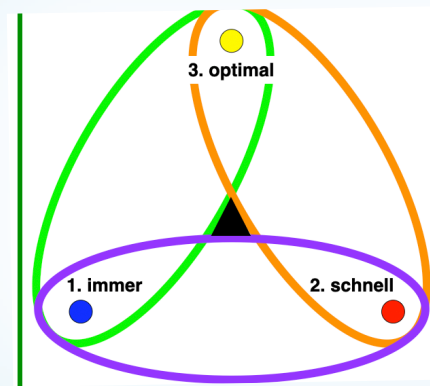
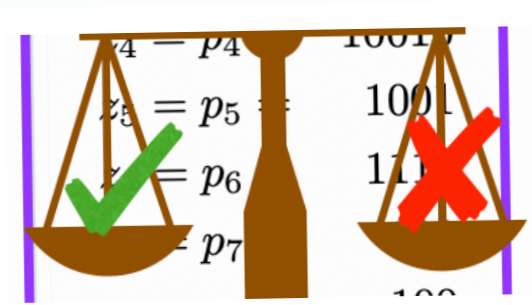




it dem Schicksal hadern
uf Glück vertrauen
art arbeiten
rwartungen zurückschau



5 Komplexität

Algorithmen und Datenstrukturen 2
Sommer 2026

Prof. Dr. Sándor Fekete

5.1 Die Klasse P: „Perfekte“ Algorithmen

Erwünschte Eigenschaften von Algorithmen

Ziele: Ein „perfekter“ Algorithmus sollte

1.immer: für jede beliebige Instanz

2.schnell: in polynomieller Zeit

3.optimal: eine bestmögliche Lösung

liefern.

Definition 5.1. Ein algorithmisches Problem Π gehört zur Klasse P , wenn es für Π einen Algorithmus mit polynomieller Laufzeit gibt, der immer eine optimale Lösung findet.

Fractional Knapsack

The diagram shows a table with 6 columns representing items. The first column has headers 'Nummer', 'Minuten', and 'Punkte'. The items are listed as follows:

Nummer	Minuten	Punkte
1	20	3
2	32	3
3	40	10
4	8	10
5	16	16
6	16	10

A diagonal banner across the table reads: "Algorithmus 1.4 Greedy-Algorithmus für FRACTIONAL KNAPSACK". Below the banner, the input and output are specified: "Eingabe: $z_1, \dots, z_n, Z, p_1, \dots, p_n$ " and "Ausgabe: $x_1, \dots, x_n \in [0, 1]$ ". A red box at the bottom right contains the text: "Kann Knut die Klausur bestehen?"

Achtung:

Das klassifiziert Probleme,
nicht Algorithmen!

Erwünschte Eigenschaften von Algorithmen

Ziele: Ein „perfekter“ Algorithmus sollte

1.immer: für jede beliebige Instanz

2.schnell: in polynomieller Zeit

3.optimal: eine bestmögliche Lösung

liefern.

Definition 5.1. Ein algorithmisches Problem Π gehört zur Klasse P , wenn es für Π einen Algorithmus mit polynomieller Laufzeit gibt, der immer eine optimale Lösung findet.

0-1 Knapsack ?!

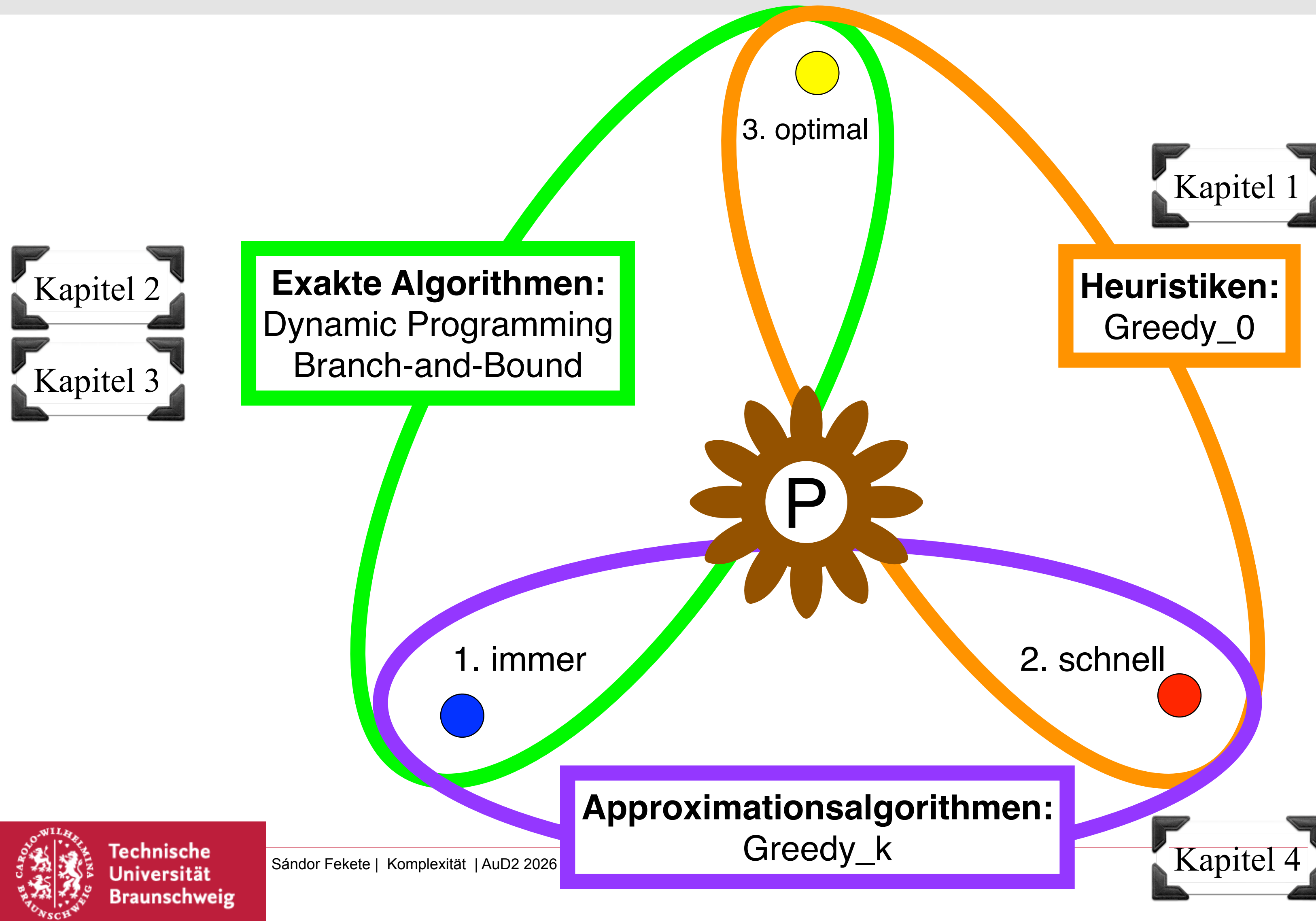
1	Nummer	2	3	4	5	6
20	Minuten	32	40	8	16	4
3	Punkte	3	10	5	2	4
			10			
		8	32	11	12	13
7		40	5	28	20	16
32		9		3	9	10
2						
		9	14	15	16	
		8	20	40	24	
		2	3	10	4	

Kann Knut die Klausur bestehen?

Achtung:

Das klassifiziert Probleme,
nicht Algorithmen!

Das Dreieck der Perfektion



0-1 Knapsack ?!

1	Nummer	2	3	4	5	6
20	Minuten	32	40	8	16	4
3	Punkte	3	10	5	2	4
			10			
		8	32	11	12	13
7		40	5	28	20	16
32		9		3	9	10
2						
		9	14	15	16	
		8	20	40	24	
		2	3	10	4	

Kann Knut die Klausur bestehen?

Gehört 0-1 Knapsack zu P?

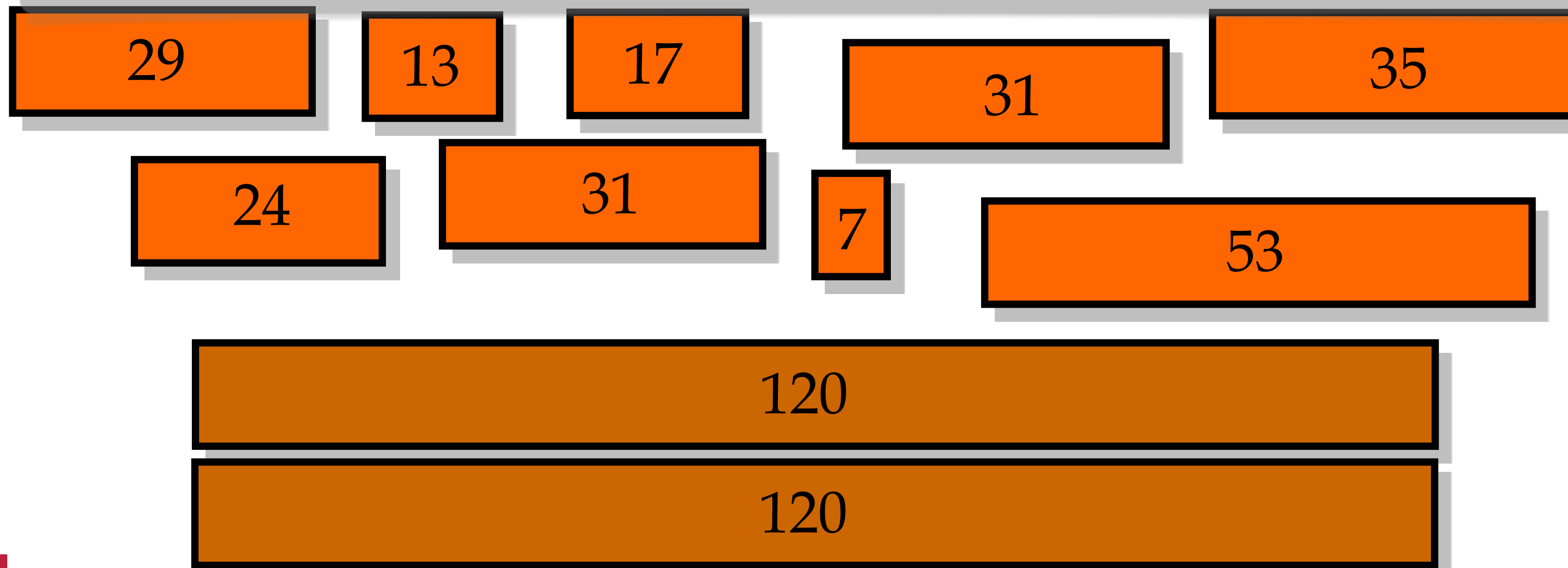
5.2 Die Klasse NP: „NachPrüfen“

Nachprüfen!

Beispiel 1.10.

Partition für $\{z_1, \dots, z_9\} = \{7, 13, 17, 24, 29, 31, 31, 35, 53\}$

Gesamtsumme: 240



Nachprüfen!

Beispiel 1.10.

Partition für $\{z_1, \dots, z_9\} = \{7, 13, 17, 24, 29, 31, 31, 35, 53\}$

Gesamtsumme: 240

Existenz einer Lösung: einfach zu überprüfen!

24	13	17	31	35
29	7	31	53	

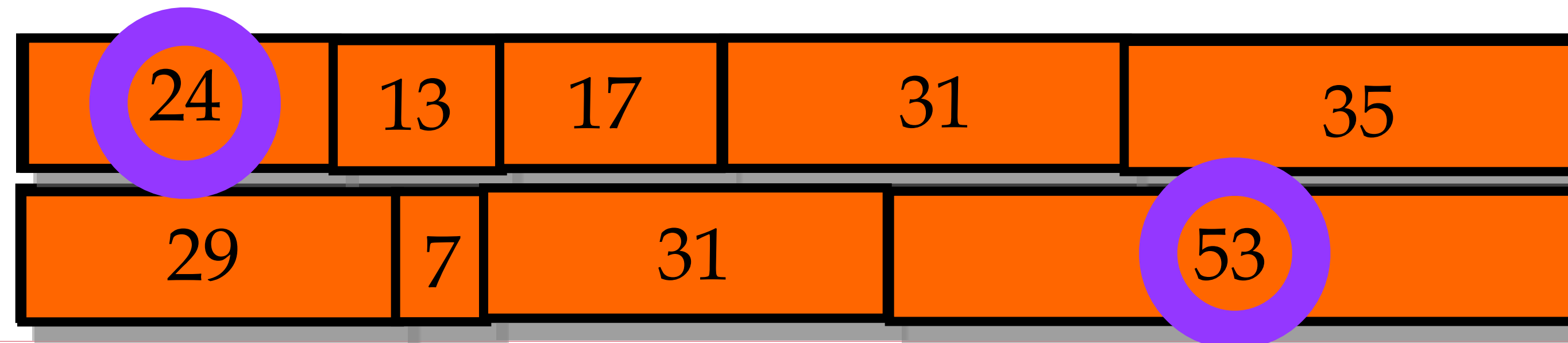
Nachprüfen!

