verzweigen an Variablen mit Wert $0{,}5$
  - Nie mehr als elf B&B-Knoten, oft nur zwei
  - LP-Relaxierung macht Großteil der Laufzeit aus
- Der Zielfunktionswert der LP-Relaxierung liegt nur knapp unter dem der optimalen Ausrichtung; ausreichend nahe für den Vergleich mehrerer Kernschablonen
- Die Menge der real vorkommenden PTP ist also in „*polynomialer*“ Zeit lösbar

Praktischer Vergleich · LPs mit gebrochener Lösung



Zusätzliche Lösungsstrategien

- Biologischer Hintergrund
- Modellierung
- MIP-Formulierungen
- ► **Zusätzliche Lösungsstrategien**
- Zusammenfassung
- Literatur

Divide et Impera

- **Zusätzliche Lösungsstrategien**
  - **Divide et Impera**

    Parallelisierung

Divide et Impera · Motivation

- LP-Relaxierung liefert zwar sehr gute Ergebnisse, manchmal kann sie jedoch wegen der enormen Problemgröße nicht gelöst werden
- Beispiel mit $\widetilde{n} = 508$, $n = 81$, $m = 36$:
  LP-Relaxierung umfasst 741 264 Zeilen, 360 945 Spalten und 54 145 231 Nichtnull-Einträge
- Ziele der Divide&Conquer-Methode:
  - Ermöglichen der Lösung von großen Problemen
  - Beschleunigen der Lösung von kleinen bis mittelgroßen Problemen

Divide et Impera · Aufteilen

- Aufteilung durch Vorgabe von Schranken an die möglichen Positionen *einer* Kernregion C_i:

$$1 \leq L_i \leq t_i \leq U_i \leq n$$

- Dazu werden in der MIP-Fomulierung (hier MYZ) gewisse Variablen auf 0 fixiert:

$$\begin{aligned} y_{ik} &:= 0 \quad \forall\, k \notin \{L_i, \ldots, U_i\} \\ z_{ikjl} &:= 0 \quad \forall\, (i,j) \in L \wedge k \notin \{L_i, \ldots, U_i\} \end{aligned}$$

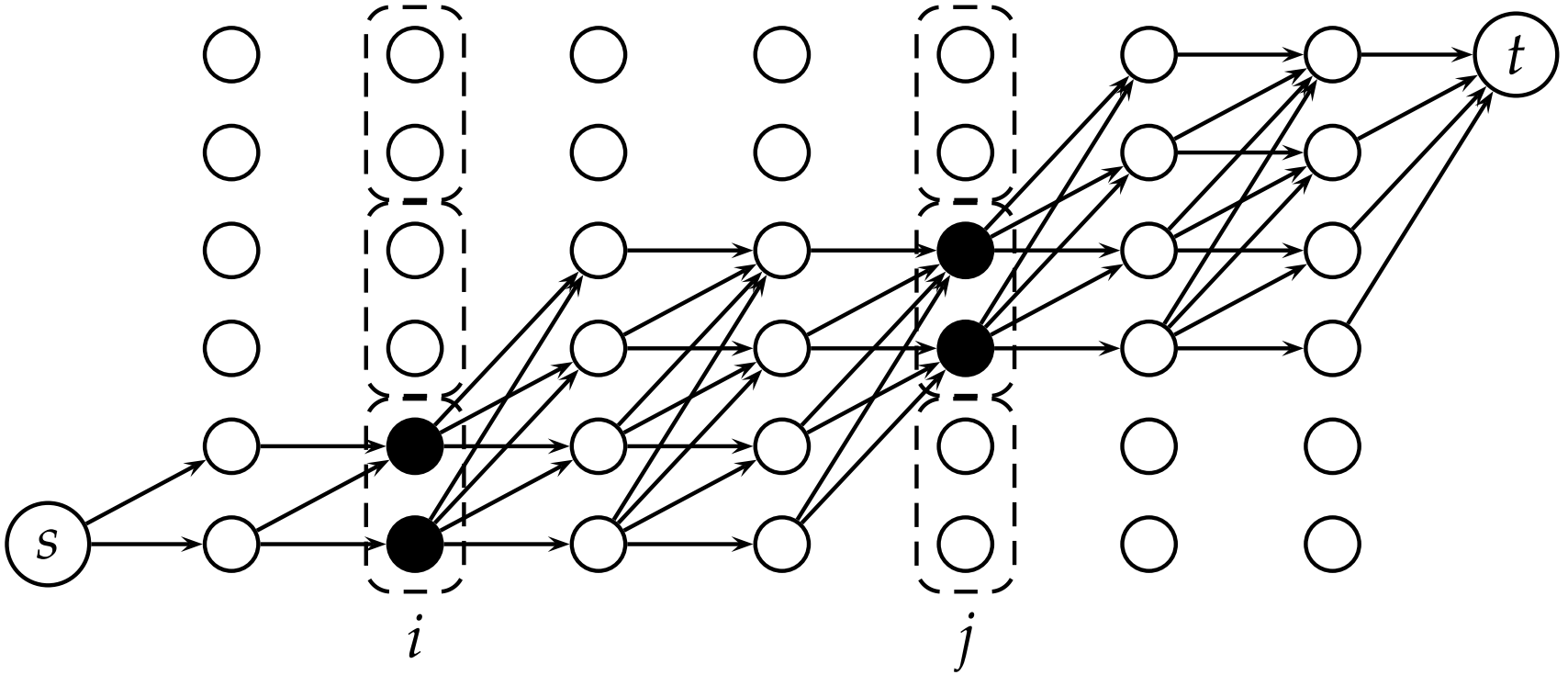
- Aufteilen in q Teilprobleme durch Schranken L_i^α, U_i^α für $\alpha = 1, \ldots, q$ mit

