Stabile Matchings

(Betrachte zunächst bipartite Graphen)

Gegeben: n Frauen, n Männer,
Präferenzlisten f_i und m_i

Gesucht: stabiles Matching

Was heißt "stabil"?

Angenommen $\{f_1, m_2\}$ und $\{f_2, m_1\}$ sind im Matching, aber
f_1 findet m_1 besser als m_2 **und**
m_1 findet f_1 besser als f_2

$\rightarrow$ dies ist **nicht** stabil!

<u>Definition</u>: Ein Matching ist <u>nicht stabil</u> wenn

1. Es gibt ein Element A welches ein Element B gegenüber dem Element mit dem es aktuell gematcht ist präferiert

2. Element B präferiert A gegenüber aktuellem Match ebenfalls

Also in anderen Worten:

Ein Matching ist stabil, wenn es kein Paar von Elementen gibt, bei dem sich beide gegenüber ihren aktuellen Matches präferieren...

Frage: Gibt es in einem $K_{n,n}$ immer ein stabiles Matching?

Antwort: Ja! $\to$ Nobelpreis...

Algorithmus von Gale-Shapley

Erste Runde: Jede Frau fragt den Mann
den sie am Besten findet
(der also ganz oben auf der
jeweiligen Liste steht)

Jeder Mann sagt "ja" zu
der Frau die er am Besten findet
und nein zu allen anderen

Algorithmus von Gale-Shapley

Erste Runde: Jede Frau fragt den Mann den sie am Besten findet (der also ganz oben auf der jeweiligen Liste steht)

Jeder Mann sagt "ja" zu der Frau die er am Besten findet und nein zu allen anderen

Jede weitere Runde: Jede Frau ohne "vorläufiges" Match fragt den Mann den sie am meisten präferiert und noch nicht gefragt hat

Jeder Mann sagt "ja" zu der Frau die er am meisten präferiert!

→ Jeder ist am Ende vergeben und die Matches sind stabil.

<u>Lemma 1:</u> Das vom Algorithmus berechnete Matching ist perfekt.

<u>Beweis:</u> Angenommen eine Frau f_i ist nicht gematcht. Da es genauso viele Männer wie Frauen gibt, ist ein Mann m_j nicht gematcht.

Wenn ein Mann einmal „ja" sagte, besitzt er immer ein Matching; er fragte ihn also keine Frau... aber die Frau f_i fragt jeden Mann... ↯

□

Lemma 2: Das berechnete Matching ist stabil.

Beweis: Angenommen f_i und m_j sind ein präferierendes Paar.

1. Fall: f_i fragte m_j nie

→ nach Algo fragt jede Frau in Reihenfolge ihrer Liste

→ f_i präferiert m_j vor ihrem Match m_k

→ f_i und m_j sind nicht präferierend

2. Fall: f_i fragt m_j

→ m_j hat irgendwann "nein" gesagt

→ nach Algo verlässt m_j f_i für besseren Match

→ m_j präferiert f_ℓ über f_i

→ f_i und m_j sind nicht präferierend ↯

□

→ Es kann mehr als ein stabiles Matching geben!

Aber: unter allen von diesen Möglichkeiten läuft Gale-Shapley:

Theorem: Jede Frau bekommt ihren bestmöglichen Mann!

Theorem: Jeder Mann bekommt seine schlechtest mögliche Frau!

→ Beweis: Übung...

(Widerspruchsbeweis: Betrachte erste Frau die nicht ihren bestmöglichen Mann bekommen hat, ...)