Beispiel 1.10.

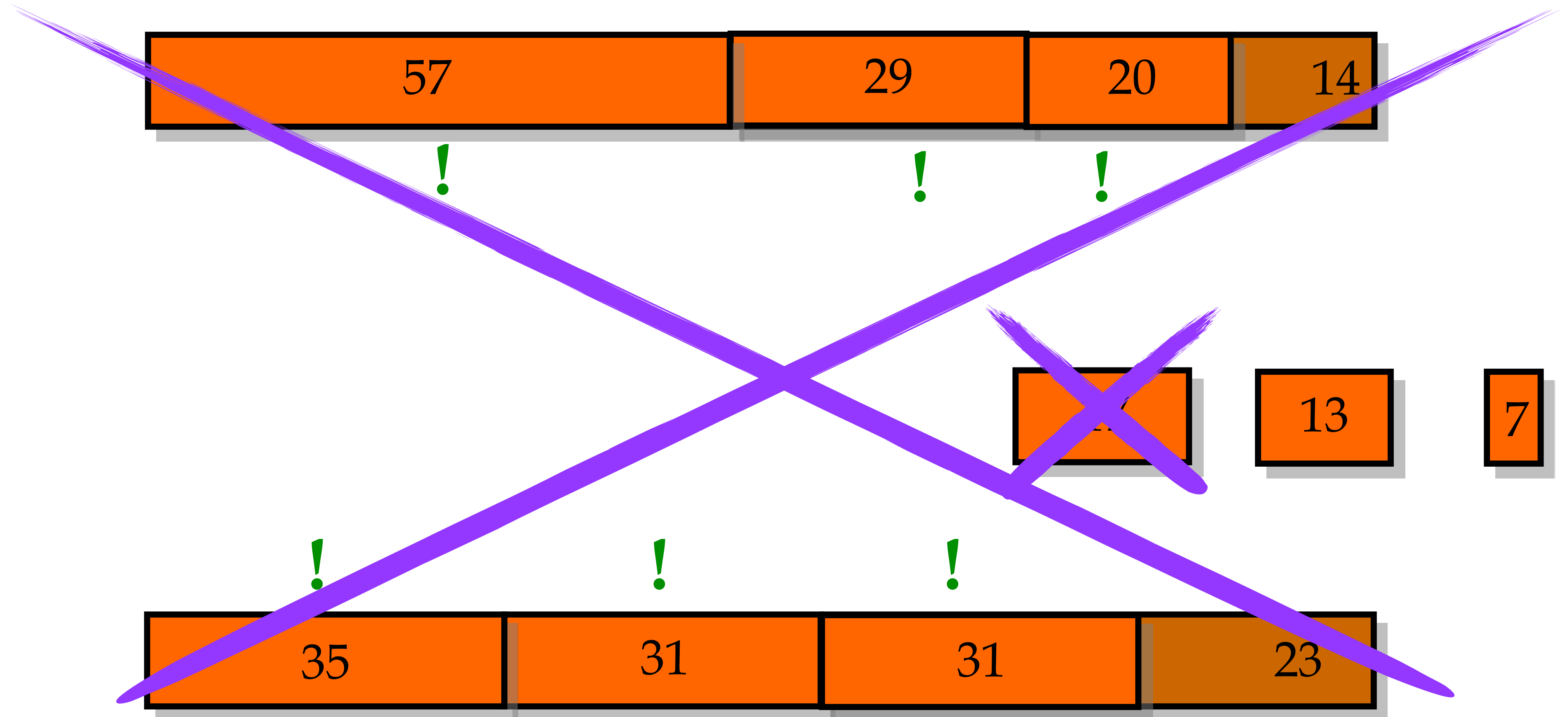
Partition für $\{z_1, \dots, z_9\} = \{7, 13, 17, 20, 29, 31, 31, 35, 57\}$

Gesamtsumme: 240

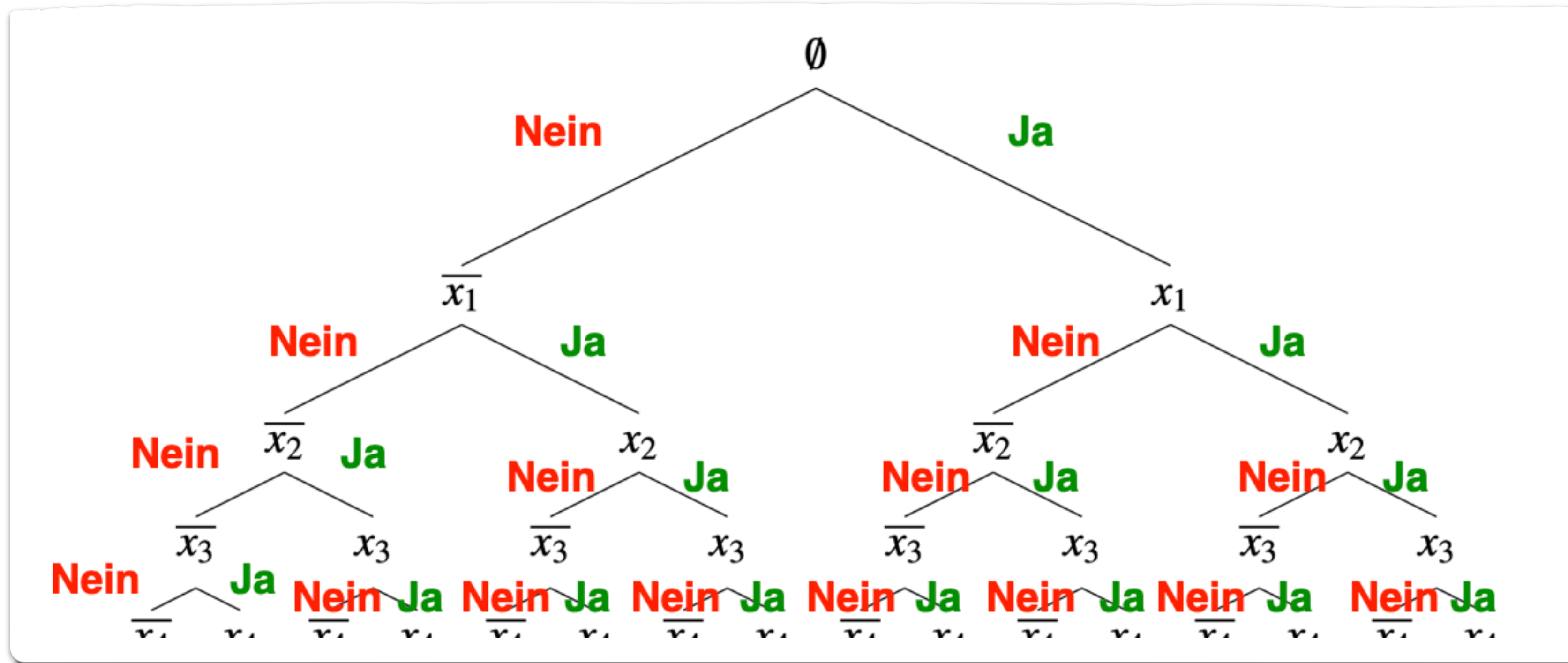
Existenz einer Lösung: einfach zu überprüfen!



Keine Lösung?!

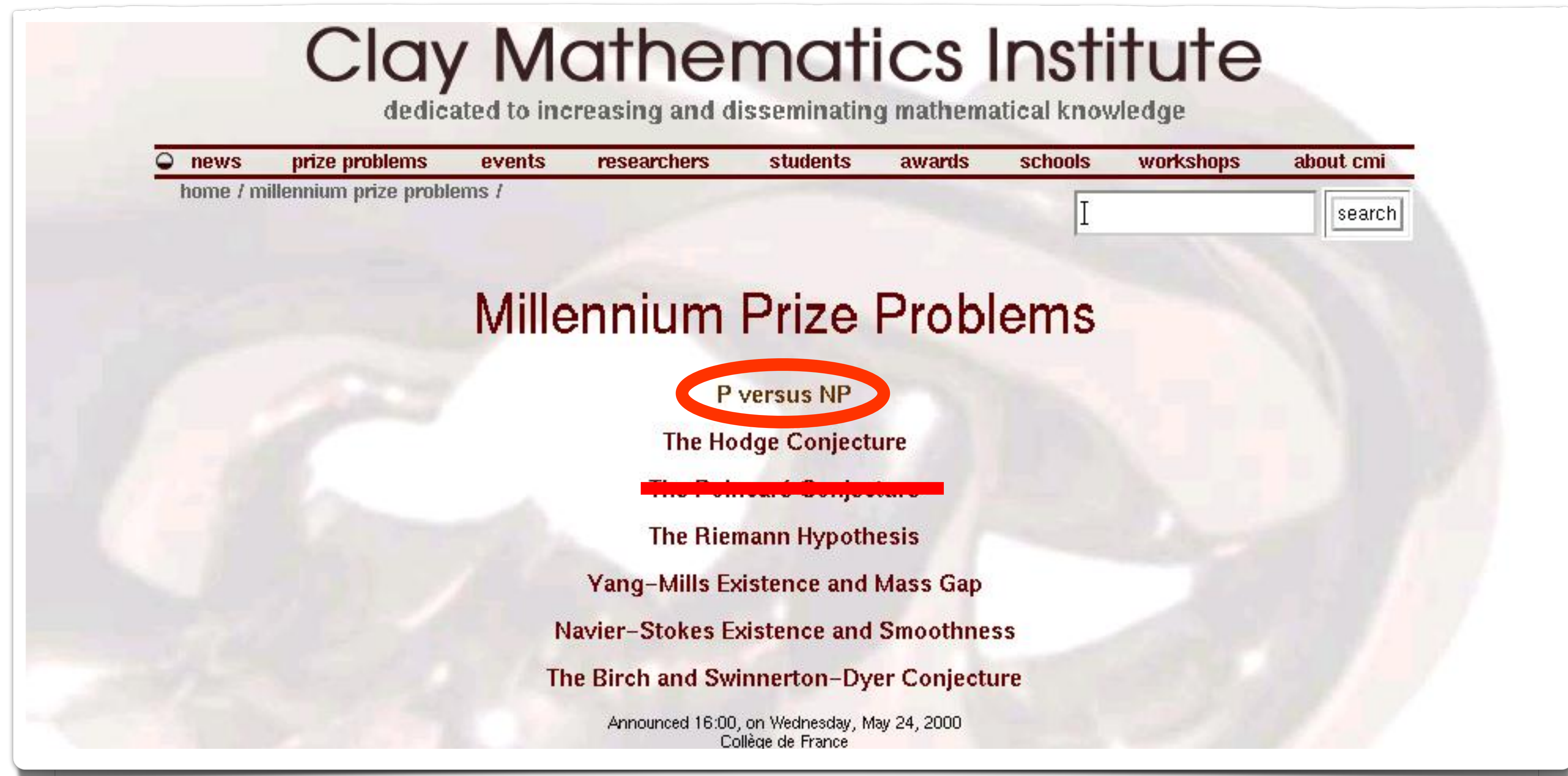


Enumeration



- Exponentiell viele Fälle!
- Kann man Arbeit sparen?

P vs. NP



5.3 Ein Beispiel mit Logik

Beispiel 5.5: Knapsack

Beispiel 5.5.

Wir betrachten die folgende Knapsack-Instanz mit $n = 12$, $z_i = p_i$, $Z = 111444$ und den folgenden Objekten:

Gibt es eine Menge $S \subseteq \{1, \dots, 12\}$ mit $\sum_{i \in S} z_i = \sum_{i \in S} p_i = Z$?

$z_1 = p_1 =$	100110
$z_2 = p_2 =$	100001
$z_3 = p_3 =$	10101
$z_4 = p_4 =$	10010
$z_5 = p_5 =$	1001
$z_6 = p_6 =$	1110
$z_7 = p_7 =$	200
$z_8 = p_8 =$	100
$z_9 = p_9 =$	20
$z_{10} = p_{10} =$	10
$z_{11} = p_{11} =$	2
$z_{12} = p_{12} =$	1

Beispiel 5.5: Beobachtungen

$$x_1 \vee \overline{x_1}$$

$$x_2 \vee \overline{x_2}$$

$$x_3 \vee \overline{x_3}$$

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

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

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

1. Ziffer: Man muss 1 oder 2 auswählen, aber nicht beide.

2. Ziffer: Man muss 3 oder 4 auswählen, aber nicht beide.

3. Ziffer: Man muss 5 oder 6 auswählen, aber nicht beide.

4. Ziffer: Man muss 1, 3 oder 6 auswählen, dann kann man mit 7 und 8 den Wert 4 erzeugen.

5. Ziffer: Man muss 1, 4 oder 6 auswählen, dann kann man mit 9 und 10 den Wert 4 erzeugen.

6. Ziffer: Man muss 2, 3 oder 5 auswählen, dann kann man mit 11 und 12 den Wert 4 erzeugen.

$$z_1 = p_1 = 100110$$

$$z_2 = p_2 = 100001$$

$$z_3 = p_3 = 10101$$

$$z_4 = p_4 = 10010$$

$$z_5 = p_5 = 1001$$

$$z_6 = p_6 = 1110$$

$$z_7 = p_7 = 200$$

$$z_8 = p_8 = 100$$

$$z_9 = p_9 = 20$$

$$z_{10} = p_{10} = 10$$

$$z_{11} = p_{11} = 2$$

$$z_{12} = p_{12} = 1$$

Beispiel 5.5: Äquivalenz

$$x_1 \vee \overline{x_1}$$

$$x_2 \vee \overline{x_2}$$

$$x_3 \vee \overline{x_3}$$

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

Konkret: Jede Lösung der logischen Formel entspricht einer Lösung der Instanz SUBSET SUM — und umgekehrt.

