Näherungsalgorithmen

(Approximationsalgorithmen)

WiSe 2024/25 in Trier

Henning Fernau
Universität Trier
fernau@uni-trier.de

11. Oktober 2024

Näherungsalgorithmen Gesamtübersicht

- Organisatorisches
- Einführung / Motivation
- Grundtechniken für Näherungsalgorithmen
- Approximationsklassen (Approximationstheorie)

Organisatorisches

Vorlesung Besprechung bzw. Übungen (nach Absprache)
Termin: donnerstags 12-14 Uhr im H13

Modulprüfungen werden bei uns immer als mündliche Prüfungen abgelegt

Meine Sprechstunde: DO, 13-14 Uhr im F213
Sprechstunde Kevin Mann: DI, 13-14 Uhr im H407

Kontakt: fernau@uni-trier.de, mann@uni-trier.de

Einbettung

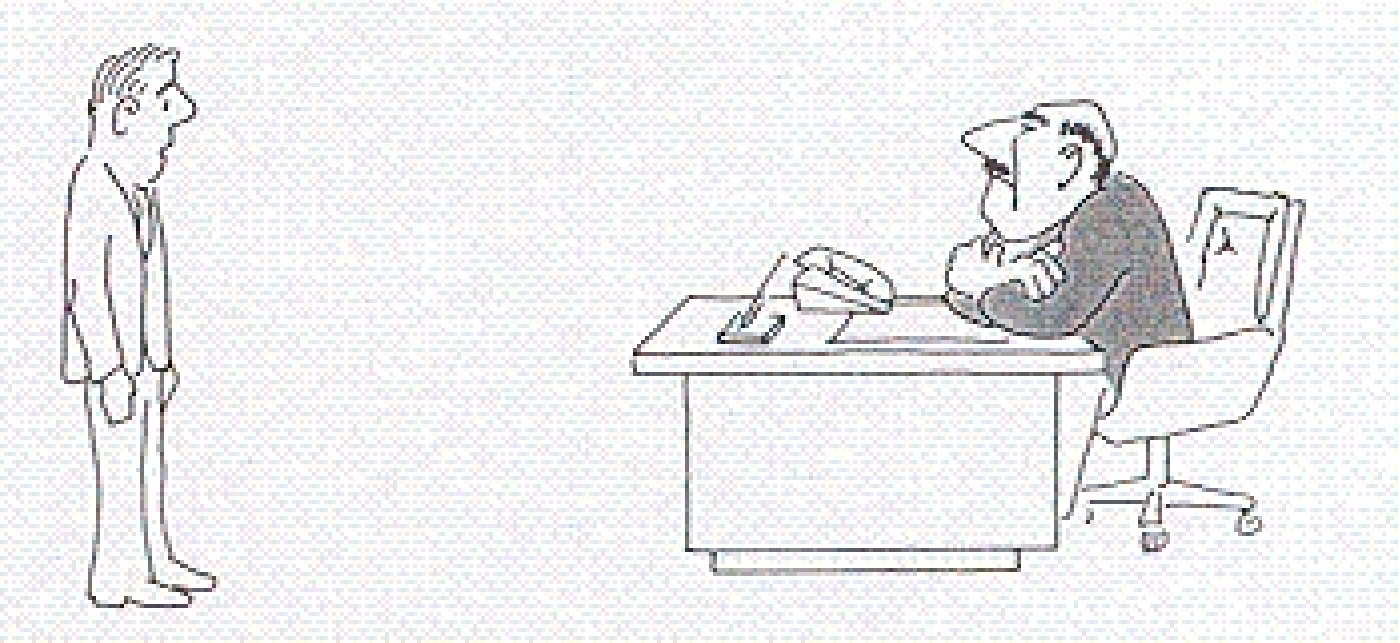
- Zum Umgang mit NP-schweren Problemen:
  Näherungsalgorithmen im Wechsel mit Parametrisierten Algorithmen
- Zur Komplexität von Problemen:
  Komplexitätstheorie A im Wechsel mit Komplexitätstheorie B
- Algorithmische Vorlesungen bei Kollegen Näher und Kindermann
- Optimierungsvorlesungen in der Mathematik

Motivation

Viele interessante Probleme (aus der Praxis!) sind NP-schwer
$\Rightarrow$ wohl keine Polynomialzeitalgorithmen sind zu erwarten.