Sally	Peppermint	Lucy	Marcie
Charlie	Linus	Charlie	Charlie
Schroeder	Schroeder	Linus	Linus
Franklin	Charlie	Schroeder	Schroeder
Linus	Franklin	Franklin	Franklin

Charlie	Schroeder	Franklin	Linus
Sally	Sally	Peppermint	Lucy
Peppermint	Peppermint	Sally	Sally
Lucy	Lucy	Marcie	Peppermint
Marcie	Marcie	Lucy	Marcie

Sally	Peppermint	Lucy	Marcie
Charlie	Linus	Charlie	Charlie
Schroeder	Schroeder	Linus	Linus
Franklin	Charlie	Schroeder	Schroeder
Linus	Franklin	Franklin	Franklin

Charlie	Schroeder	Franklin	Linus
Sally	Sally	Peppermint	Lucy
Peppermint	Peppermint	Sally	Sally
Lucy	Lucy	Marcie	Peppermint
Marcie	Marcie	Lucy	Marcie

Tag 1: Sally → Charlie

Peppermint → Linus

Lucy → Charlie

Marcie → Charlie

Sally	Peppermint	Lucy	Marcie
Charlie	Linus	Charlie	Charlie
Schroeder	Schroeder	Linus	Linus
Franklin	Charlie	Schroeder	Schroeder
Linus	Franklin	Franklin	Franklin

Charlie	Schroeder	Franklin	Linus
Sally	Sally	Peppermint	Lucy
Peppermint	Peppermint	Sally	Sally
Lucy	Lucy	Marcie	Peppermint
Marcie	Marcie	Lucy	Marcie

Tag 1: Sally → Charlie
Peppermint → Linus
Lucy → Charlie
Marcie → Charlie

Tag 2: Lucy → Linus
Marcie → Linus

Sally	Peppermint	Lucy	Marcie
Charlie	Linus	Charlie	Charlie
Schroeder	Schroeder	Linus	Linus
Franklin	Charlie	Schroeder	Schroeder
Linus	Franklin	Franklin	Franklin

Charlie	Schroeder	Franklin	Linus
Sally	Sally	Peppermint	Lucy
Peppermint	Peppermint	Sally	Sally
Lucy	Lucy	Marcie	Peppermint
Marcie	Marcie	Lucy	Marcie

Tag 1: Sally → Charlie
Peppermint → Linus
Lucy → Charlie
Marcie → Charlie

Tag 2: Lucy → Linus
Marcie → Linus

Tag 3: Peppermint → Schroeder
Marcie → Schroeder

Sally	Peppermint	Lucy	Marcie
Charlie	Linus	Charlie	Charlie
Schroeder	Schroeder	Linus	Linus
Franklin	Charlie	Schroeder	Schroeder
Linus	Franklin	Franklin	Franklin

Charlie	Schroeder	Franklin	Linus
Sally	Sally	Peppermint	Lucy
Peppermint	Peppermint	Sally	Sally
Lucy	Lucy	Marcie	Peppermint
Marcie	Marcie	Lucy	Marcie

Tag 1: Sally → Charlie
Peppermint → Linus
Lucy → Charlie
Marcie → Charlie

Tag 2: Lucy → Linus
Marcie → Linus

Tag 3: Peppermint → Schroeder
Marcie → Schroeder

Tag 4: Marcie → Franklin

Sally	Peppermint	Lucy	Marcie
Charlie	Linus	Charlie	Charlie
Schroeder	Schroeder	Linus	Linus
Franklin	Charlie	Schroeder	Schroeder
Linus	Franklin	Franklin	Franklin

Charlie	Schroeder	Franklin	Linus
Sally	Sally	Peppermint	Lucy
Peppermint	Peppermint	Sally	Sally
Lucy	Lucy	Marcie	Peppermint
Marcie	Marcie	Lucy	Marcie

Tag 1:
Sally → Charlie
Peppermint → Linus
Lucy → Charlie
Marcie → Charlie

Sally ↔ Charlie

Tag 2:
Lucy → Linus
Marcie → Linus

Lucy ↔ Linus

Tag 3:
Peppermint → Schroeder
Marcie → Schroeder

Peppermint → Schroeder

Tag 4:
Marcie → Franklin

Marcie ↔ Franklin

Und in allgemeinen Graphen?

- Jeder kann mit jedem gematcht werden
- nicht immer stabiles Matching möglich!

A	B	C	D
B	C	A	?
C	A	B	?
D	D	D	?