Allgemein: Für jede logische Formel dieser Art lässt sich schnell eine äquivalente Instanz von SUBSET SUM konstruieren.

Also: Wenn wir einen „perfekten“ Algorithmus für SUBSET SUM haben, dann können wir auch schnell entscheiden, ob eine logische Formel lösbar ist.

$z_1 = p_1 =$	100110
$z_2 = p_2 =$	100001
$z_3 = p_3 =$	10101
$z_4 = p_4 =$	10010
$z_5 = p_5 =$	1001
$z_6 = p_6 =$	1110
$z_7 = p_7 =$	200
$z_8 = p_8 =$	100
$z_9 = p_9 =$	20
$z_{10} = p_{10} =$	10
$z_{11} = p_{11} =$	2
$z_{12} = p_{12} =$	1

111444

5.4 3SAT

Das Logikproblem 3SAT

Definition 5.6. (3-Satisfiability (3SAT))

Gegeben: Eine Boolesche Formel, zusammengesetzt aus:

- n Boolesche Variablen $x_1, \dots, x_n$, aus denen wir Literale ℓ_i der Form x_k oder $\bar{x}_k$ bilden können.
- m Klauseln, jede zusammengesetzt aus genau drei Literalen $C_j = (\ell_{j,1} \vee \ell_{j,2} \vee \ell_{j,3})$ mit $1 \leq j \leq m$.

Gesucht: Eine alle m Klauseln erfüllende (engl: satisfying) Wahrheitsbelegung der n Variablen.

$$(x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$

Wie schwer sind 3SAT und KNAPSACK?

✓ 3SAT ✗

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



KNAPSACK

$z_1 = p_1 =$	111	
$z_2 = p_2 =$	100001	
$z_3 = p_3 =$	101	
$z_4 = p_4 =$	1	
$z_5 = p_5 =$	1001	
$z_6 = p_6 =$	1110	
$z_7 = p_7 =$	200	
$z_8 = p_8 =$	100	
$z_9 = p_9 =$	20	
$z_{10} = p_{10} =$	10	
$z_{11} = p_{11} =$	2	
$z_{12} = p_{12} =$	1	

Satz 5.9. Wenn 0-1-KNAPSACK $\in P$ ist, dann ist auch 3SAT $\in P$.

Satz 5.12 (Satz von Cook 1971). Wenn 3SAT $\in P$, dann gilt $P = NP$.

Korollar 5.11. Wenn Knapsack $\in P$, dann gilt $P = NP$.

5.5 Konsequenzen

Algorithmen und Investmentbanker

Ziele: Ein „perfekter“ Algorithmus liefert

1. immer

2. schnell

3. eine **optimale** Lösung.

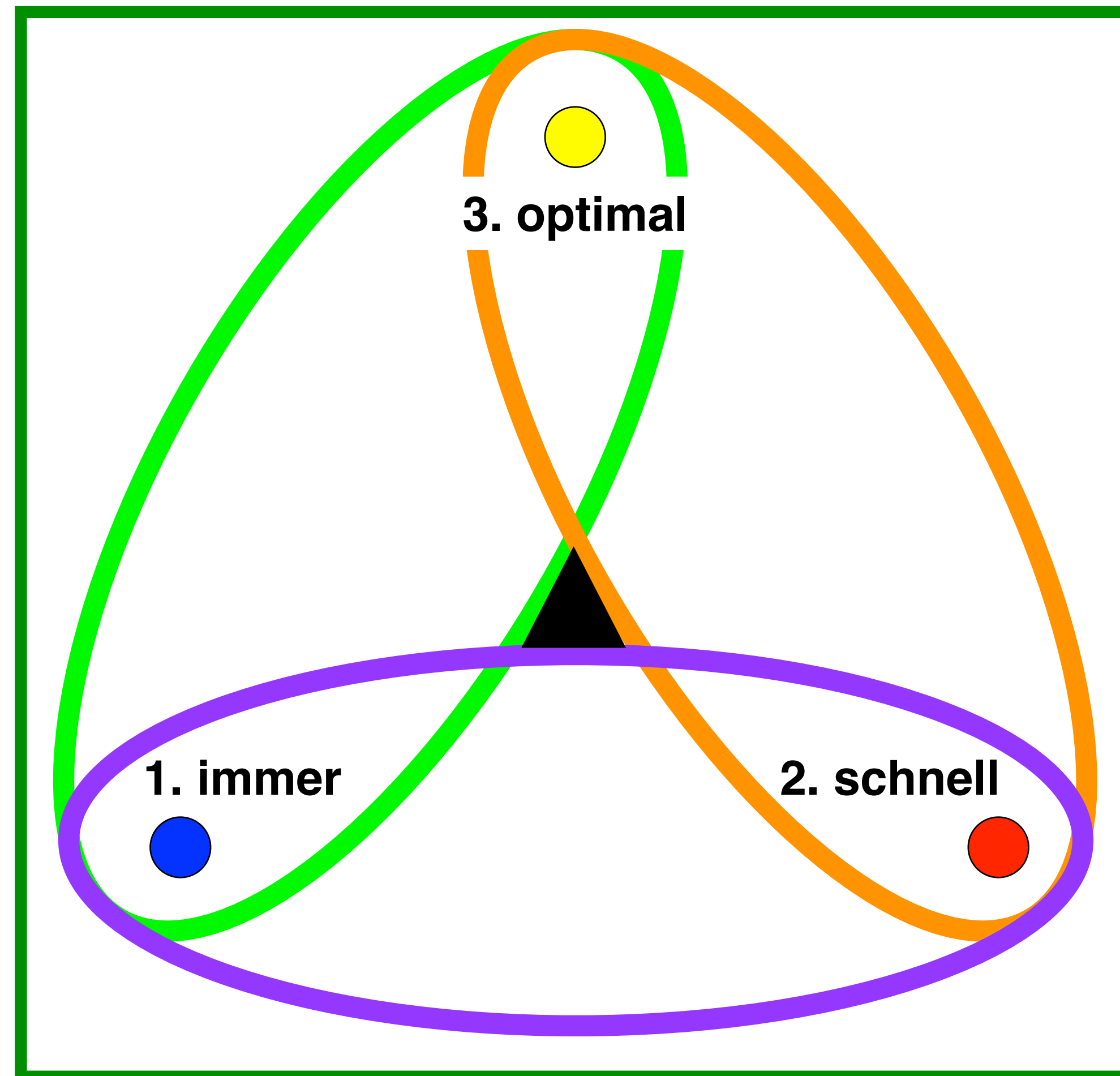
Ziele: Ein „perfekter“ Finanzberater ist ein

