

AAK Cheatsheet – Problem-Arsenal

Referenz: Definition · Beispiel · Reduktion · Konstruktion · ETH · Beweisidee je Richtung

SAT

Def: KNF-Formel φ ; existiert erfüllende Belegung?

Bsp: $(x \vee y) \wedge (\bar{x} \vee y \vee z) \wedge (\bar{y})$ – ja: $x=0, y=0, z=1$.

3-SAT

Def: KNF, jede Klausel ≤ 3 Literale; erfüllbar?

Bsp: $(x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)$ – ja: $x_1=1, x_2=0$.

Reduktion: von SAT.

Konstruktion: Lange Klausel $(\ell_1 \vee \dots \vee \ell_k)$ mit frischen y_i zerhacken: $(\ell_1 \vee \ell_2 \vee y_1)(\bar{y}_1 \vee \ell_3 \vee y_2) \dots$

$\Rightarrow$: Erfülltes ℓ_j wählen, y -Kette davor wahr, danach falsch setzen.

$\Leftarrow$: Wären alle ℓ_j falsch, erzwingt die y -Kette einen Widerspruch (Dominoeffekt).

ETH: $n, m, n \leq 3m$.

k -CLIQUE

Def: Graph G , Zahl k ; existiert paarweise verbundene Knotenmenge $|C| \geq k$?

Bsp: Dreieck $\{1, 2, 3\}$ plus Kante $\{3, 4\}$: $k=3$ ja, $k=4$ nein.

Reduktion: von SAT (Vorlesung).

Konstruktion:

– $V' = \{(j, \ell) \mid j \in [m], \ell \in C_j\}$

– $E' = \{\{(i, \ell), (j, \ell')\} \mid i, j \in [m], i \neq j, \ell \neq \bar{\ell}'\}$

– $k' = m$

$\Rightarrow$: Erfüllende Belegung: pro Klausel ein wahres Literal wählen – paarweise kompatibel $\rightarrow m$ -Clique.

$\Leftarrow$: m -Clique hat je Klausel genau einen Knoten; Literale widerspruchsfrei $\rightarrow$ Belegung erfüllt alle Klauseln.

ETH: $|V| = O(m)$, $|E| = O(m^2)$, $k = m$.

k -INDEPENDENTSET

Def: Existieren k paarweise *nicht* verbundene Knoten?

Bsp: Pfad $1-2-3-4$: $\{1, 3\}$ oder $\{1, 4\}$, $k=2$ ja.

Reduktion: von k -CLIQUE.

ETH: $|V'| = |V|$, $|E'| \leq |V|^2$, $k' = k$.

VERTEXCOVER

Def: Existiert S , $|S| \leq k$, die jede Kante mit einem Endpunkt trifft?

Bsp: Stern (Zentrum z , 4 Blätter): $S = \{z\}$, $k=1$ ja.

Reduktion: von k -CLIQUE (Präsenz 10.2).

Konstruktion:

– $V' = V$

– $E' = \bar{E}$

– $k' = n - k$

$\Rightarrow$: Clique C in $G \rightarrow V \setminus C$ deckt $\bar{G}$: in $\bar{G}$ gibt es keine Kante innerhalb C .

$\Leftarrow$: VC V' in $\bar{G} \rightarrow V \setminus V'$ ist IS in $\bar{G}$ = Clique in G , Größe $\geq n - (n - k) = k$.

ETH: $|V'| = |V|$, $|E'| \leq |V|^2$, $k' \leq |V|$.

FEEDBACKVERTEXSET (gerichtet)

Def: $X \subseteq V$, $|X| \leq k$, sodass $G \setminus X$ kreisfrei.

Bsp: Kreis $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$: $X = \{1\}$, $k=1$ ja.

Reduktion: von VERTEXCOVER (HA 10.1).

Konstruktion:

– $V' = V$

– $E' = \{(u, v), (v, u) \mid \{u, v\} \in E\}$

– $k' = k$

$\Rightarrow$: VC entfernt $\rightarrow$ keine Kanten mehr $\rightarrow$ kreisfrei.

$\Leftarrow$: Wäre Kante $\{u, v\}$ ungedeckt, bildet (u, v, u) einen 2-Kreis – Widerspruch.

ETH: $|V'| = |V|$, $|E'| = O(|E|)$, $k' = k$.

Δ COVER (Dreiecksüberdeckung)

Def: C_Δ , $|C_\Delta| \leq k$: jedes Dreieck enthält einen Knoten aus C_Δ .