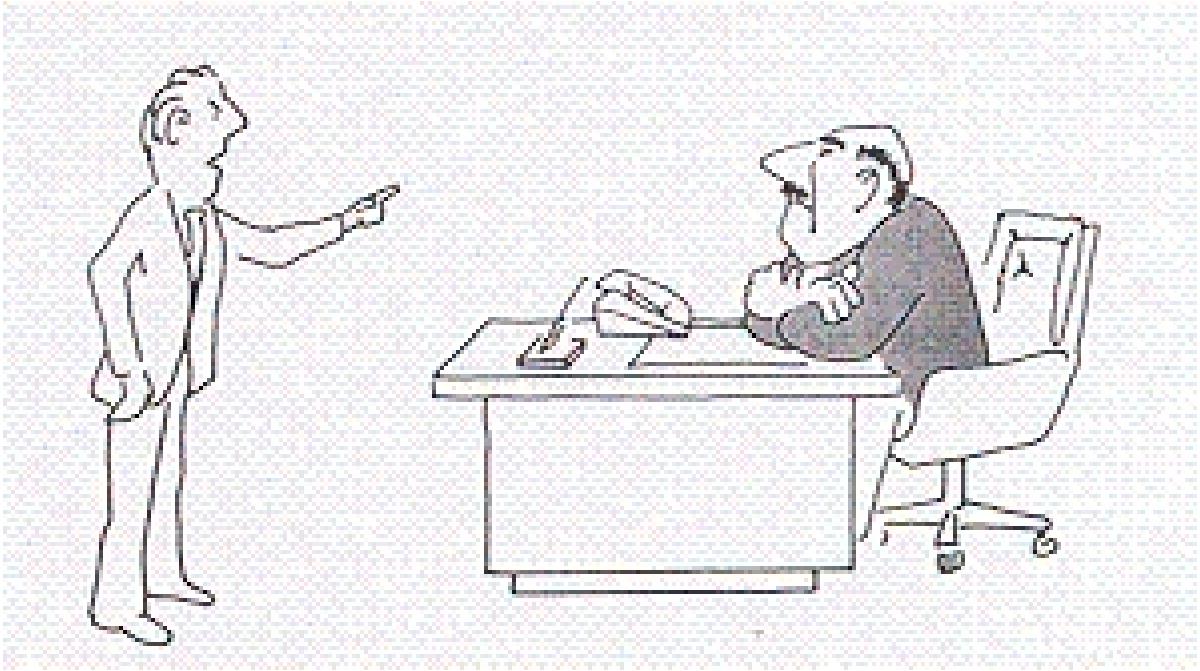
Motivation

siehe http://max.cs.kzoo.edu/~kschultz/CS510/ClassPresentations/NPCartoons.html wiederum aus Garey / Johnson



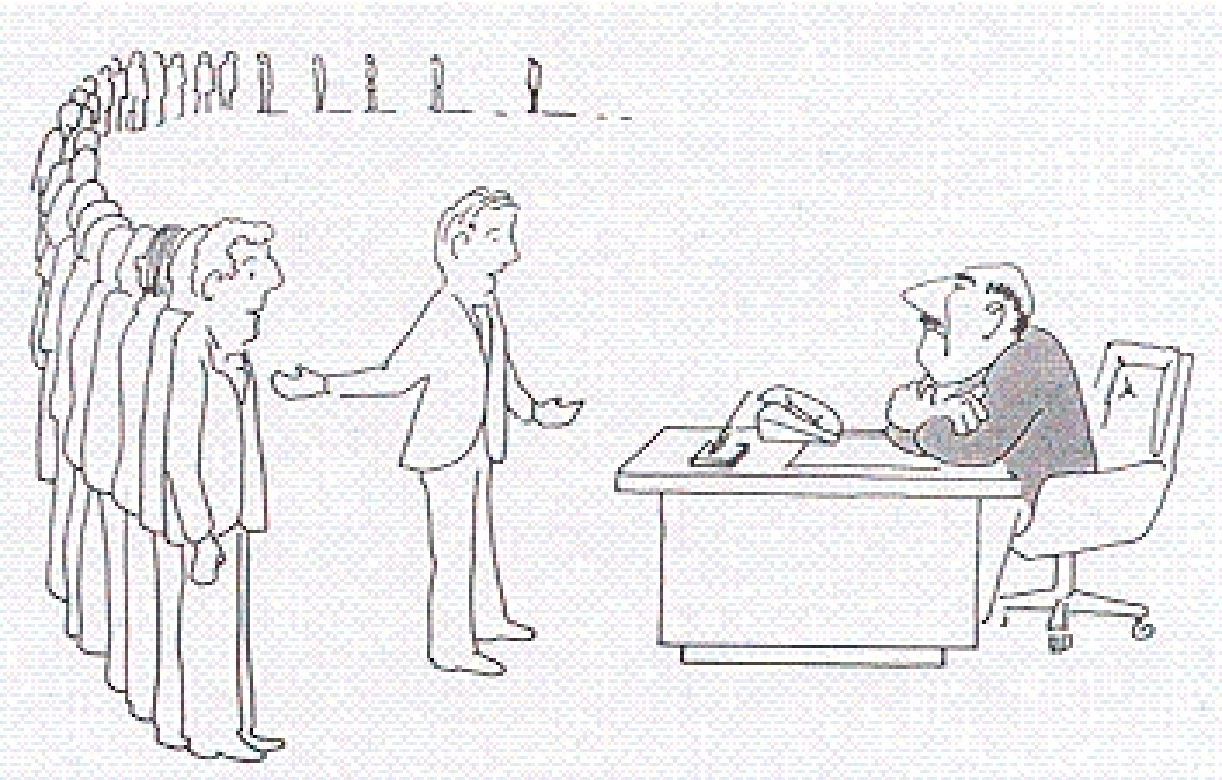
Sorry Chef, aber ich kann für das Problem keinen guten Algorithmus finden. . .

Die beste Antwort wäre hier aber. . .



. . . Ich kann aber beweisen, dass es für das Problem keinen guten Algorithmus geben kann !