1. ehrlicher

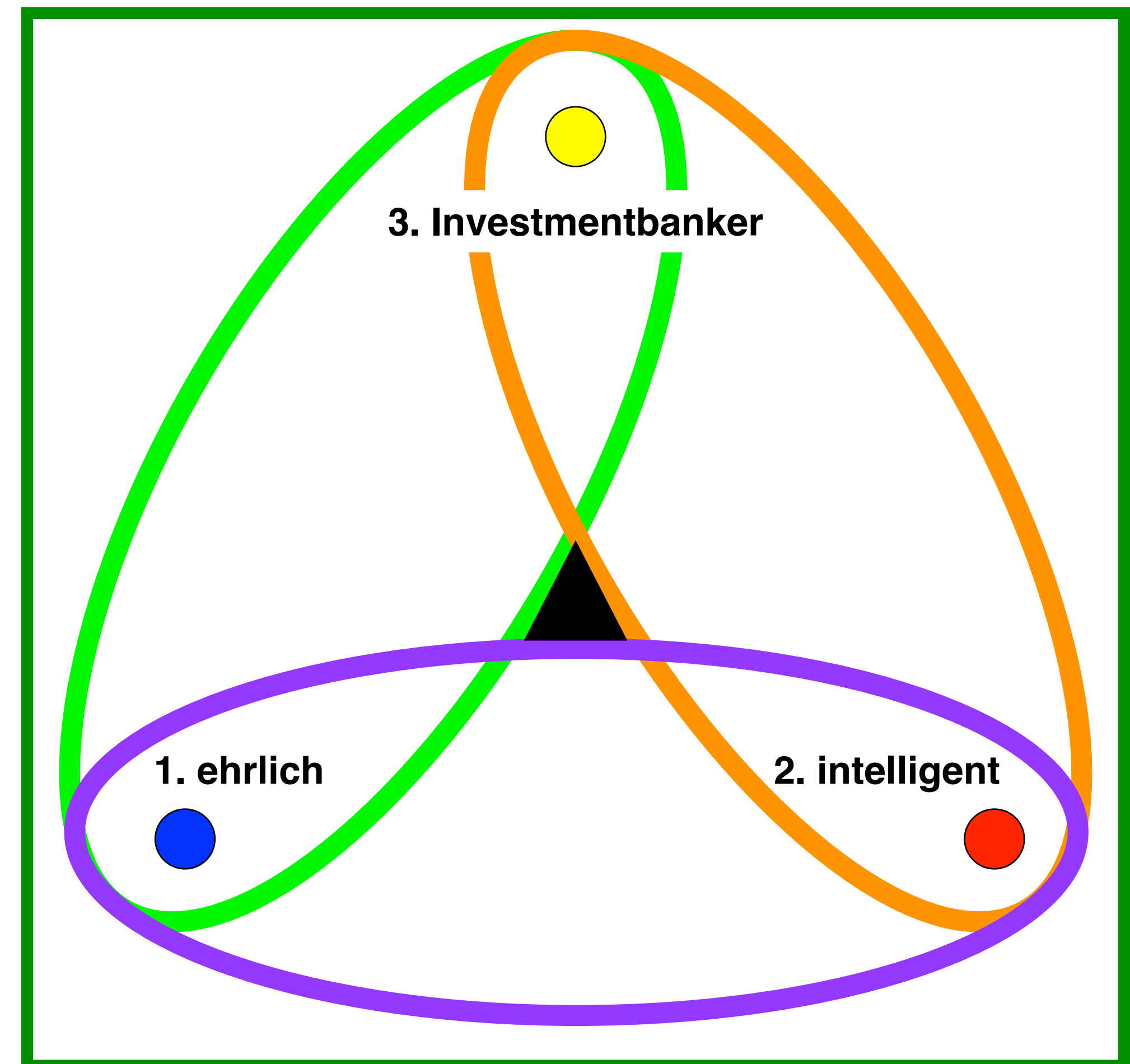
2. intelligenter

3. **Investmentbanker.**

NP-Vollständigkeit



Finanzkrise



WE OFFER THREE • KINDS OF SERVICE

GOOD • FAST • CHEAP

YOU CAN PICK ANY TWO



GOOD & CHEAP WONT BE FAST

GOOD & FAST WON'T BE CHEAP

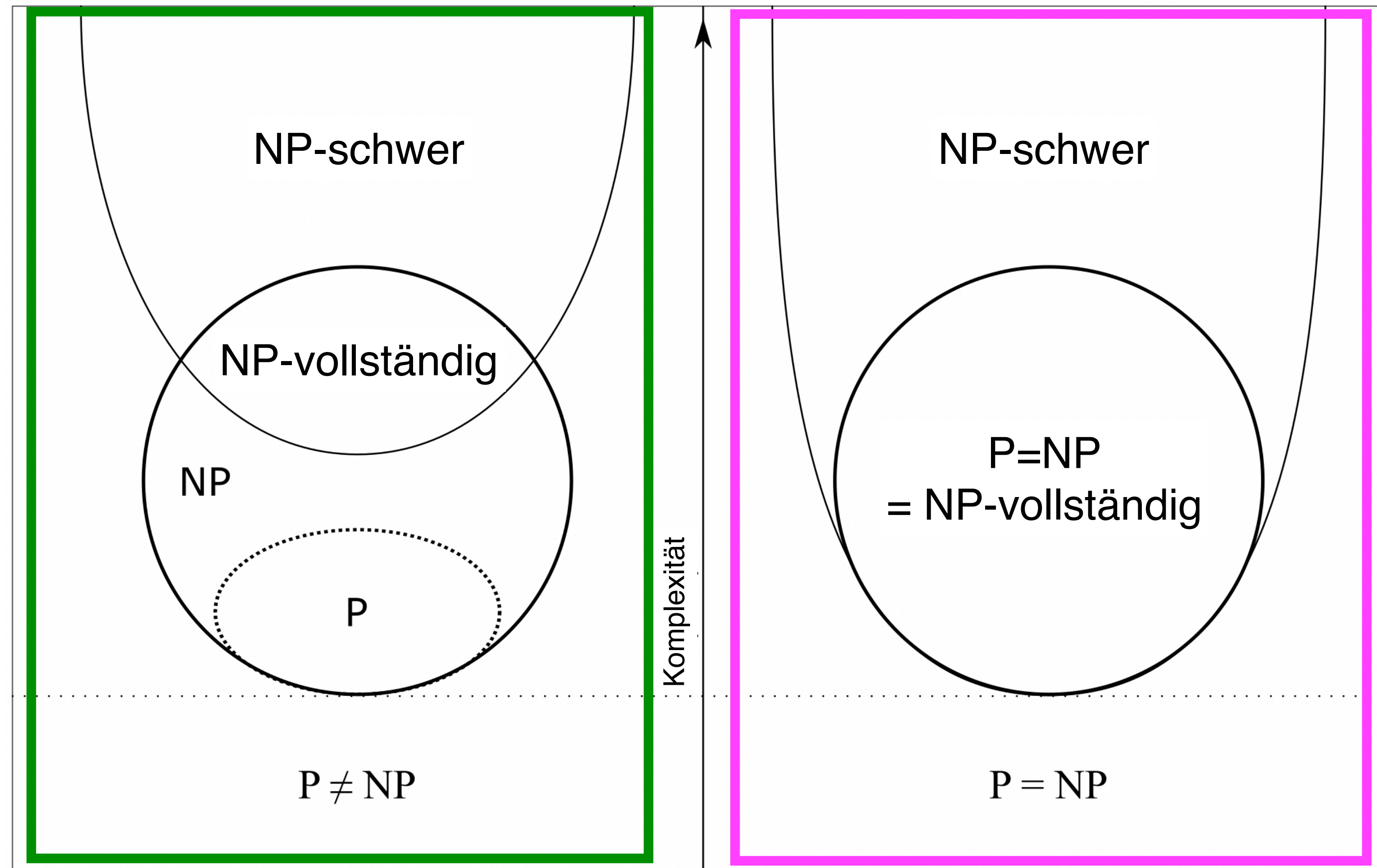
FAST & CHEAP WON'T BE GOOD



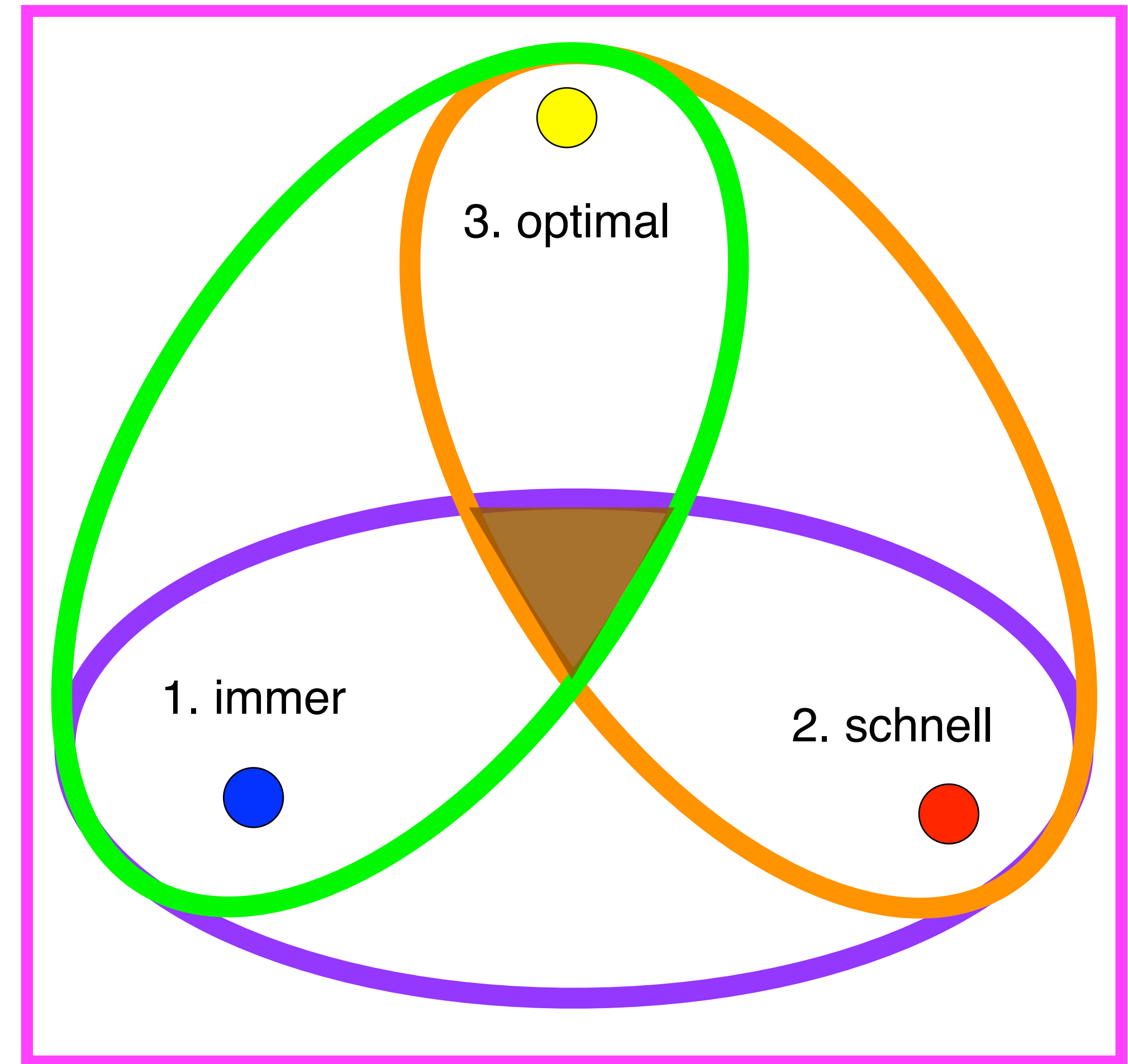
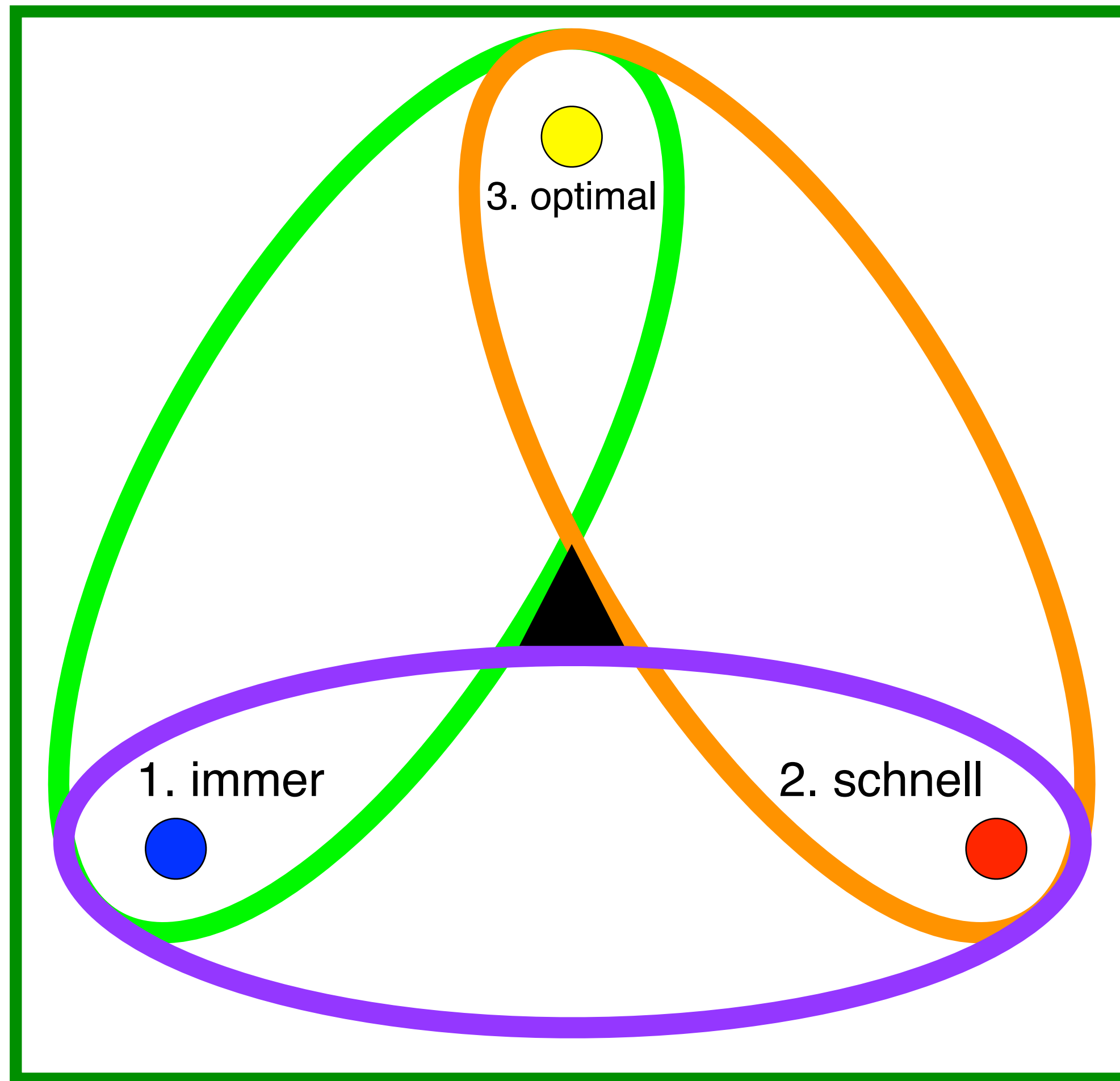
Technische
Universität
Braunschweig

Die Komplexitätslandschaft

Die Komplexitätslandschaft



Das Bild für Knapsack



Umgang mit schwierigen Situationen

Ziel: Algorithmus für NP-schweres Problem

1. immer

2. schnell

3. eine optimale Lösung.

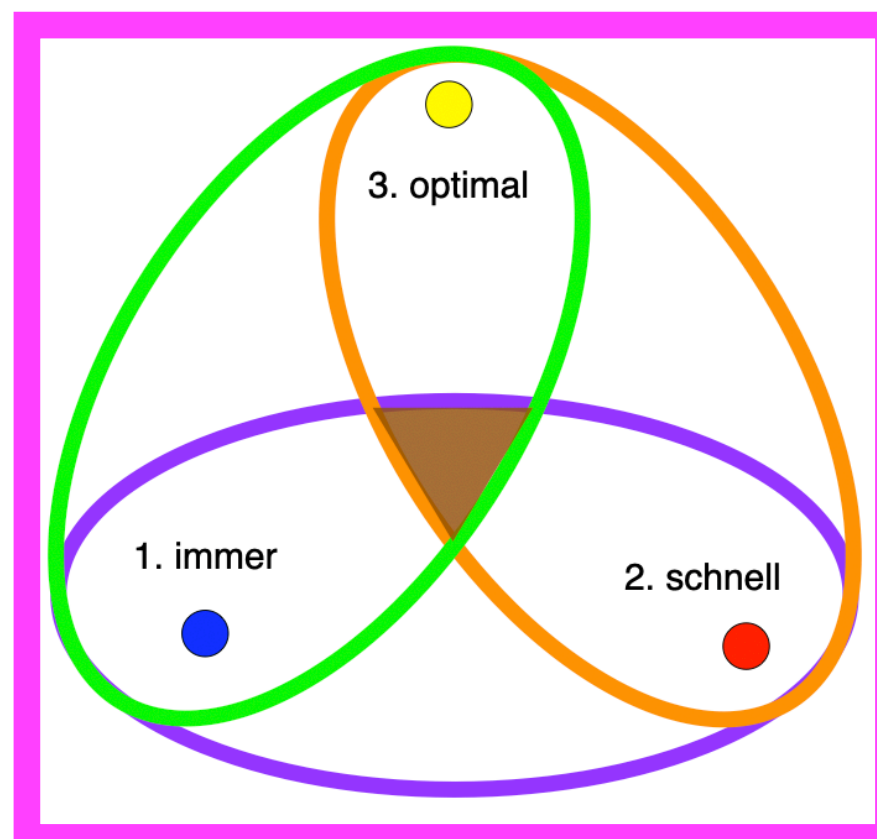
Persönlichkeitsprofile:

(A) Mit dem Schicksal hadern und diskutieren

(B) Auf Glück vertrauen

(C) Hart arbeiten

(D) Erwartungen zurückschrauben



(A) Komplexitätsanalyse: Spezialfall nicht NP-schwer?