$$\{1, \ldots, n\} = \bigcup_{1 \leq \alpha \leq q} \{L_i^\alpha, \ldots, U_i^\alpha\}$$

Divide et Impera · Beschleunigen

- Kleinster Wert aller Zielfunktionsminima der Teilprobleme entspricht dann dem minimalen Zielfunktionswert des Originalproblems
- Verallgemeinerung auf mehrere Kernregionen möglich
- Beschleunigung der Problemlösung:
  - Beim Lösen von Teilproblemen erhält man vorläufige Minimalwerte der Zielfunktion
  - Die folgenden Teilprobleme können *vorzeitig abgebrochen* werden, wenn dieser Wert überschritten wird (dualer Simplex)

Divide et Impera · Aufteilen an zwei Kernregionen



- Äquidistante Aufteilung an den Kernregionen C_i und C_j

Divide et Impera · Wieviele Teilprobleme?

- Wieviele Teilprobleme sollen erzeugt werden?
  - Je mehr, desto einfacher werden die einzelnen Teilprobleme
  - Je weniger, desto wahrscheinlicher ist ein frühes Auffinden der Optimallösung und das Abbrechen vieler Teilprobleme
  - Beantwortung durch praktische Tests
- Wie werden die „Intervalle“ $\{L_i^\alpha, \ldots, U_i^\alpha\}$ gewählt?
  - Einfachheitshalber disjunkt und äquidistant

Divide et Impera · Welche Kernregionen?

- An wievielen Kernregionen wird aufgespalten?
  - SPLIT1: Eine Kernregion mit q Teilproblemen
  - SPLIT2: Zwei Kernregionen mit $\frac{q}{2}(q+1)$ Teilproblemen
- An welchen Kernregionen wird aufgespalten?
  - Teile so auf, dass schwierigstes Teilproblem möglichst leicht wird
  - Sei $\nu_{j\alpha}$ die Anzahl der Variablen des α. Teilproblems beim Aufteilen an Kernregion C_j. Dann wähle C_i mit

$$i = \operatorname*{argmin}_{1 \le j \le m} \left\{ \max_{1 \le \alpha \le q} \nu_{j\alpha} \right\}$$