D muss mit jemandem gematcht werden, dieser findet anderen besser und dieser wiederum anderen, ...

Es gibt aber Algorithmen für die Existenz etc...

3D-Matching

Gegeben: Drei Mengen A, B, C mit
$|A| = |B| = |C|$ und Menge
$E \subseteq \{(a,b,c) : a \in A, b \in B, c \in C\}$

Gesucht: $M \subseteq E$ mit

1. für je zwei Elemente (a_1, b_1, c_1) und $(a_2, b_2, c_2) \in M$ gilt
$a_1 \neq a_2$, $b_1 \neq b_2$, $c_1 \neq c_2$

2. Jedes Element aus A, B, C ist in M.

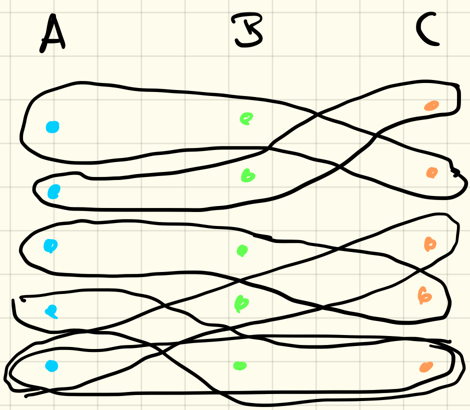
3D-Matching

Gegeben: Drei Mengen A, B, C mit $|A| = |B| = |C|$ und Menge $E \subseteq \{(a,b,c) : a \in A, b \in B, c \in C\}$

Gesucht: $M \subseteq E$ mit

1. für je zwei Elemente (a_1, b_1, c_1) und $(a_2, b_2, c_2) \in M$ gilt $a_1 \neq a_2$, $b_1 \neq b_2$, $c_1 \neq c_2$

2. Jedes Element aus A, B, C ist in M.



Bisheriges Matching war 2D-Matching...

3D-Matching ist nicht so einfach wie 2D-Matching, ... → NP-complete!

(1) 3D-Matching $\in$ NP:

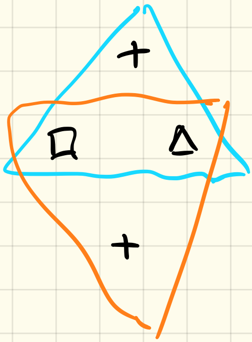
Für gegebene Lösung prüfen wir auf Disjunktheit und ob alle Elemente gematcht wird.

(2) Wenn 3D-Matching $\in$ P, dann auch 3-SAT:

Sei I eine Instanz von 3-SAT mit Variablen $x_1, \dots, x_n$ und mit Klauseln $c_i = (\ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3})$, $1 \leq i \leq m$

Wir erstellen daraus eine Instanz I' von 3D-Matching

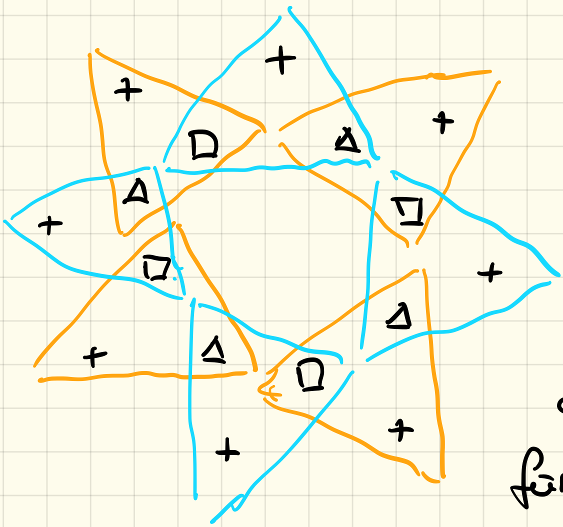
Codiere wahr / falsch:



<u>Entweder</u> blau <u>oder</u> orange können gewählt werden.

$$\left\{ \begin{array}{l} \text{blau} \rightarrow \text{wahr} \\ \text{orange} \rightarrow \text{falsch} \end{array} \right\}$$