Umgang mit schwierigen Situationen

Ziel: Algorithmus für NP-schweres Problem

1. immer

2. schnell

3. eine optimale Lösung.

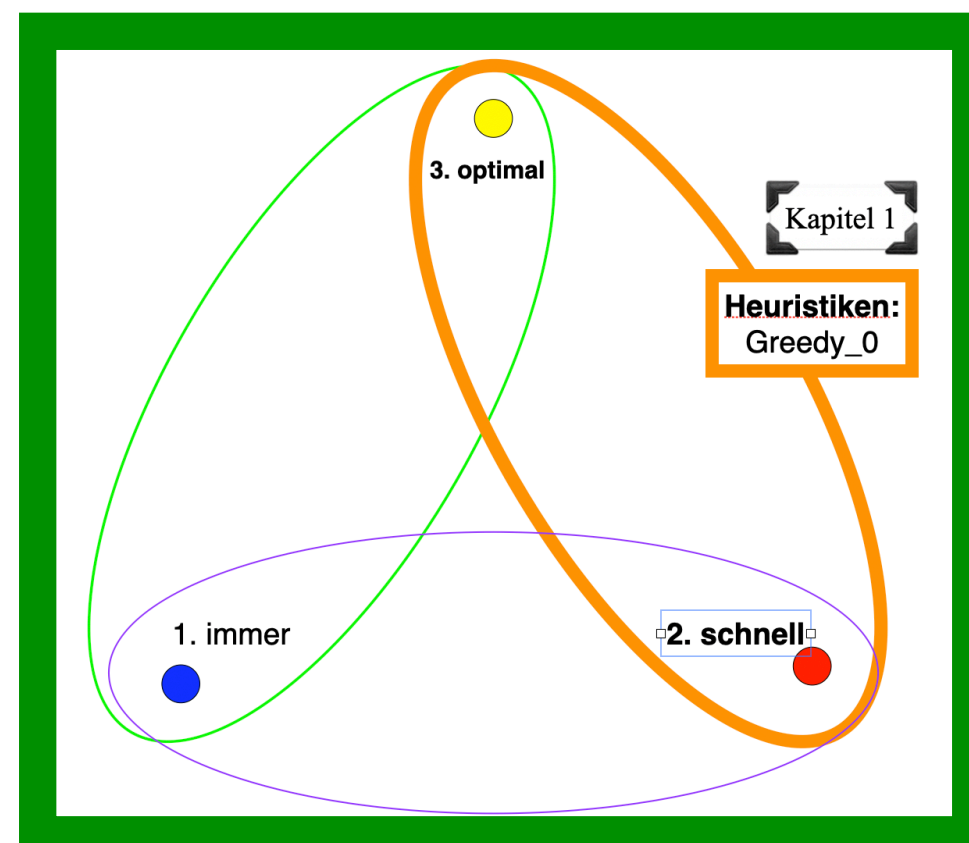
Persönlichkeitsprofile:

(A) Mit dem Schicksal hadern und diskutieren

(B) Auf Glück vertrauen

(C) Hart arbeiten

(D) Erwartungen zurückschrauben



(A) Komplexitätsanalyse: Spezialfall nicht NP-schwer?

(B) Heuristiken: raten und hoffen

Umgang mit schwierigen Situationen

Ziel: Algorithmus für NP-schweres Problem

1. immer

2. schnell

3. eine optimale Lösung.

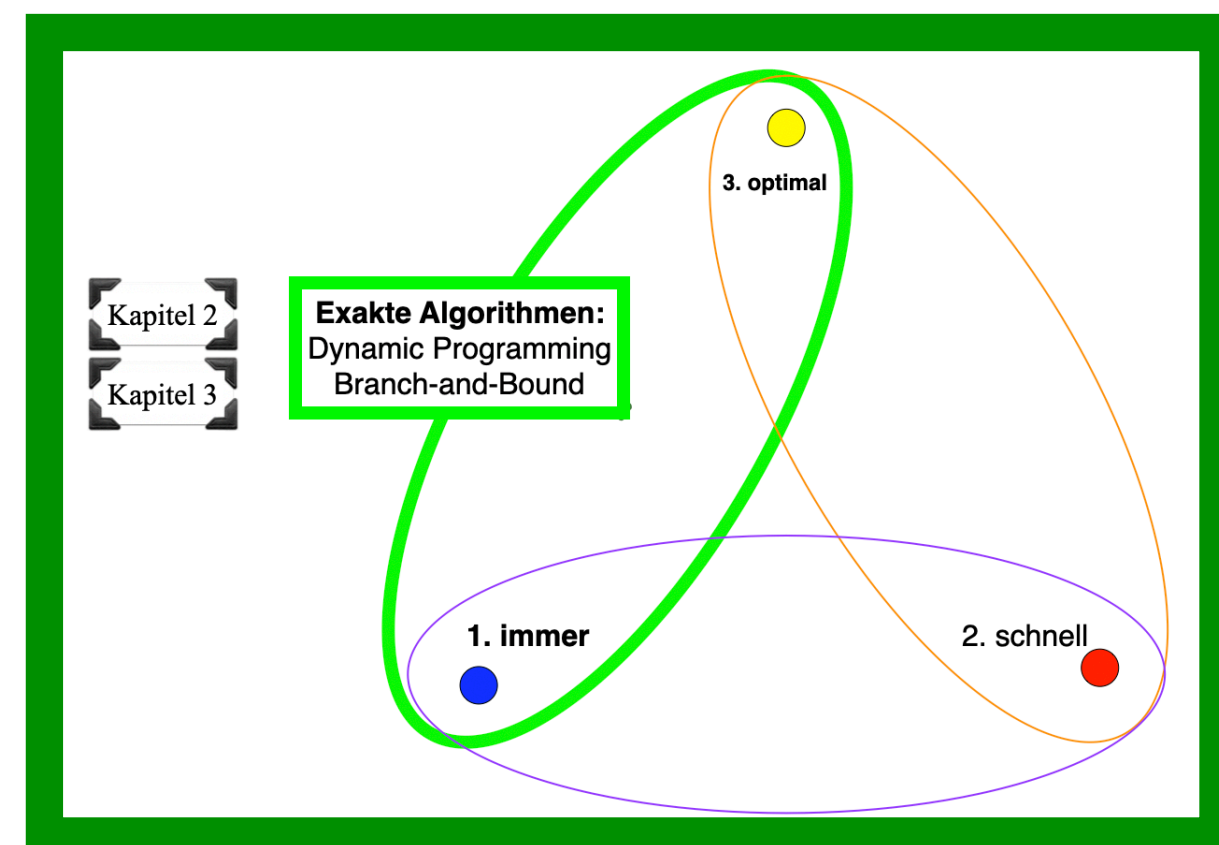
Persönlichkeitsprofile:

(A) Mit dem Schicksal hadern und diskutieren

(B) Auf Glück vertrauen

(C) Hart arbeiten

(D) Erwartungen zurückschrauben



(A) Komplexitätsanalyse: Spezialfall nicht NP-schwer?

(B) Heuristiken: raten und hoffen

(C) Exakte Algorithmen

Umgang mit schwierigen Situationen

Ziel: Algorithmus für NP-schweres Problem

1. immer

2. schnell

3. eine optimale Lösung.

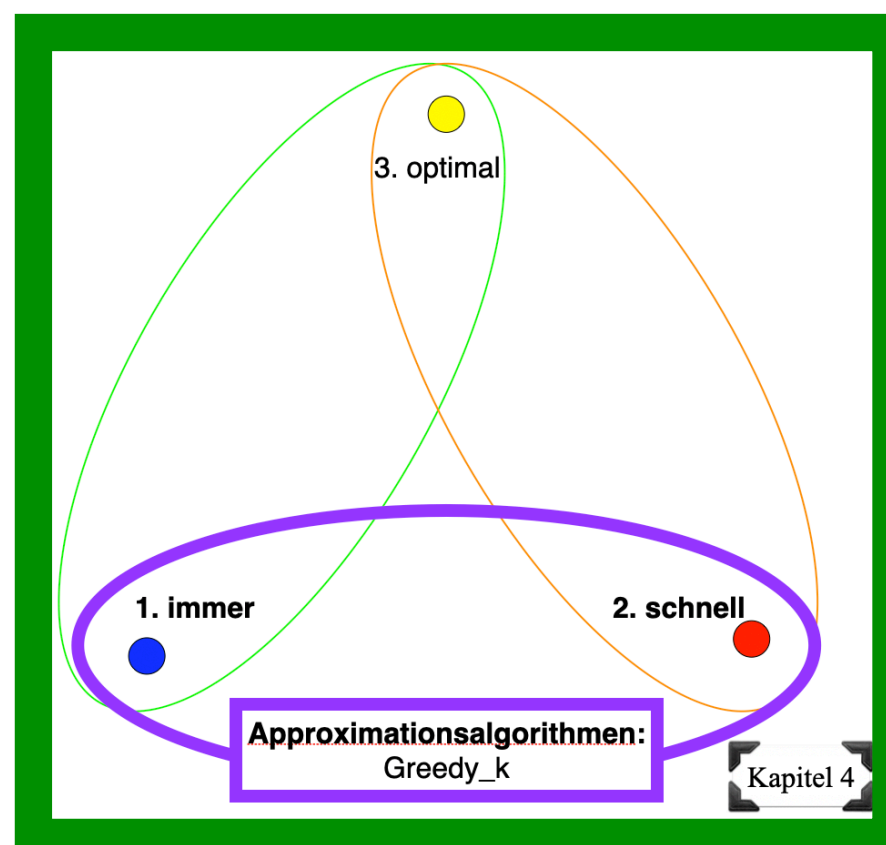
Persönlichkeitsprofile:

(A) Mit dem Schicksal hadern und diskutieren

(B) Auf Glück vertrauen

(C) Hart arbeiten

(D) Erwartungen zurückschrauben



(A) Komplexitätsanalyse: Spezialfall nicht NP-schwer?

(B) Heuristiken: raten und hoffen

(C) Exakte Algorithmen

(D) Approximationsalgorithmen: „gut“ statt „optimal“

Umgang mit schwierigen Situationen

Ziel: Algorithmus für NP-schweres Problem

1. immer

2. schnell

3. eine optimale Lösung.

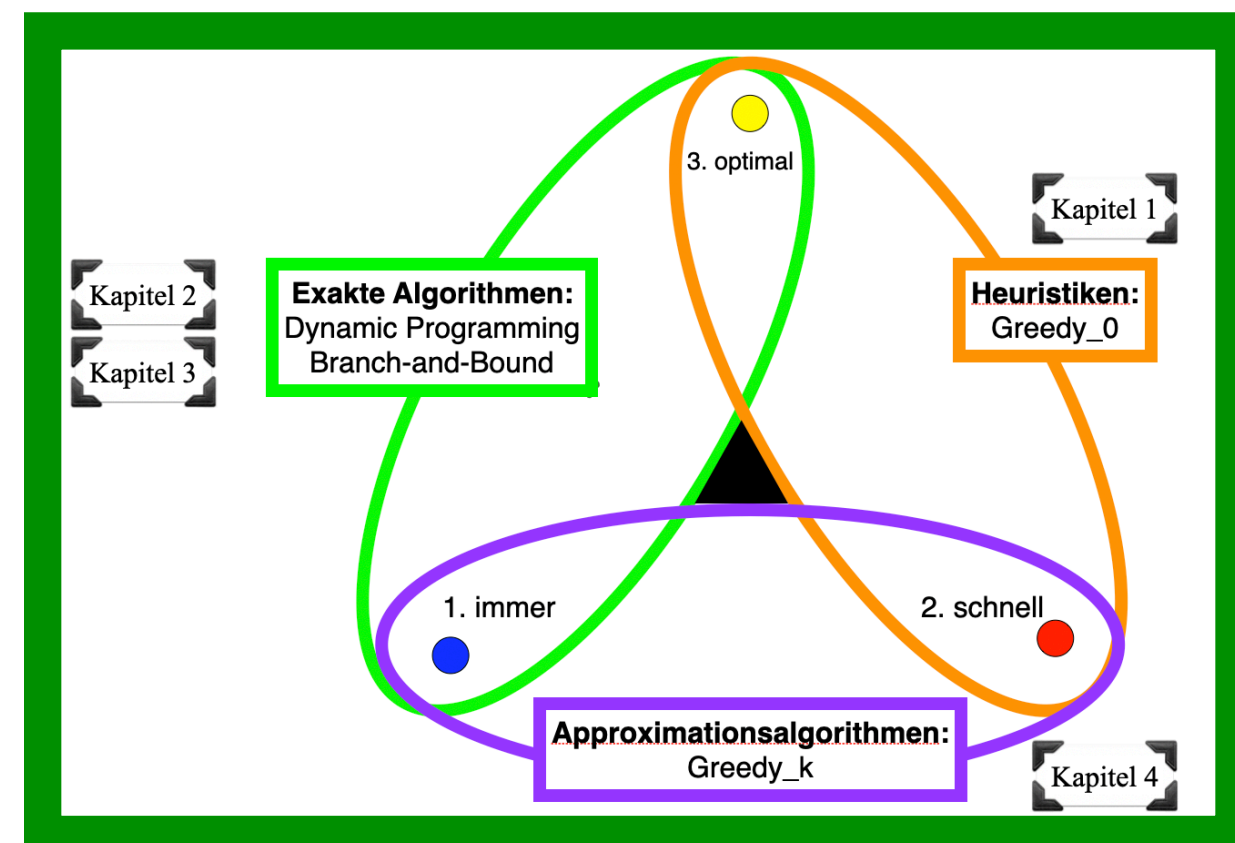
Persönlichkeitsprofile:

(A) Mit dem Schicksal hadern und diskutieren

(B) Auf Glück vertrauen

(C) Hart arbeiten

(D) Erwartungen zurückschrauben



(A) Komplexitätsanalyse: Spezialfall nicht NP-schwer?