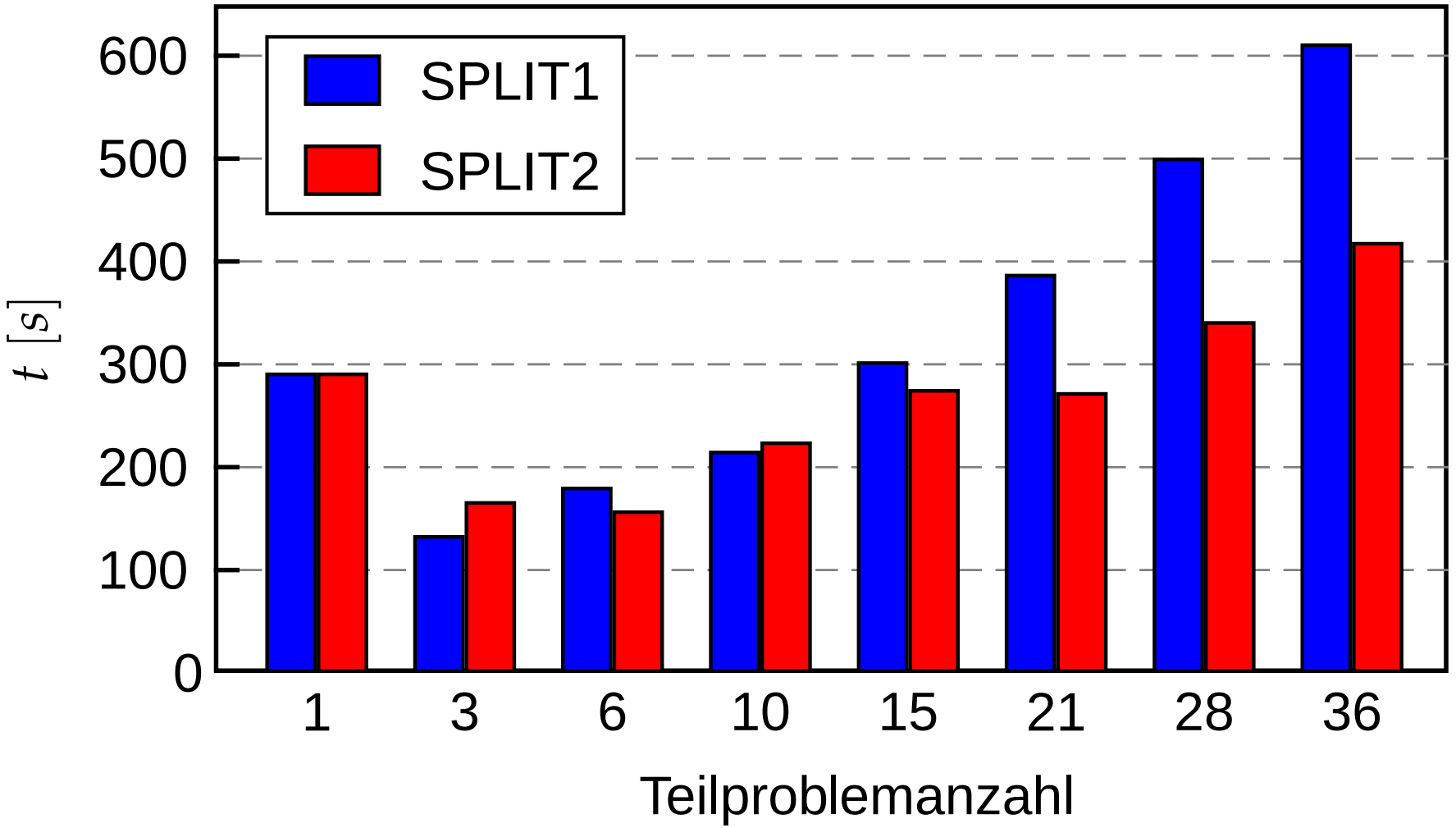
Divide et Impera · Welche Reihenfolge?