Bsp: Zwei Dreiecke mit gemeinsamer Kante $\{u, v\}$: $C_\Delta = \{u\}$, $k=1$ ja.

Reduktion: von VERTEXCOVER (HA 11.1).

Konstruktion:

– $V' = V \cup \{v_e \mid e \in E\}$

– $E' = E \cup \{\{v_e, u\}, \{v_e, v\} \mid e = \{u, v\} \in E\}$

– $k' = k$

$\Rightarrow$: VC trifft jede Kante, also jedes Alt-Dreieck *und* jedes neue Dreieck $\{u, v, v_e\}$.

$\Leftarrow$: ObdA $C_\Delta \subseteq V$ (v_e durch e_1 tauschen, deckt weiter); dann trifft C_Δ jedes $\{u, v, v_e\}$, also jede Kante.

ETH: $|V'| = O(|V|^2)$, $|E'| = O(|E|)$.

HAMILTONIANCYCLE (HC)

Def: Kreis, der jeden Knoten genau einmal besucht?

Bsp: C_5 (Fünfeck): ja. $K_{1,3}$ (Stern): nein.

Reduktion: von HP.

Konstruktion:

– $V' = V \cup \{u\}$

– $E' = E \cup \{\{u, v\} \mid v \in V\}$

$\Rightarrow$: Ham-Pfad $v_1, \dots, v_n \rightarrow$ Kreis $u, v_1, \dots, v_n, u$ (beide Anschlusskanten existieren via u).

$\Leftarrow$: HC in G' besucht u genau einmal; u herausschneiden $\rightarrow$ Ham-Pfad über alle Altknoten in G .

ETH: $|V'| = O(|V|)$, $|E'| = O(|V|^2)$.

HAMILTONIANPATH (HP)

Def: Pfad, der jeden Knoten genau einmal besucht?

Bsp: Pfad 1–2–3: ja. $K_{1,3}$ (Stern): nein.

Reduktion: von HC.

Konstruktion: Wähle v .

– $V' = V \cup \{v^*, s, t\}$

– $E' = E \cup \{\{v^*, w\} \mid \{v, w\} \in E\} \cup \{\{s, v\}, \{t, v^*\}\}$

$\Rightarrow$: HC bei v aufschneiden $\rightarrow$ Pfad $s, v, \dots, v^*, t$.

$\Leftarrow$: Pfad muss s, t als Enden haben; $v \dots v^*$ zusammenkleben ergibt HC.

ETH: $|V'| = O(|V|)$, $|E'| = O(|V|^2)$.

TSP (Entscheidung)

Def: Vollständiger Graph mit Distanzen, Budget L ; Rundreise $\leq L$?

Bsp: 4 Städte im Quadrat, Seitenlänge 1: OPT = 4.

Reduktion: von HC (Skript).

Konstruktion:

– $d(u, v) = 1$ falls $\{u, v\} \in E$, sonst $|V| + 1$

– $L = |V|$

$\Rightarrow$: HC $\rightarrow$ Tour aus lauter 1er-Kanten, Länge $|V| = L$.

$\Leftarrow$: Schon eine Nicht-Kante kostet $|V| + 1 > L \rightarrow$ Tour $\leq L$ nutzt nur Kanten aus $E \rightarrow$ HC.

ETH: $|V'| = |V|$, $|E'| = O(|V|^2)$, $L = |V|$.

k -COLOR

Def: Färbung $f : V \rightarrow [k]$ mit $f(u) \neq f(v)$ für Kanten?

Bsp: C_5 : 3 Farben nötig; C_4 : 2 reichen.

Reduktion: von 3-SAT (Skript, vollständig).

Konstruktion:

- $V' = \{x_i, \bar{x}_i, v_i \mid i \in [n]\} \cup \{F_j \mid j \in [m]\} \cup \{z\}$
- $E' = \{\{v_i, v_j\}, \{v_i, x_j\}, \{v_i, \bar{x}_j\} \mid i \neq j\} \cup \{\{x_i, \bar{x}_i\}\} \cup \{\{\ell, F_j\} \mid \text{Literal } \ell \notin C_j\} \cup \{\{v_i, z\}, \{F_j, z\}\}$
- $k' = n+1$

Vorüberlegung: obdA $f(v_i) = i$, $f(z) = n+1$; dann $f(x_j), f(\bar{x}_j) \in \{j, n+1\}$ – kodiert die Belegung.