Was die Komplexitätstheorie statt dessen liefert. . .



. . . Ich kann aber beweisen, dass das alle anderen auch nicht können !

Heuristische Verfahren

- Ziel: schnelle Laufzeit
- „hoffentlich“ wird „gute“ Lösung gefunden.
  $\hookrightarrow$ keine „mathematische“ Garantie, nur „Empirie“.
- typische Beispiele: Greedy-Verfahren

Randomisierte Verfahren

- finden optimale Lösung „mit groer Wahrscheinlichkeit“.
  Hinweis: Eine Vorlesung „randomisierte Algorithmen“ wird manchmal angeboten; oder auch Seminar zum Thema.

Parametrisierte Verfahren

- finden stets optimale Lösung.

- versuchen, den nicht-polynomiellen Laufzeitanteil auf einen (als klein angenommenen) sogenannten Parameter zu beschränken.
  Hinweis: Spezialvorlesung „paramet(e)risierte Algorithmen“
  wird im Wechsel mit „Näherungsalgorithmen“
  in jedem zweiten Wintersemester angeboten.

Näherungsverfahren

- sind „Heuristiken mit Leistungsgarantie“.
- Güte von Näherungsverfahren kann
  - absolut oder
  - relativ zum Optimum gemessen werden

Ein *Optimierungsproblem* $\mathcal{P}$ wird beschrieben durch ein Quadrupel $(I_\mathcal{P}, S_\mathcal{P}, m_\mathcal{P}, \text{opt}_\mathcal{P})$:

1. $I_\mathcal{P}$ ist die Menge der möglichen Eingaben (*Instanz*en),

2. $S_\mathcal{P} : I_\mathcal{P} \to$ Menge der *zulässigen Lösungen* (feasible solutions),

3. $m_\mathcal{P} : (x, y) \mapsto m_\mathcal{P}(x, y) \in \mathbb{N}$ (oder $\mathbb{Q}, ...$) für $x \in I_\mathcal{P}, y \in S_\mathcal{P}(x)$ liefert den *Wert* der zulässigen Lösung y und

4. $\text{opt}_\mathcal{P} \in \{\mathsf{min}, \mathsf{max}\}$: $\mathcal{P}$ Minimierungs- oder Maximierungsproblem ?

$I_\mathcal{P}$ und $S_\mathcal{P}(x)$ sind —geeignet binär codierte— formale Sprachen, also Sprachen über dem Alphabet $\{0, 1\}$.

Weitere Bezeichnungen

$S^*_{\mathcal{P}} : I_{\mathcal{P}} \to$ Menge der *bestmöglichen Lösungen* (optimum solution), d.h.

$$\forall x \in I_{\mathcal{P}}\ \forall y^*(x) \in S^*_{\mathcal{P}}(x) : m_{\mathcal{P}}(x, y^*(x)) = \mathrm{opt}_{\mathcal{P}}\{m_{\mathcal{P}}(x, z) \mid z \in S_{\mathcal{P}}(x)\}.$$

Der Wert einer bestmöglichen Lösung wird auch $m^*_{\mathcal{P}}(x)$ notiert.

Ist $\mathcal{P}$ aus dem Zusammenhang klar, so schreiben wir kurz —unter Fortlassung des Indexes $\mathcal{P}$— I, S, m, opt, S^*, m^*.

Ziel: Polynomialzeitalgorithmus A, der zu x eine zulässige Lösung y berechnet, für die $m(x, y)/m^*(x)$ beschränkt ist.

Unser Ziel: Genauer definiert

Ist $\mathcal{P}$ ein Optimierungsproblem, so ist —für jede Instanz x von $\mathcal{P}$ und für jede zulässige Lösung y von x— die *Leistungsgüte* (engl: performance ratio) von y bezüglich x definiert als

$$R(x,y) := \max\left\{\frac{m^*(x)}{m(x,y)}, \frac{m(x,y)}{m^*(x)}\right\}.$$

Ist $\mathcal{A}$ ein Approximationsalgorithmus für $\mathcal{P}$, so heißt $\mathcal{A}$ *r-Approximation*, wenn

$$\forall x \in I_{\mathcal{P}} : R(x, \mathcal{A}(x)) \leq r.$$

Bem.: Die Leistungsgüte ist gleich Eins, wenn die Lösung optimal ist, und sie ist sehr groß, wenn die Lösung schlecht ist.

Eine erste Definition für ein einführendes Beispiel

Eine *Knotenüberdeckung* (engl: vertex cover) eines Graphen $G = (V, E)$ ist eine Menge $C \subseteq V$ derart, dass
für jede Kante $e = \{v_1, v_2\} \in E$ gilt: $e \cap C \neq \emptyset$, d.h.,
e wird durch einen Knoten aus C abgedeckt.