- In welcher Reihenfolge werden die Teilprobleme gelöst?
  - Sortiere Teilprobleme absteigend nach Größe und beginne mit dem größten
  - Anzahl der (s,t)-Wege durch Knoten (i,k) ist gegeben durch

$$\binom{i+k-2}{i-1}\binom{m-i+n-k}{m-i}$$

  - Aufsummieren ergibt Größe des Suchraums

Divide et Impera · Vergleich der SPLIT-Varianten



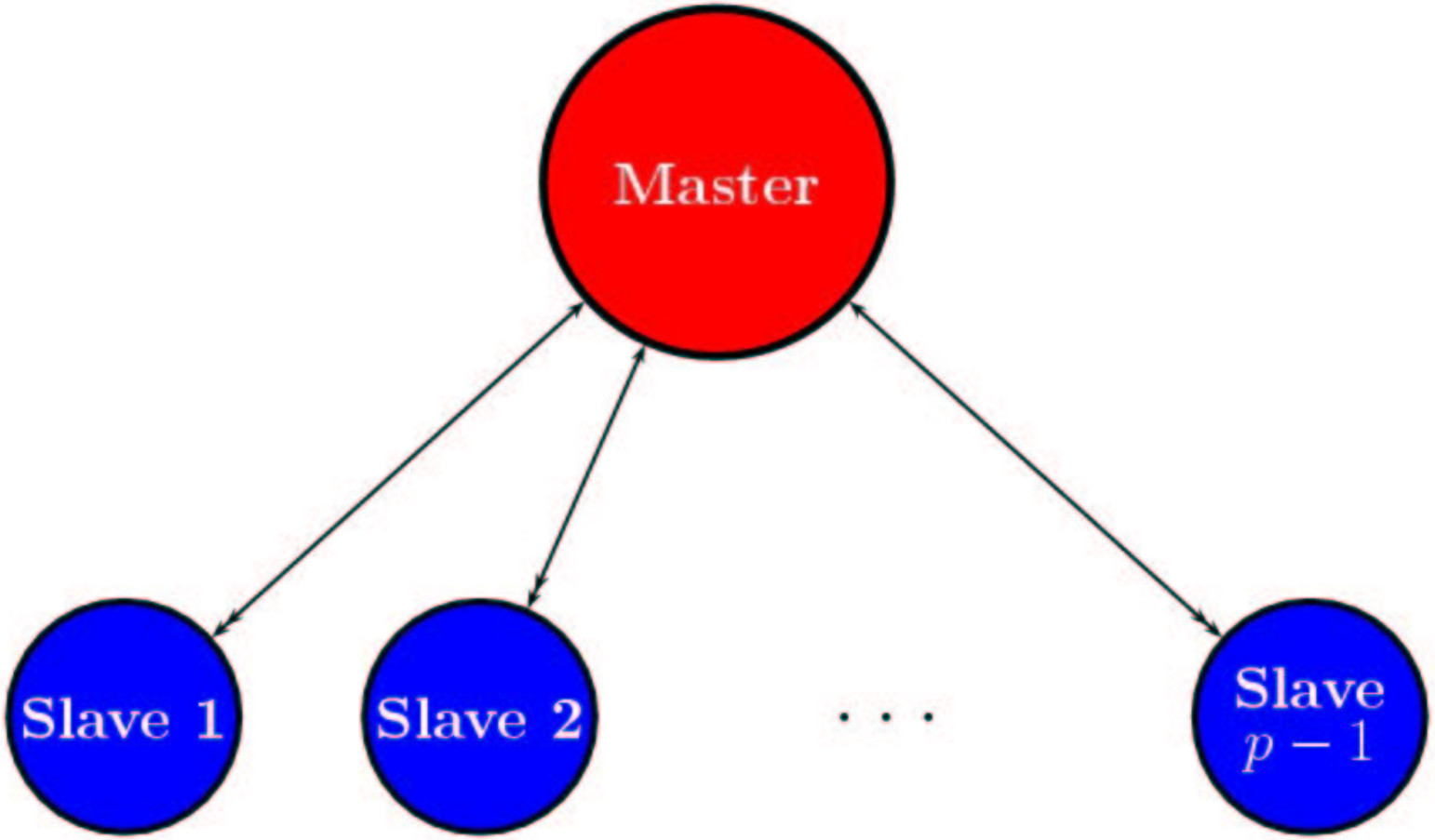
Parallelisierung

- **Zusätzliche Lösungsstrategien**
  - Divide et Impera
  - **Parallelisierung**