(B) Heuristiken: raten und hoffen

(C) Exakte Algorithmen

(D) Approximationsalgorithmen: „gut“ statt „optimal“

5.6 Integer Programming und Vertex Cover

Ganzzahlige und lineare Optimierungsprobleme

Ganzzahliges Optimierungsproblem

$$\max \sum_{i=1}^n x_i p_i$$

$$\begin{array}{ccccccc} a_{11}x_1 & + \dots & + a_{1n}x_n & \leq & b_1 \\ a_{21}x_1 & + \dots & + a_{2n}x_n & \leq & b_2 \\ \vdots & \vdots & \vdots & \vdots & \\ a_{m1}x_1 & + \dots & + a_{mn}x_n & \leq & b_m \end{array}$$

$$x_i \in \{0, 1\}$$

Beweisbar schwer zu lösen

Lineares Optimierungsproblem

$$\max \sum_{i=1}^n x_i p_i$$

$$\begin{array}{ccccccc} a_{11}x_1 & + \dots & + a_{1n}x_n & \leq & b_1 \\ a_{21}x_1 & + \dots & + a_{2n}x_n & \leq & b_2 \\ \vdots & \vdots & \vdots & \vdots & \\ a_{m1}x_1 & + \dots & + a_{mn}x_n & \leq & b_m \end{array}$$

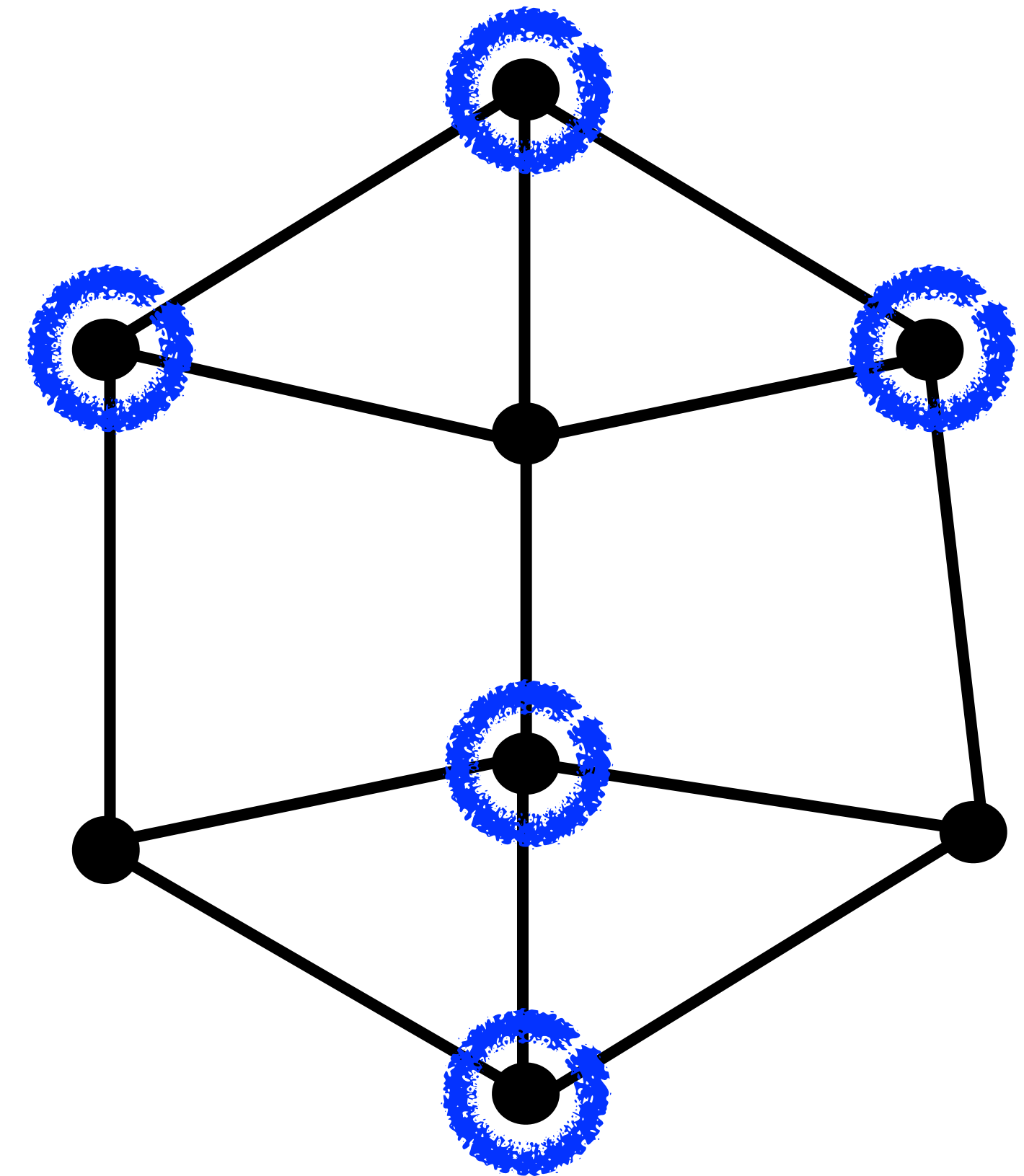
$$x_i \in [0, 1]$$

Beweisbar effizient lösbar

Vertex Cover

Gegeben: Ein Graph $G=(V,E)$

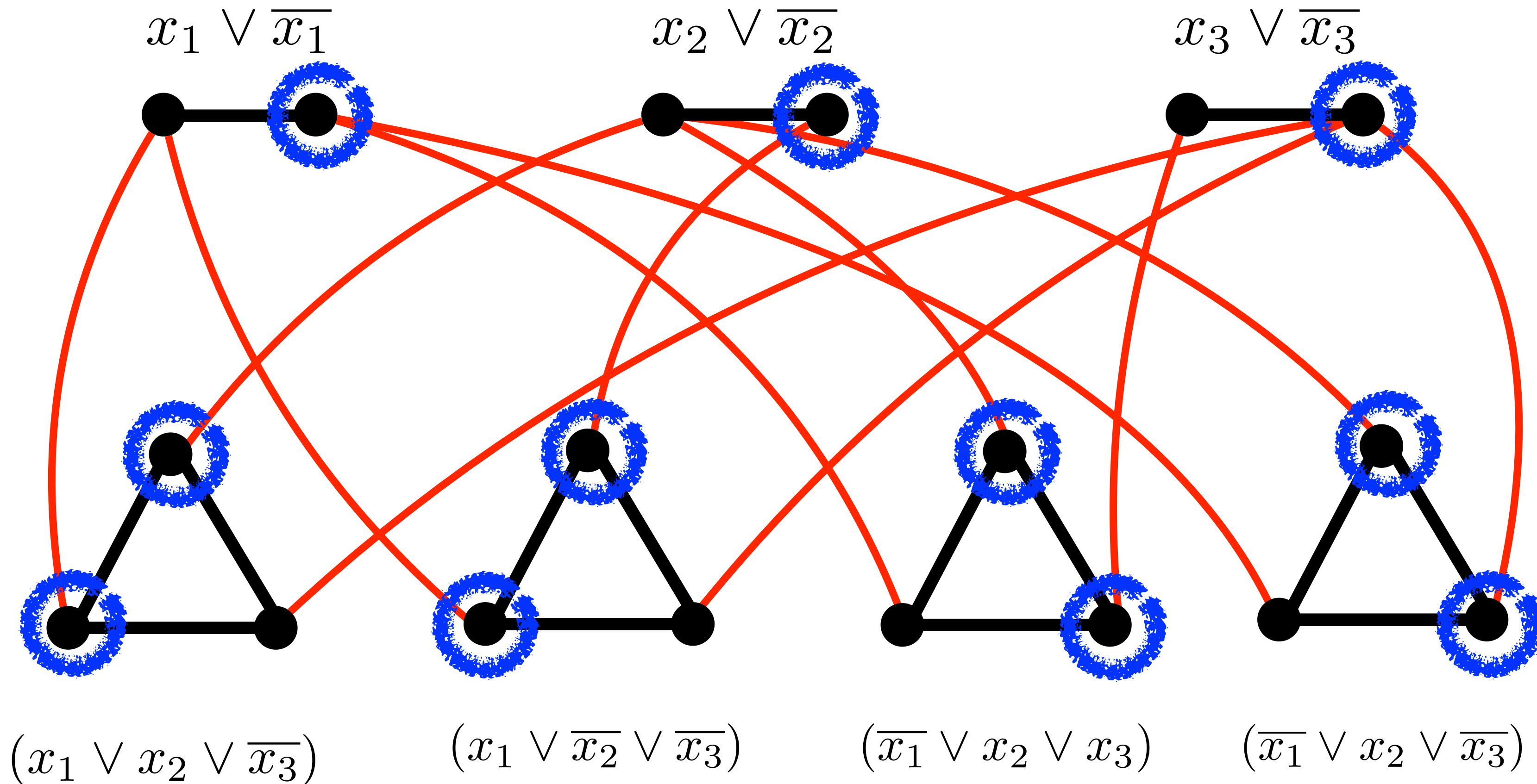
Gesucht: Ein kleinstmögliches **Vertex Cover** S in G :
eine möglichst kleine Menge von Knoten,
die alle Kanten überdecken