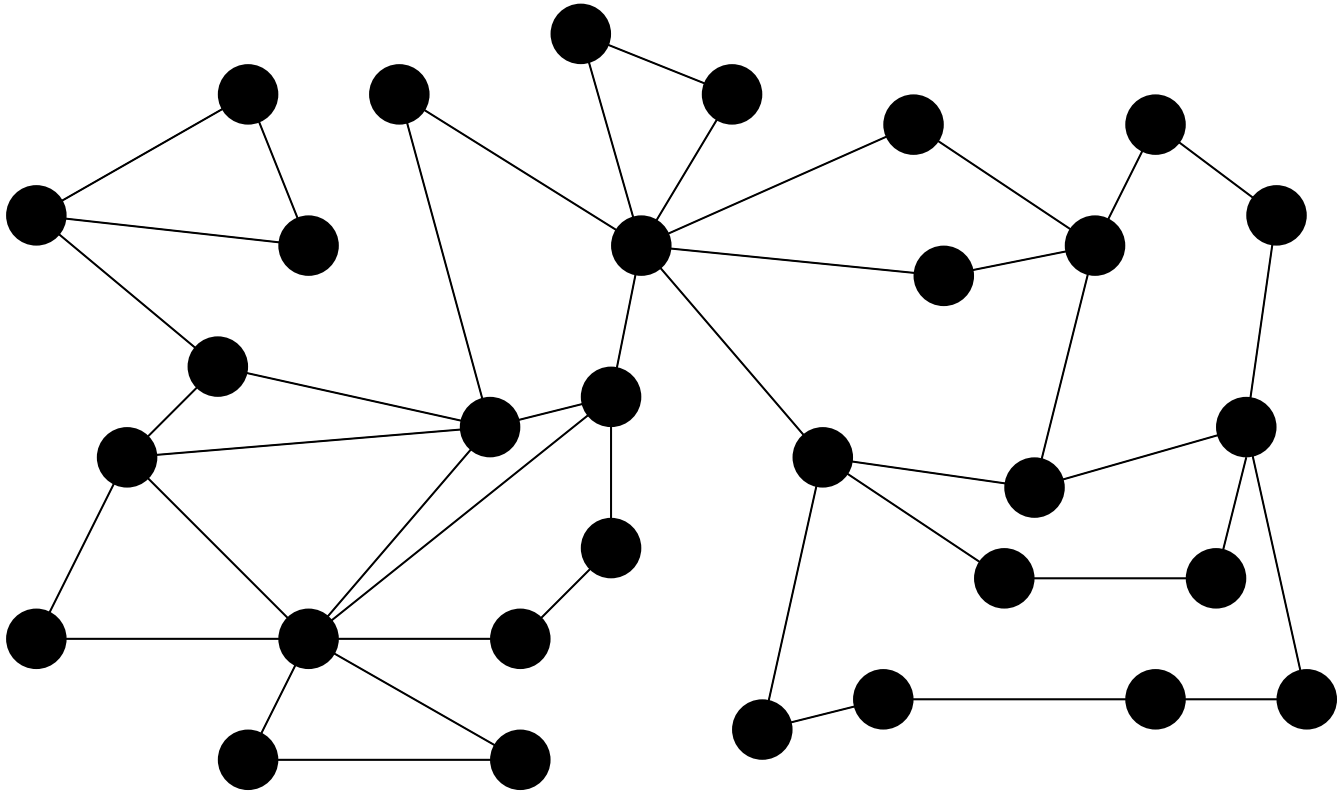
Knotenüberdeckungsproblem VC: Finde kleinstmögliche Knotenüberdeckung !

Hinweis: VC ist eines der grundlegenden NP-schweren Probleme.

Ein Beispiel: das Knotenüberdeckungsproblem VC (als Optimierungsproblem)

1. $I = \{G = (V, E) \mid$ G ist Graph $\}$
2. $S(G) = \{U \subseteq V \mid \forall \{x, y\} \in E : x \in U\}$ (Knotenüberdeckungseigenschaft)
3. $m = |U|$
4. opt $=$ min

Ein Beispiel: Wie groß ist ein kleinstmögliches VC ?