Parallelisierung · Parallelisierungsansatz

- Ausführen der Teilprobleme auf $p > 1$ verschiedenen Prozessoren
- Laufzeit der Probleme kann stark variieren und ist kaum vorhersagbar
- Daher *dynamische Lastverteilung*
- Am einfachsten ist dies mit einem *Master/Slave*-Ansatz zu realisieren

Parallelisierung · Master/Slave-Prinzip



Parallelisierung · Kommunikation

- Der Master verteilt Probleme an nichtbeschäftigte Slaves
- Sobald ein Slave seine *lokale* obere Schranke verbessert, teilt er sie dem Master mit
- Der Master aktualisiert ggf. die *globale* obere Schranke
- Alle Slaves fragen in regelmäßigen Abständen (z. B. alle 500-1000 Simplex-Iterationen) beim Master die aktuelle globale obere Schranke ab
- Diese wird dann als vorzeitiges Abbruchkriterium benutzt
- Ohne zwischenzeitliches Abfragen könnte ein einziger Prozessor mit einem sehr aufwendigen Teilproblem die Rechenzeit dominieren

Parallelisierung · Praktische Tests

- Genaue Vorhersagen der optimalen Kombination von Teilproblem- und Prozessorzahl sind schwierig weil problemabhängig
- Günstige Wahl von Teilproblem- und Prozessorzahl lässt Effizienz teilweise über 1 (bis zu 1,7) steigen
- Steigende Prozessorzahlen verringern deutlich den Einfluss der Teilproblemanzahlen auf die Laufzeiten
- Variante SPLIT2 ist noch überlegener als im sequentiellen Fall und erreicht selbst bei zwölf Prozessoren noch Effizienzwerte zwischen 0,5 und 1

Parallelisierung · Effizienz größer 1?

- Erinnerung:

$$\text{Speedup} = \frac{\text{Laufzeit eines Prozessors im sequentiellen Fall}}{\text{Laufzeit mit } p \text{ Prozessoren im parallelen Fall}}$$

$$\text{Effizienz} = \frac{\text{Laufzeit eines Prozessors im sequentiellen Fall}}{\text{Laufzeit eines Prozessors im parallelen Fall}}$$

- Durch Austausch der aktuellen globalen oberen Schranke können ggf. manche großen Teilprobleme frühzeitig abgebrochen werden
- Dadurch kann jeder einzelne Parallelrechner bei gleichem Algorithmus schneller als im sequentiellen Betrieb sein

Zusammenfassung

- Biologischer Hintergrund
- Modellierung
- MIP-Formulierungen
- Zusätzliche Lösungsstrategien
- ► **Zusammenfassung**
- Literatur

Praktische Gesichtspunkte

- Proteinfaltung ist der „heilige Gral“ der modernen Molekularbiologie; Protein Threading bietet eine Lösungsmethode dafür
- Ausgehend vom Netzwerk-Fluss-Modell wurden MIP-Formulierung vorgestellt, die eine Amino-Sequenz mit de facto *polynomialem* Zeitaufwand in eine Kernschablone einfädeln können
- Angesichts der riesigen Anzahl von kombinatorischen Möglichkeiten – 10^{40} und mehr – sind Laufzeiten von wenigen Minuten bereits beachtenswert