$\Rightarrow$: Wahres Literal bekommt Farbe i , falsches $n+1$; F_j färbt man mit der Farbe eines erfüllenden Literals (keine Kante dorthin).

$\Leftarrow$: Wäre F_j unerfüllt, sind alle Literale in F_j mit $n+1$ gefärbt; F_j sieht dann alle Farben $1, \dots, n+1$ – Widerspruch.

ETH: $|V'| = O(m)$, $|E'| = O(m^2)$.

k -COLOR-PRECOLORING

Def: Zusätzlich: vorgegebene Knoten $v_1, \dots, v_k$ mit $f(v_i) = i$.

Bsp: Dreieck v_1, v_2, w mit $k = 3$, $f(v_1)=1$, $f(v_2)=2$: w bekommt Farbe 3 – ja.

Reduktion: von k -COLOR.

Konstruktion:

- $V' = V \cup \{v_1, \dots, v_k\}$
- $E' = E \cup \{\{v_i, v_j\} \mid i \neq j\}$
- $k' = k$
- ausgezeichnete Knoten: $v_1, \dots, v_k$ (die frischen)

$\Rightarrow$: k -Färbung von G bleibt gültig; neue Clique bekommt Farben 1.. k per Definition.

$\Leftarrow$: Einschränkung der Precoloring-Färbung auf G ist k -Färbung.

ETH: $|V'| = O(|V|)$, $|E'| = O(|V|^2)$, $k' = k$.

COVERCLIQUE

Def: k paarweise disjunkte Cliquen, die V überdecken?

Bsp: C_4 (Viereck): 2 Cliquen (gegenüberliegende Kanten) – ja für $k=2$.

Reduktion: von k -COLOR (SS25-Klausur).

ETH: $|V'| = |V|$, $|E'| \leq |V|^2$, $k' = k$.

3-DIMENSIONALMATCHING (3-DM)

Def: Disjunkte Mengen W, X, Y ($|W|=|X|=|Y|=q$), Tripel $T \subseteq W \times X \times Y$; q disjunkte Tripel, die alles überdecken?

Bsp: Ehe-Analogie mit 3 Geschlechtern: perfekte Dreier-Zuordnung.

Reduktion: von SAT (Vorlesung; für ETH auf 3-SAT-Instanzen anwenden).

ETH: $|W| = O(m^2)$, $|T| = O(m^4)$.

3-EXACTCOVER (X3C)

Def: Grundmenge U ($|U| = 3q$), 3er-Mengen F ; exakte Überdeckung durch disjunkte Mengen?

Bsp: $U = \{1, \dots, 6\}$, $F = \{\{1, 2, 3\}, \{4, 5, 6\}, \{2, 3, 4\}\}$: ja (erste zwei Mengen).

Reduktion: von 3-DM.

ETH: $|U'| = O(|W|)$, $|F'| = |T|$.

SUBSETSUM

Def: Größen $a_1, \dots, a_n$, Ziel T ; Teilmenge mit Summe genau T ?

Bsp: $\{3, 5, 7, 11\}$, $T = 12$: ja ($5 + 7$).

Reduktion: von X3C (Vorlesung).

Konstruktion:

- $c_j = \sum_{u_i \in S_j} (n+1)^{i-1}$ (Menge als Bitvektor zur Basis $n+1$)
- $K = \sum_{j=0}^{3m-1} (n+1)^j$ (überall Ziffer 1)

$\Rightarrow$: Exact Cover $\rightarrow$ jede Position genau einmal überdeckt $\rightarrow$ Summe $= K$.

$\Leftarrow$: $\leq n$ Summanden, Basis $n+1 \rightarrow$ kein Übertrag $\rightarrow$ jede Position genau eine 1 $\rightarrow$ gewählte Mengen überdecken exakt.

ETH: $n' = |F|$.

SUBSETSUMCARDINALITY

Def: SUBSETSUM mit Zusatz $|S| = n/2$ (n gerade).

Bsp: $\{1, 2, 3, 8\}$, $K = 9$, $|S| = 2$: ja $(1 + 8)$.

Reduktion: von SUBSETSUM (Präsenz 11.3).

Konstruktion:

- $c'_i = c_i + 1$ für $i \in [n]$
- n Einser-Items: $c'_{n+1} = \dots = c'_{2n} = 1$
- $K' = K + n$

$\Rightarrow$: Lösung S plus $n - |S|$ Einser: Kardinalität n , Summe $K + |S| + (n - |S|) = K + n$.