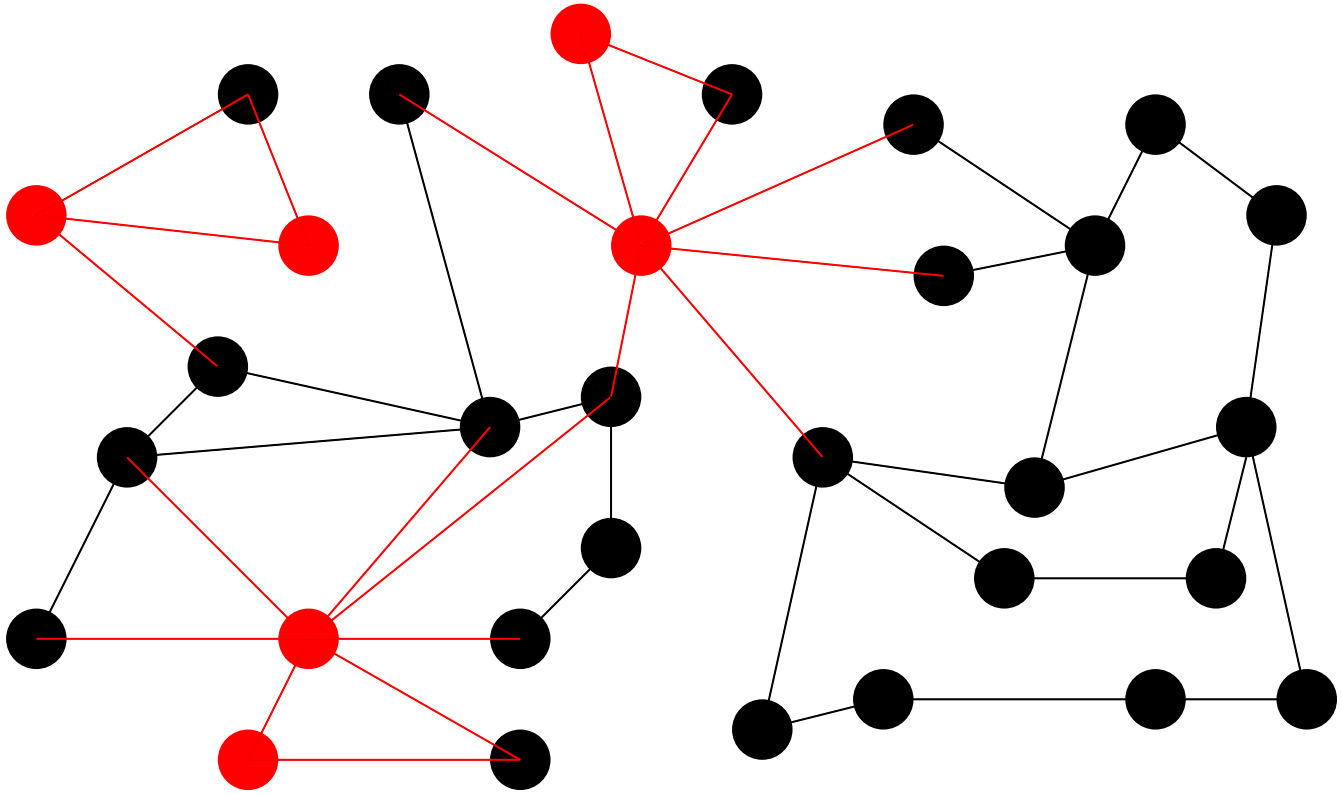


Einfache Beobachtungen und Regeln

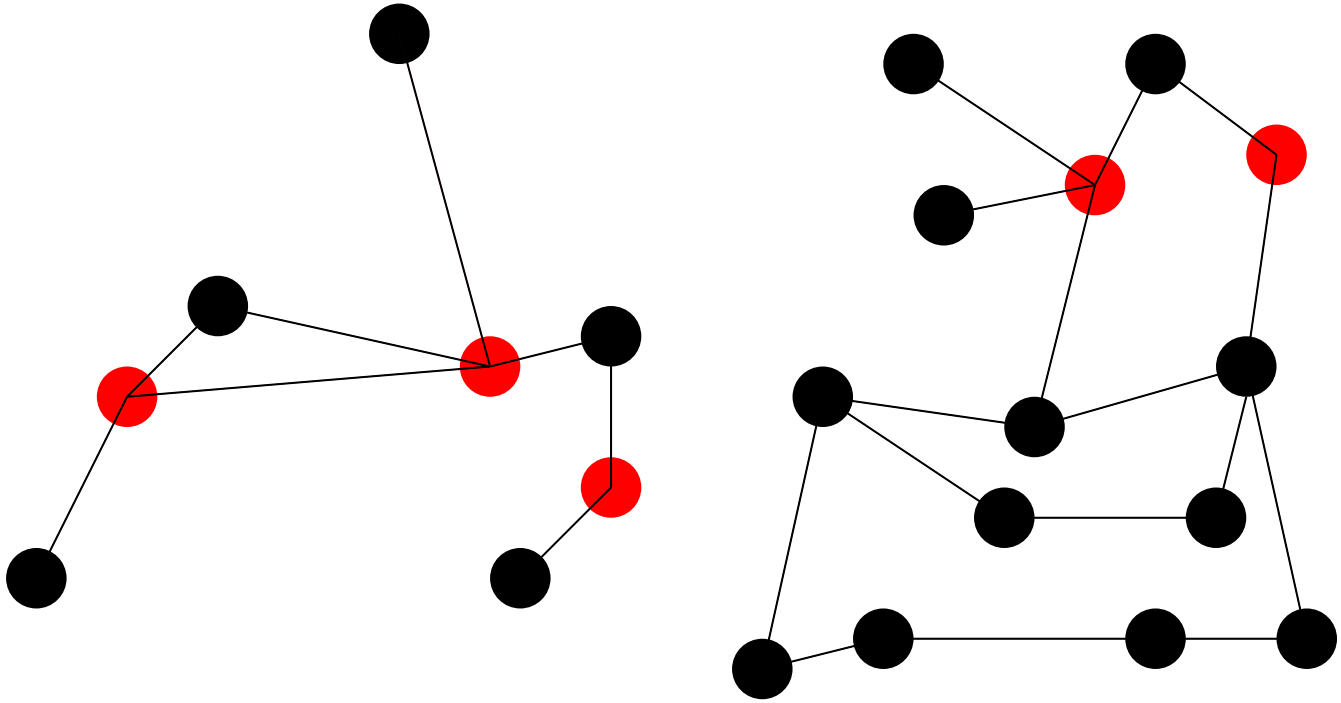
- Zwei Knoten in einem Dreieck gehören in ein VC.