Theoretische Gesichtspunkte

- Der Vergleich eines mittelgroßen Proteins mit einigen tausend Kernschablonen dauert dennoch etwa eine Woche
- Neben den biologischen Anwendungen ist das Problem auch theoretisch interessant
- Warum ist es trotz nachgewiesener $\mathcal{NP}$- und $\mathcal{MAX}$-$\mathcal{SNP}$-Härte in den allermeisten Fällen „leicht“ lösbar?
- Eine Antwort darauf könnte helfen, eine Vielzahl anderer Algorithmen effizienter zu gestalten

Literatur

- Biologischer Hintergrund
- Modellierung
- MIP-Formulierungen
- Zusätzliche Lösungsstrategien
- Zusammenfassung
- ► **Literatur**

Teil 1

[1] AKUTSU, T.; SATORU, M.: *On the approximation of protein threading*. Theoretical Computer Science, Vol. 210, S. 261-275 (1999)

[2] ANDONOV, R.; BALEV, S.; YANEV, N.: *Protein Threading: From Mathematical Models to Parallel Implementations*. INFORMS Journal on Computing, Vol. 16, S. 393-405 (2004)

[3] COOK, W. J.; CUNNINGHAM, W. H., PULLEYBLANK, W. R.; SCHRIJVER, A.: *Combinatorial Optimization*. Wiley, New York (1998)

[4] CORNUÉJOLS, G.; NEMHAUSER, G. L.; WOLSEY, L. A.: *The Uncapacitated Facility Location Problem*. Erschienen in (Kapitel 3): MIRCHANDANI, P.; FRANCIS, R. (EDS.): *Discrete Location Theory*. Wiley, New York (1990)

[5] KNIPPERS, R.: *Molekulare Genetik*. Georg Thieme Verlag, Stuttgart (1997)

Teil 2

[6] LATHROP, R. H.: *The Protein Threading Problem With Sequence Amino Acid Interaction Preferences Is NP-Complete*. Protein Engineering, Vol. 7, S. 1059-1068 (1994)

[7] LATHROP, R. H.; ROGERS JR., R. G.; BIENKOWSKA, J.; BRYANT, B. K. M.; BUTUROVIĆ, L. J.; GAITATZES, C.; NAMBUDRIPAD, R.; WHITE, J. V.; SMITH, T. F: *Analysis and algorithms for protein sequence-structure alignment*. Erschienen in (Kapitel 12): SALZBERG, S. L.; SEARLS, D. B.; KASIF, S. (EDS.): *Computational Methods in Molecular Biology*. Elsevier, Amsterdam (1998)

[8] LENGAUER,T.: *Computational Biology at the Beginning of the Post-genomic Era*. Erschienen in: WILHELM, R. (ED.): *Informatics—10 Years Back, 10 Years Ahead*. Lecture Notes in Computer Science, No. 2000, Springer, Heidelberg (2001)

[9] MARIN, A.; POTHIER, J.; ZIMMERMANN, K.; GIBRAT, J.-F.: *FROST: A Filter-Based Fold Recognition Method*. PROTEINS: Structure, Function and Genetics, Vol. 49, S. 493-509 (2002)

Teil 3

[10] RCSB PROTEIN DATA BANK: `http://www.rcsb.org/pdb/`.

[11] SETUBAL, J. C.; MEIDANIS, J.: *Introduction to Computational Molecular Biology*. Brooks/Cole Publishing Company, Pacific Grove, CA (1997)

[12] XU, J.; LI, M.; LIN, G.; KIM, D.; XU, Y.: *Protein Threading Problem by Linear Programming*. Proceedings of the 17th Pacific Symposium on Biocomputing (PSB), S. 264-275 (2003)

[13] YANEV, N.; ANDONOV, R.: *Solving the Protein Threading Problem in Parallel*. Workshop on HiCOMB'03, 17th IPDPS (2003)

[14] YANEV, N.; ANDONOV, R.: *The Protein Threading Problem is in P?*. Institut National de Recherche en Informatique et en Automatique, Research Report 4577 (2002)

Hinweis

- Das Handout und die Präsentationsfolien stehen unter

  `www.mathi.uni-heidelberg.de`
  `/~ferreau/proteinThreading/`

  zum Download bereit.

Fragen

Fragen?