Von 3SAT zu Vertex Cover

n Variablen

m Klauseln

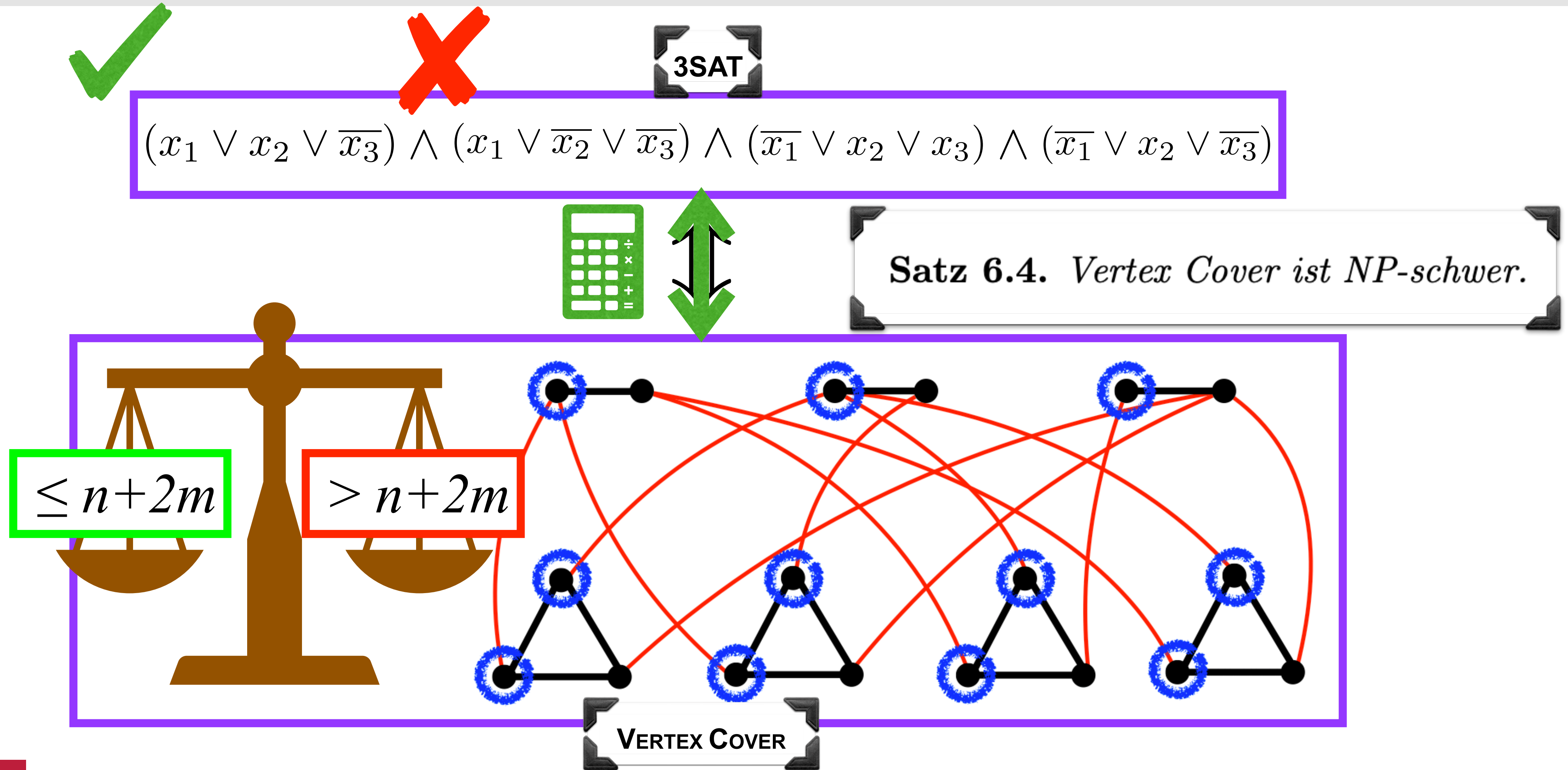


n Knoten

0 Knoten

$2m$ Knoten

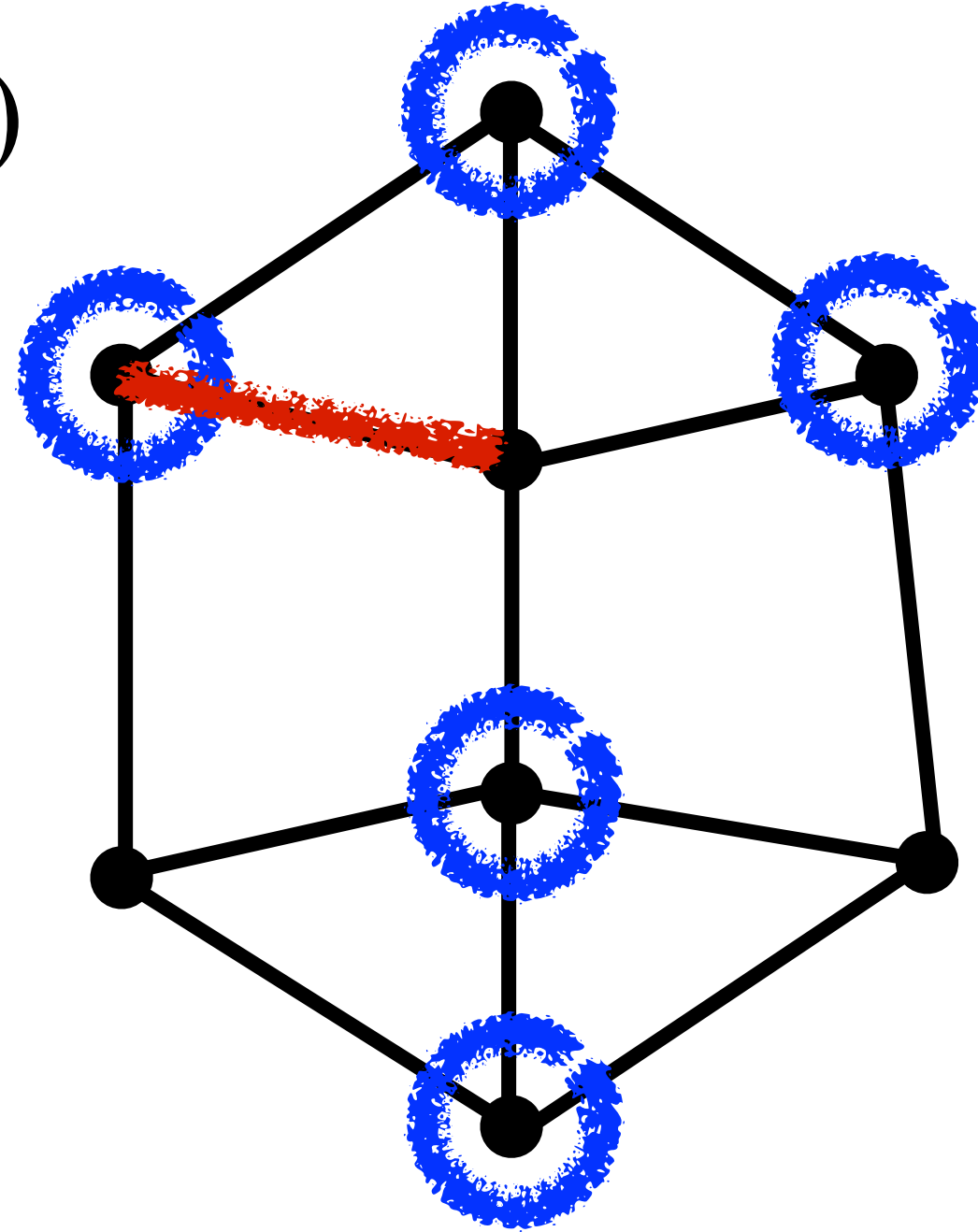
Reduktion von 3SAT auf VERTEX COVER



Vertex Cover und Integer Programming

Gegeben: Ein Graph $G=(V,E)$

$$\bigcirc y_v = 1$$



$$\begin{aligned} \min \sum_{v \in V} y_v \\ \forall e \in E : \sum_{v \in \delta(e)} y_v \geq 1 \\ y_v \in \{0, 1\} \end{aligned}$$

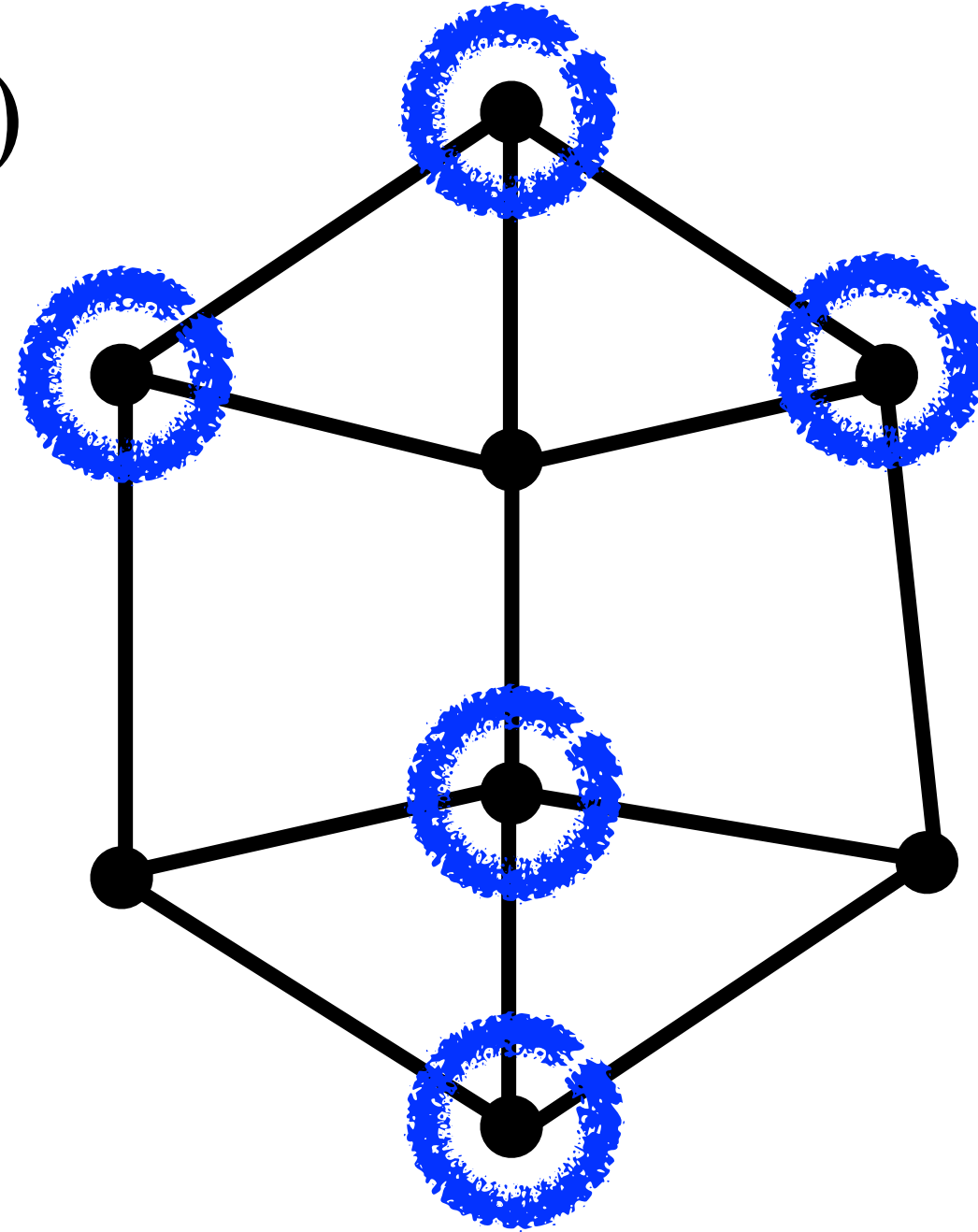
Gesucht: Ein kleinstmögliches **Vertex Cover** S in G :
eine möglichst kleine Menge von Knoten,
die alle Kanten überdecken

Vertex Cover und Integer Programming

Gegeben: Ein Graph $G=(V,E)$

$$\bigcirc y_v = 1$$

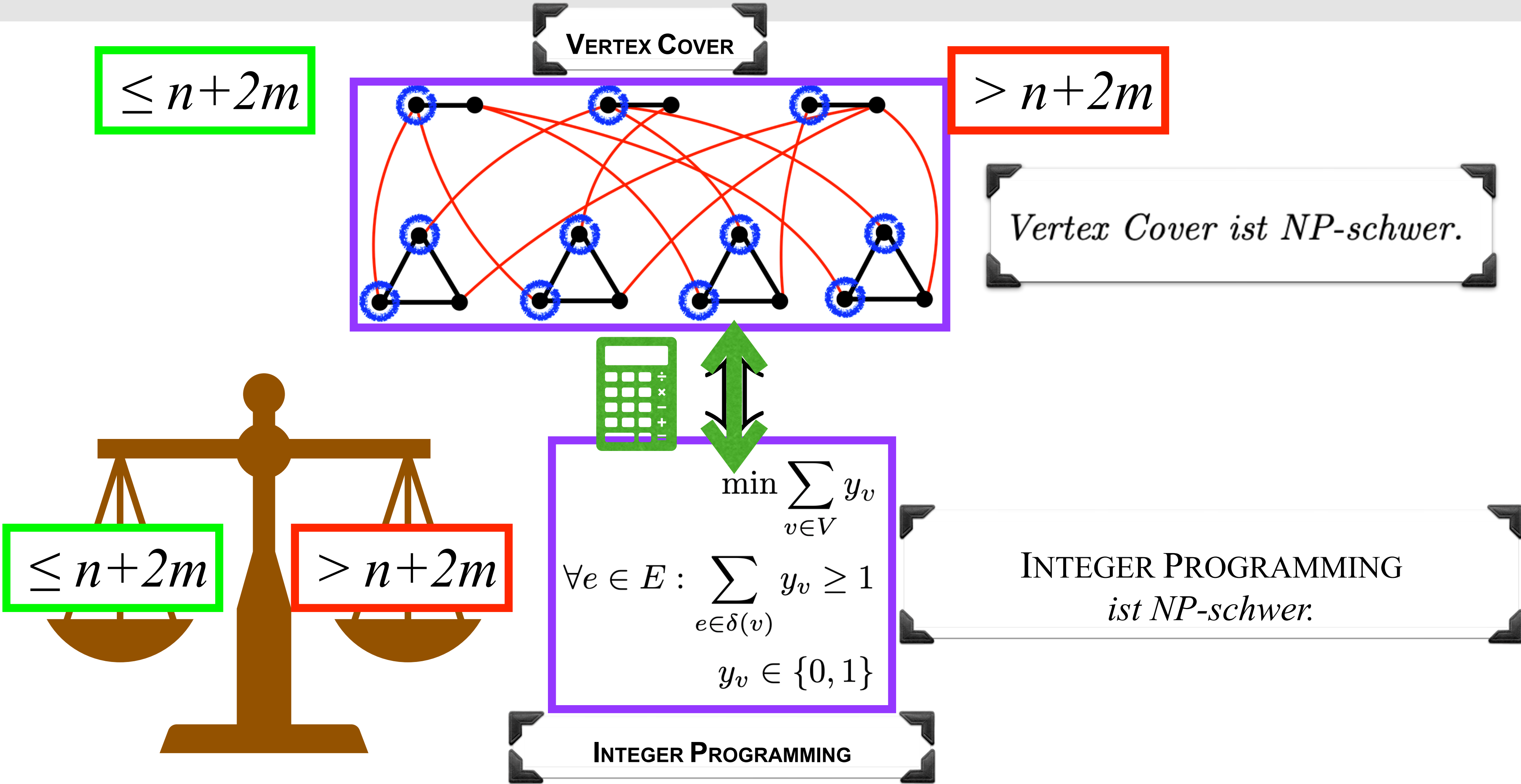
$$\sum_{v \in V} y_v = 5$$



$$\begin{aligned} & \min \sum_{v \in V} y_v \\ & \forall e \in E : \sum_{v \in \delta(e)} y_v \geq 1 \\ & y_v \in \{0, 1\} \end{aligned}$$