- Wenn Wahlmöglichkeit besteht, nimm solche Knoten, die „noch mehr“ abdecken können.
  $\leadsto$ Nimm Nachbarn eines Grad-1-Knotens ins VC. (Blattregel)
  $\leadsto$ Gibt es in einem Dreieck einen Knoten vom Grad zwei, so nimm dessen beide Nachbarn ins VC. ($\triangle$-Regel)

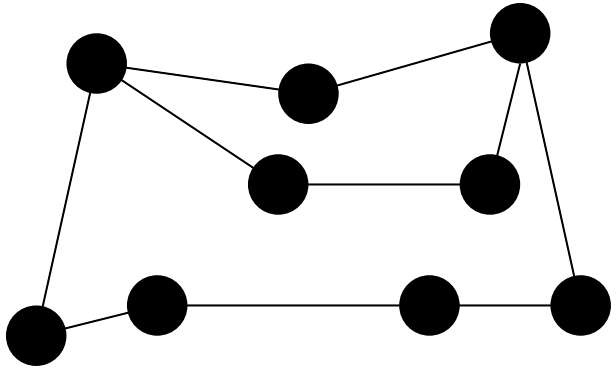
Unser Beispiel: Wende die $\triangle$-Regeln nacheinander an: ergibt 6 VC-Knoten.



Unser Beispiel: Die Blattregeln kaskadieren $\rightsquigarrow$ 5 weitere VC-Knoten.



Unser Beispiel: Was bleibt übrig ?

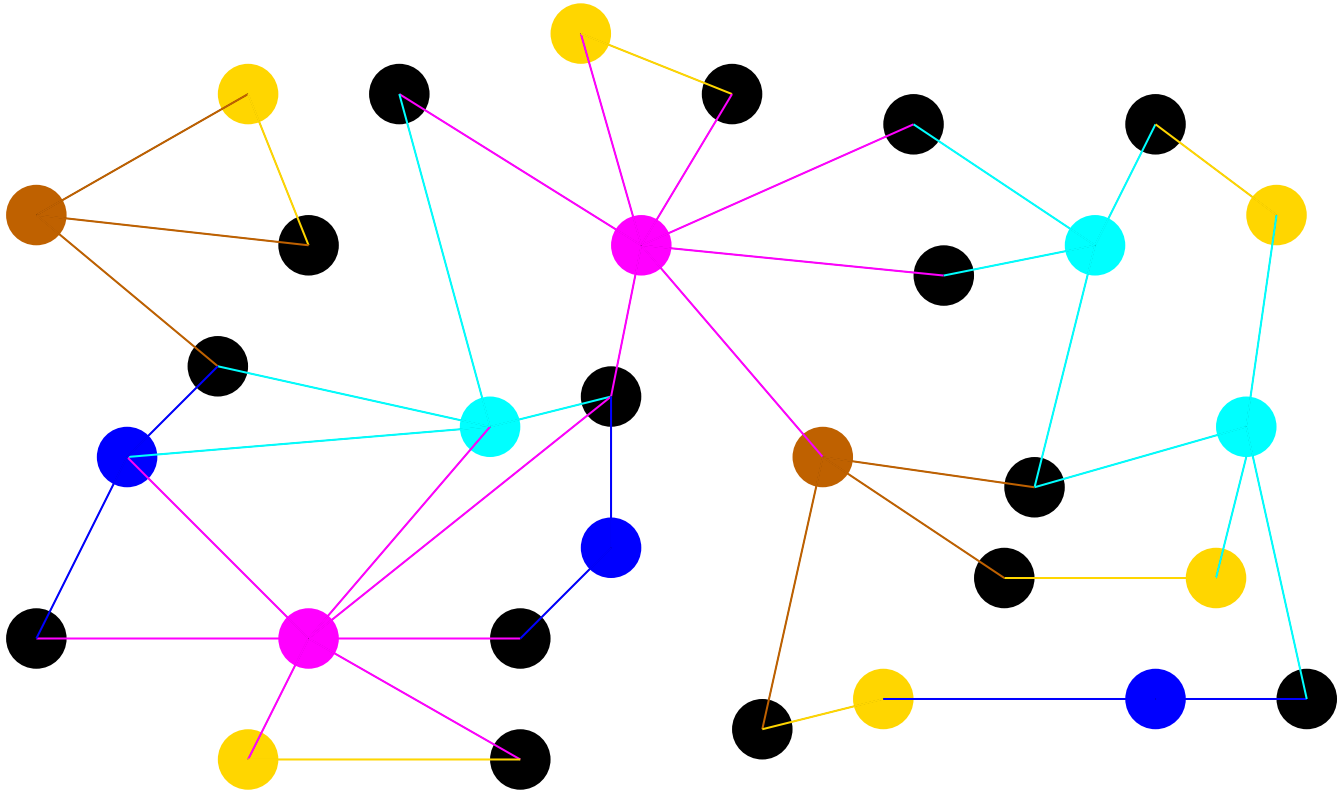


Ein Greedyverfahren GreedyVC (G, C)

- Falls $E(G)$ leer, gib C aus; exit.
- Suche Knoten v mit maximalem Grad in G.
- Berechne GreedyVC $(G - v, C \cup \{v\})$

Hinweis: $G - v$ entsteht aus G, indem v (mit anliegenden Kanten) aus G gelöscht wird.