Kommt Literal mehrfach vor, erstellen wir so einen Ring, der beliebig erweitert werden kann:

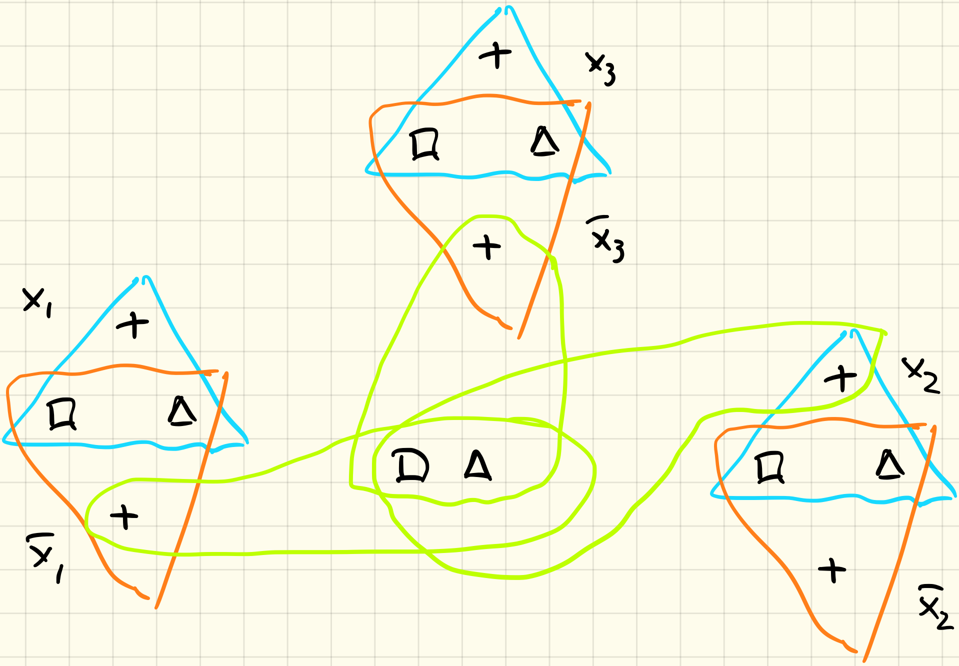


Es können entweder alle blauen oder alle orangenen gewählt werden für perfektes Matching!

Kodiere Klauseln:

Eine Klausel ist einfach □ Δ verbunden mit dem + des jeweiligen Variablen bzgl. des negierten Literals

z.B. $(x_1 \vee \bar{x}_2 \vee x_3)$



Dies reicht aber nicht, da wir zu viele Elemente vom Typ "+" haben!

Sei p die Anzahl an "+". Die Hälfte wird von den Variablengadjets abgedeckt, weitere m Stück von den Klauseln.
Wir fügen also $\frac{p}{2} - m$ viele $(\square \Delta)$ ein und verbinden diese mit allen "+" Elementen.

Zu zeigen ist jetzt noch:

(1) I erfüllbar $\Longleftrightarrow$ I' perfektes Matching

(2) Reduktion ist polynomiell

Beweis für (i)

„$\Rightarrow$" Sei $\sigma = (\ell_1, \dots, \ell_n)$ erfüllende Belegung für I. Wähle für ℓ_i blau falls ℓ_i wahr oder orange falls ℓ_i falsch.

Weise jedem Klauselgadget ein + zu und decke die restlichen + mit den zusätzlichen Paaren (▲ ◻) ab.

$\rightarrow$ I' hat perfektes Matching

„$\Leftarrow$" Sei M perfektes Matching. Blockiert Klauselgadget orange, wähle für entsprechende Variable Wert $= 1$; falls blau analog.

Da Klausel nie beides wählen kann, gibt es erfüllende Belegung für I.