Gesucht: Ein kleinstmögliches **Vertex Cover** S in G :
eine möglichst kleine Menge von Knoten,
die alle Kanten überdecken

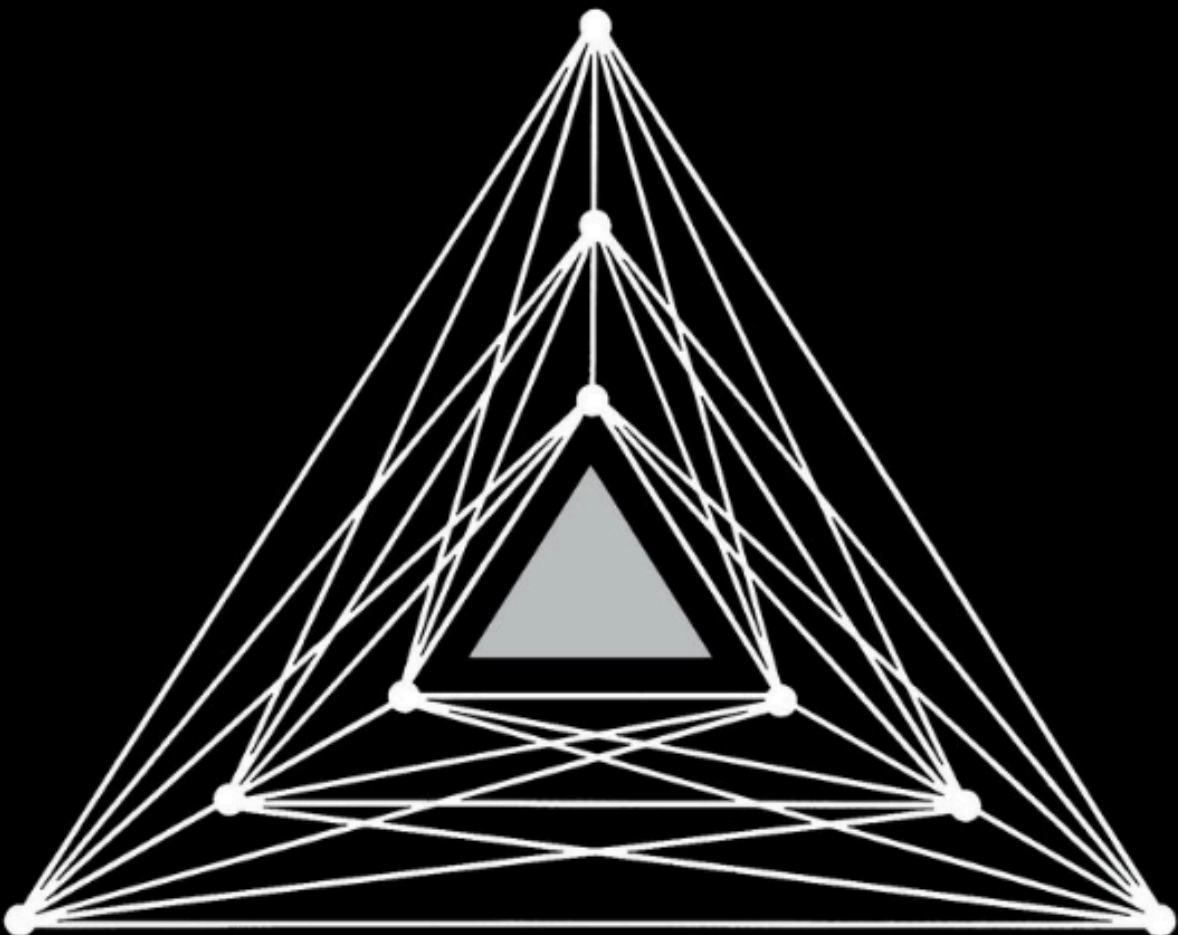
Reduktion von VERTEX COVER auf INTEGER PROGRAMMING



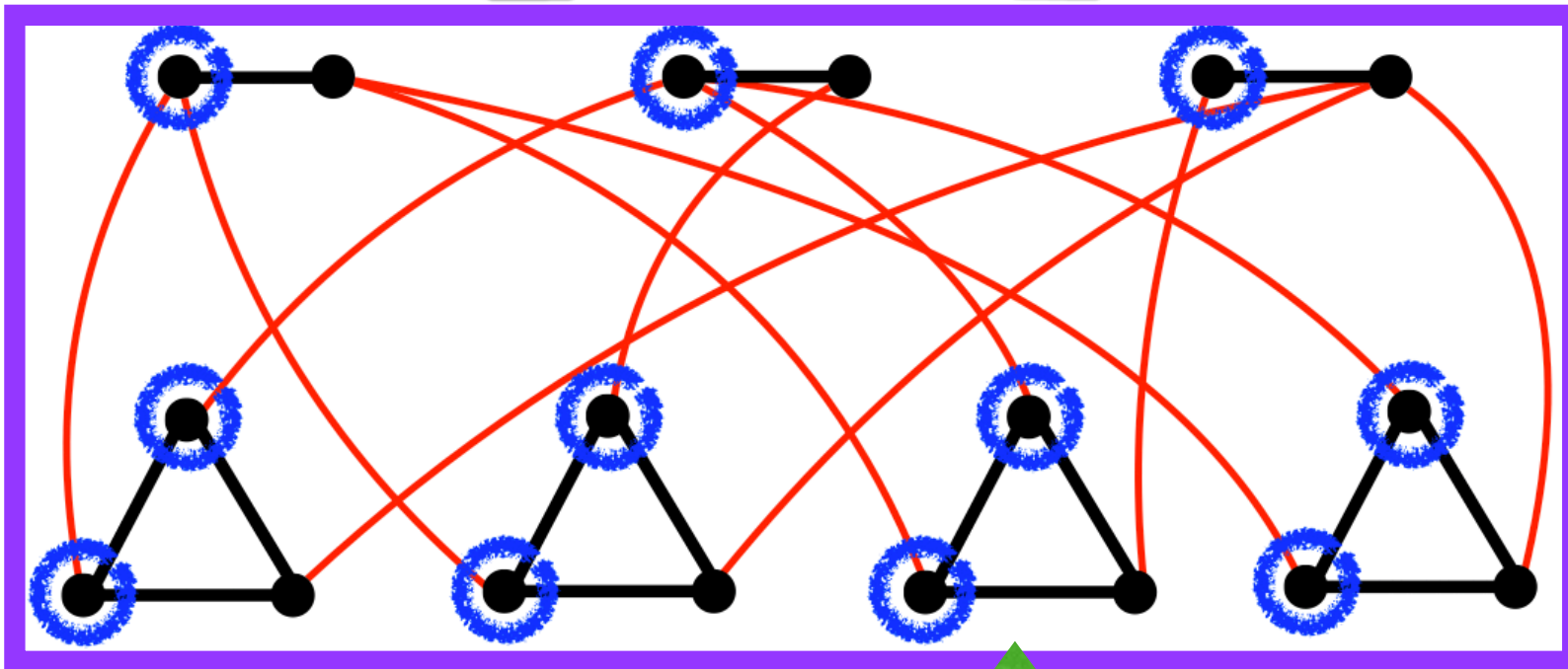
Reduktion von VERTEX COVER auf INTEGER PROGRAMMING

COMPUTERS AND INTRACTABILITY
A Guide to the Theory of NP-Completeness

Michael R. Garey / David S. Johnson

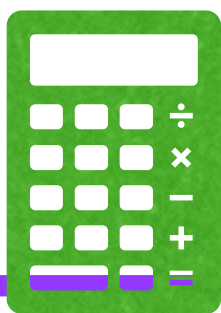


VERTEX COVER



$$> n + 2m$$

Vertex Cover ist NP-schwer.



$$> n + 2m$$

$$\min \sum_{v \in V} y_v$$

$$\forall e \in E : \sum_{e \in \delta(v)} y_v \geq 1$$
$$y_v \in \{0, 1\}$$

INTEGER PROGRAMMING
ist NP-schwer.

INTEGER PROGRAMMING

Ganzzahlige und lineare Optimierungsprobleme

Ganzzahliges Optimierungsproblem

$$\max \sum_{i=1}^n x_i p_i$$

$$\begin{array}{ccccccc} a_{11}x_1 & + \dots & + a_{1n}x_n & \leq & b_1 \\ a_{21}x_1 & + \dots & + a_{2n}x_n & \leq & b_2 \\ \vdots & \vdots & \vdots & \vdots & \\ a_{m1}x_1 & + \dots & + a_{mn}x_n & \leq & b_m \end{array}$$

$$x_i \in \{0, 1\}$$

Beweisbar schwer zu lösen

Lineares Optimierungsproblem

$$\max \sum_{i=1}^n x_i p_i$$

$$\begin{array}{ccccccc} a_{11}x_1 & + \dots & + a_{1n}x_n & \leq & b_1 \\ a_{21}x_1 & + \dots & + a_{2n}x_n & \leq & b_2 \\ \vdots & \vdots & \vdots & \vdots & \\ a_{m1}x_1 & + \dots & + a_{mn}x_n & \leq & b_m \end{array}$$

$$x_i \in [0, 1]$$

Beweisbar effizient lösbar

Linear und Integer Programming: Branch-and-Bound

ECONOMETRICA

VOLUME 28

July, 1960

NUMBER 3

AN AUTOMATIC METHOD OF SOLVING DISCRETE PROGRAMMING PROBLEMS

By A. H. LAND AND A. G. DOIG

In the classical linear programming problem the behaviour of continuous, nonnegative variables subject to a system of linear inequalities is investigated. One possible generalization of this problem is to relax the continuity condition on the variables. This paper presents a simple numerical algorithm for the solution of programming problems in which some or all of the variables can take only discrete values. The algorithm requires no special techniques beyond those used in ordinary linear programming, and lends itself to automatic computing. Its use is illustrated on two numerical examples.

1. INTRODUCTION

THERE IS A growing literature [1, 3, 5, 6] about optimization problems which could be formulated as linear programming problems with additional constraints that some or all of the variables may take only integral values. This form of linear programming arises whenever there are indivisibilities. It is not meaningful, for instance, to schedule 3–7/10 flights between two cities, or to undertake only 1/4 of the necessary setting up operation for running a job through a machine shop. Yet it is basic to linear programming that the variables are free to take on any positive value,¹ and this sort of answer is very likely to turn up.

In some cases, notably those which can be expressed as transport problems, the linear programming solution will itself yield discrete values of the variables. In other cases the percentage change in the maximand² from common sense rounding of the variables is sufficiently small to be neglected. But there remain many problems where the discrete variable constraints are significant and costly.

Until recently there was no general automatic routine for solving such problems, as opposed to procedures for proving the optimality of conjectured solutions, and the work reported here is intended to fill the gap. About the time of its completion an alternative method was proposed by Gomory [5] and subsequently extended by Beale [1]. Gomory's method

¹ Or more generally, any value within a bounded interval.

² We shall speak throughout of maximisation, but of course an exactly analogous argument applies to minimisation.

1. INTRODUCTION

THERE IS A growing literature [1, 3, 5, 6] about optimization problems which could be formulated as linear programming problems with additional constraints that some or all of the variables may take only integral values. This form of linear programming arises whenever there are indivisibilities. It is not meaningful, for instance, to schedule 3–7/10 flights between two cities, or to undertake only 1/4 of the necessary setting up operation for running a job through a machine shop. Yet it is basic to linear programming that the variables are free to take on any positive value,¹ and this sort of answer is very likely to turn up.

Vielen Dank!

s.fekete@tu-bs.de