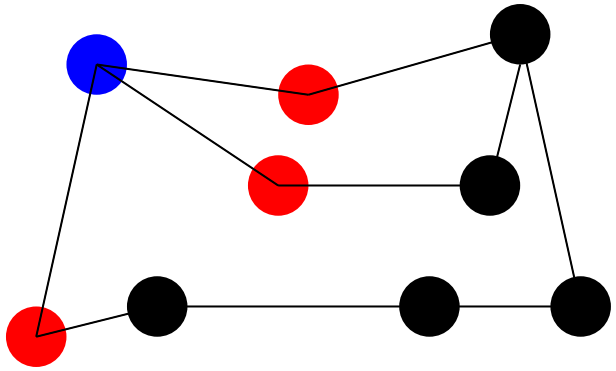
Unser Beispiel: Greedy at work. . . Farben kodieren Grad $\leadsto$ 16 VC-Knoten



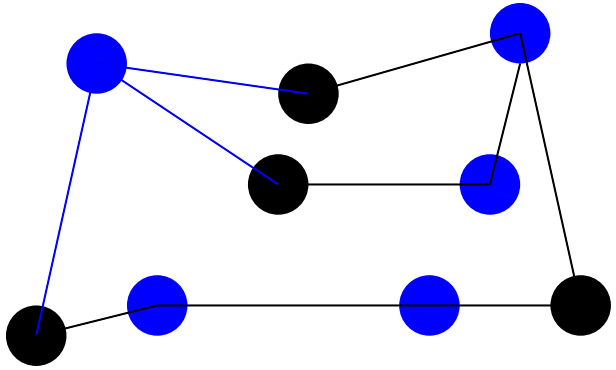
Ein Suchbaumverfahren SuchbaumVC(G, C, k)

- Falls $E(G)$ leer, gib C „nach oben“; exit.
- Falls $k = 0$: exit (Fehlerfall).
- Nimm irgendeine Kante $e = \{v_1, v_2\}$ aus G und verzweige:
  1. Berechne SuchbaumVC$(G - v_1, C \cup \{v_1\}, k - 1)$
  2. Berechne SuchbaumVC$(G - v_2, C \cup \{v_2\}, k - 1)$
  3. Liefere kleinere Lösung „nach oben“.

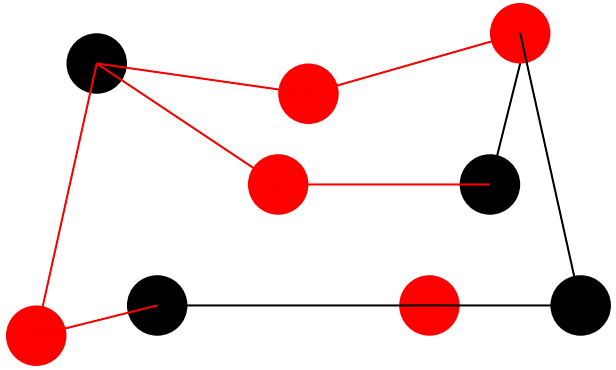
Weitere Idee: Knotenorientierter Suchbaum, kombiniert mit den Regeln $\rightsquigarrow$ zwei Zweige



Weitere Idee: Kombiniere Suchbaum mit unseren Regeln der blaue Fall



Weitere Idee: Kombiniere Suchbaum mit unseren Regeln der rote Fall



Ergebnis für unser Beispiel:

Die kleinste Knotenüberdeckung enthält 16 Knoten.

Eine kleinste Knotenüberdeckung wurde von der Heuristik gefunden.

Die Verwendung von sog. Reduktionsregeln hilft, die Verzweigungszahl bei Suchbaumalgorithmen zu verringern.