$\square$

Beweis für (2)

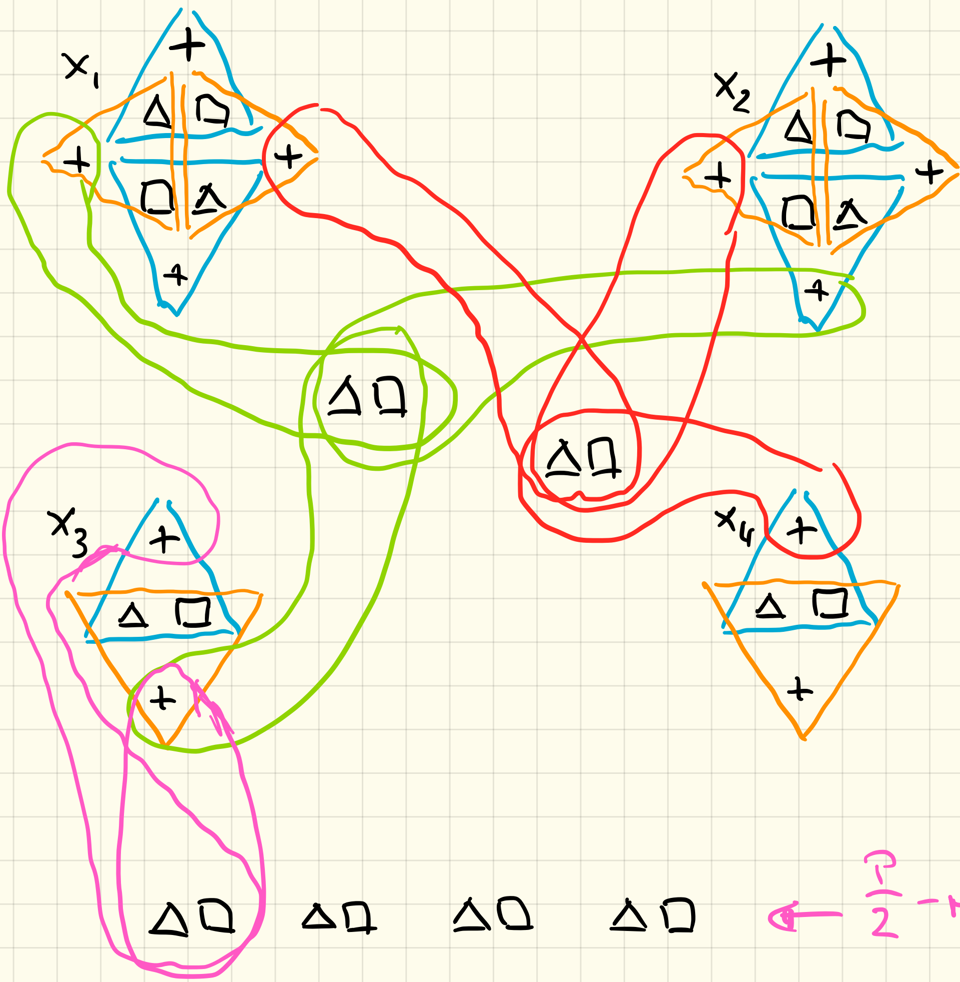
Jede Variable kann in maximal m Klauseln vorkommen, d.h. $p \in O(n \cdot m)$, es gibt höchstens $O(n^2 m^2)$ viele Kanten.

$\longrightarrow$ polynomielle Laufzeit

Rückübersetzen in $O(1)$ ...

$\square$

$$(x_1 \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_2 \vee \overline{x_4})$$



$\leftarrow \frac{p}{2} - m$

$= 6-2=4$

Füller!

Jedes Füllerpaar wird
mit jedem + verbunden...

Approximation von 3D-Matching

Theorem: Es gibt eine $\frac{1}{3}$-Approximation für Max-3DM.

Beweis: Sei M ein beliebiges, nicht-erweiterbares Matching und OPT das größtmögliche.
Sei $(a,b,c) \in M$. Diese Kante kann höchstens drei Kanten aus OPT blockieren.

$$\rightarrow |M| \geq \frac{1}{3} \text{OPT}. \qquad \square$$