Ein Näherungsverfahren $N1VC(G = (V, E), C)$

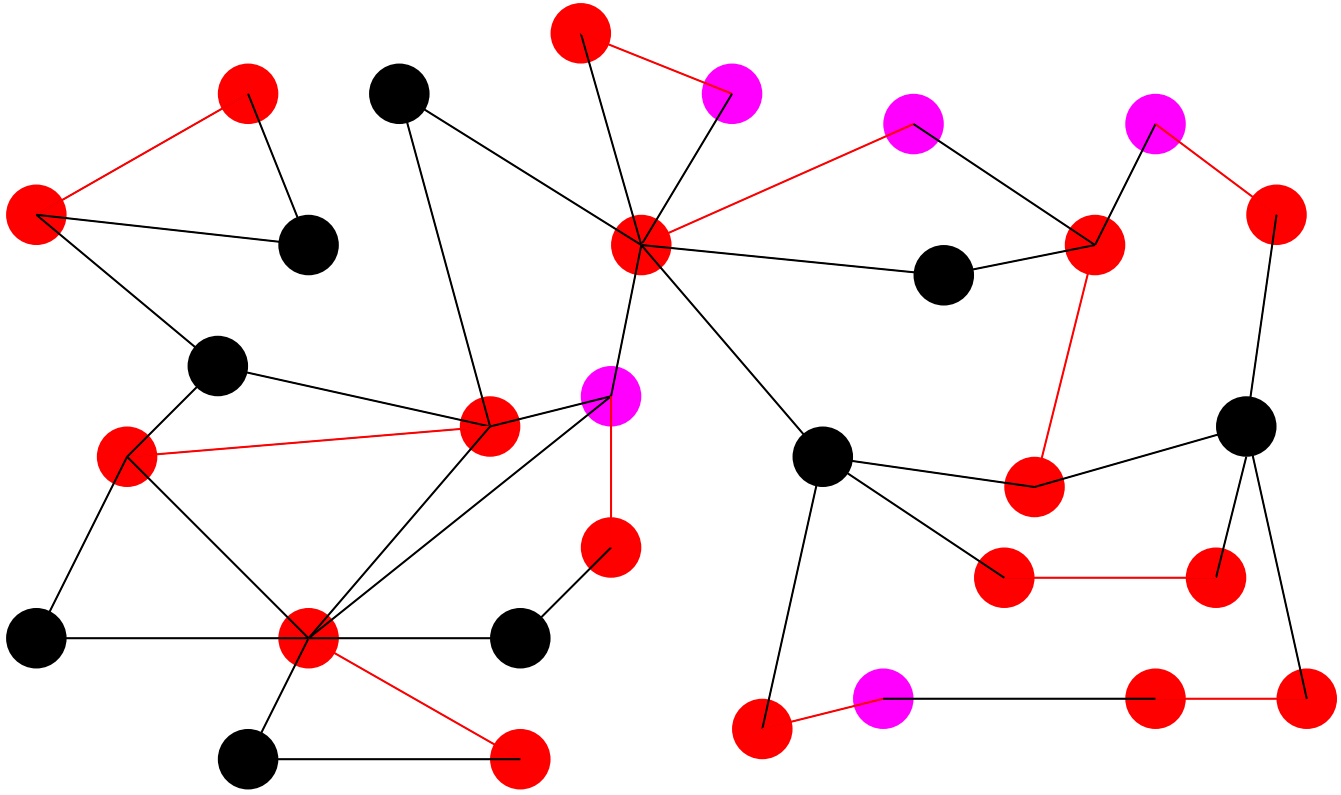
- Falls E leer, gib C aus; exit.

- Nimm irgendeine Kante $e = \{v_1, v_2\}$ aus G
  und berechne $N1VC(G - \{v_1, v_2\}, C \cup \{v_1, v_2\})$

Dies ist ein **2**-Approximations-Verfahren (s.u.)

Unser Beispiel: magentafarbene Knoten gehören nicht zu *(inklusions-)minimaler* Überdeckung; die gefundene Lösung ist daher fast bestmöglich

Wichtig: Minimal (Greedy) versus Minimum VC (NP-schwer).



Ein (weiteres) Beispiel: MAXSAT

I : Menge C von Klauseln über einer Menge V von Variablen
(Eine Klausel ist eine Disjunktion von Literalen.
Ein Literal ist eine Variable oder ihre Negation.
Formal ist C als Menge von Mengen von Literalen aufzufassen.)

S : Menge der Belegungen $f : V \to \{\text{true}, \text{false}\}$

m : $|\{c \in C \mid f(c) = \text{true}\}|$

opt : max

Beispiel: Die KNF-Formel $(x \vee \bar{y}) \wedge (\bar{x} \vee y)$ würden wir als $C = \{\{x, \bar{y}\}, \{\bar{x}, y\}\}$ schreiben.

Ein einfacher Algorithmus

1. Es sei C eine Klauselmenge über V.
2. Evaluiere die Belegungen $f_0 = 0$ und $f_1 = 1$.
3. Liefere $f = f_i$ zurück mit $m(f_i) \geq m(f_{2-i})$.

Satz: Der Algorithmus ist eine 2-Approximation.

Beweis: Für jede Klausel c gilt: $f_0(c) \vee f_1(c)$ ist wahr.

Daher gilt: $m(f_0) + m(f_1) \geq |C|$.

$\rightsquigarrow 2 \cdot m(f) \geq m(f_0) + m(f_1) \geq |C| \geq m^*(C)$.

Ein Beispiel: das Problem unabhängiger Knotenmengen (als Opt.problem)

1. $I = \{G = (V, E) \mid$ G ist Graph $\}$
2. $S(G) = \{U \subseteq V \mid \forall \{x, y\} \in E : |\{x, y\} \cap U| \leq 1\}$ (U ist unabhängig)
3. $m = |U|$
4. opt $=$ max

Bem. Das Komplement einer unabhängigen Menge ist eine Knotenüberdeckung und umgekehrt.

Ein Beispiel: das Problem unabhängiger Kantenmengen (als Opt.problem)
Auch bekannt als MAXIMUM MATCHING.

1. $I = \{G = (V, E) \mid$ G ist Graph $\}$
2. $S(G) = \{U \subseteq E \mid \forall e, f \in U : e \cap f \neq \emptyset \implies e = f\}$ (Matching-Eigenschaft)
3. $m = |U|$
4. opt $=$ max

Bem. Dieses Problem kann man in Polynomialzeit lösen, wie Edmonds zeigen konnte. Die Größe jedes maximalen (!) Matchings ist eine untere Schranke für die Größe einer jeden Knotenüberdeckung.