$\Leftarrow$: $S = S' \cap [n]$: Einser tragen $n - |S|$ bei $\rightarrow \sum_{i \in S} c_i = K$.

ETH: $n' = O(n)$.

$(a_1 = 1)$ -SUBSETSUM

Def: SUBSETSUM, wobei $a_1 = 1$ gilt.

Bsp: $\{1, 4, 6\}$, $K = 7$: ja $(1 + 6)$.

Reduktion: von SUBSETSUM.

Konstruktion:

- $a'_1 = 1$
- $a'_{i+1} = 2c_i$ für $i \in [n]$
- $K' = 2K$

$\Rightarrow$: Lösung $S \rightarrow$ verdoppelte Items summieren zu $2K$, ohne die 1.

$\Leftarrow$: $2K$ gerade, alle Items außer der 1 gerade $\rightarrow$ 1 nie nutzbar (Parität); halbieren ergibt Summe K .

ETH: $n' = O(n)$.

SUBSETSUM MIT TEILBARKEIT (durch 3 oder 7)

Def: SUBSETSUM, jede Größe durch 3 oder durch 7 teilbar.

Bsp: $\{3, 7, 21\}$, $K = 10$: ja $(3 + 7)$.

Reduktion: von SUBSETSUM.

Konstruktion:

- $a'_i = 21 c_i$
- $K' = 21 K$

$\Rightarrow$: Lösung skaliert mit: Summe $21K$.

$\Leftarrow$: Summe $21K$ durch 21 teilen $\rightarrow$ Summe K ; $21c$ ist durch 3 und 7 teilbar.

ETH: $n' = n$.

SUBSETSUM OHNE ZWEIERPOTENZEN

Def: SUBSETSUM, keine Größe ist eine Zweierpotenz 2^t .

Bsp: $\{3, 5, 6\}$, $K = 9$: ja $(3 + 6)$.

Reduktion: von SUBSETSUM.

Konstruktion:

- $a'_i = 3 c_i$
- $K' = 3K$

$\Rightarrow$: Lösung skaliert mit: Summe $3K$.

$\Leftarrow$: Summe $3K$ durch 3 teilen; $3c$ ist nie 2^t , da $3 \nmid 2^t$.

ETH: $n' = n$.

PARTITION

Def: Zahlen in zwei Hälften gleicher Summe teilen?

Bsp: $\{1, 2, 3, 4\}$: $\{1, 4\} | \{2, 3\}$ - ja.

Reduktion: von SUBSETSUM (Skript).

Konstruktion:

- $N = \sum_{j=1}^n c_j + 1$
- $c_{n+1} = N - K$, $c_{n+2} = K + 1$

– Gesamtsumme = $2N$, Hälfte = N

⇒: Lösung S : $S \cup \{c_{n+1}\}$ summiert zu $K + (N - K) = N$.

⇐: c_{n+1}, c_{n+2} nie zusammen (Summe $N + 1 > N$); obdA $c_{n+1} \in S \rightarrow$ Rest von S summiert zu $N - (N - K) = K$.

ETH: $n' = O(n)$.

$P2 \parallel C_{\max}$ (2 Maschinen)

Def: Jobs auf 2 Maschinen, Makespan $\leq T$?

Bsp: Jobs 3, 3, 2, 2, 2, $T = 6$: ja ($\{3, 3\}$ und $\{2, 2, 2\}$).

Reduktion: von PARTITION.

Konstruktion:

– Jobs = Zahlen

– $m = 2$

– $T = \frac{1}{2} \sum p_j$

⇒: Partition = perfekt balancierter Schedule.

⇐: Makespan $\frac{1}{2} \sum$ erzwingt exakte Balance = Partition.

ETH: $n' = n$, $T = O(\sum a_i)$.

KNAPSACK (Entscheidung)

Def: Gewichte w_i , Profite p_i ; Auswahl mit $\sum w_i \leq K$ und $\sum p_i \geq P$?

Bsp: Items (w, p) : $(2, 3), (3, 4)$; $K = 3$, $P = 4$: ja (zweites Item).

Reduktion: von SUBSETSUM.

Konstruktion:

– $w_i = p_i = a_i$

– $K = P = T$

⇒: Summe = $T \rightarrow$ Gewicht $\leq K$ und Profit $\geq P$.

⇐: Gewicht $\leq T$ und Profit $\geq T$ zugleich $\rightarrow$ Summe exakt T .

ETH: $n' = n$, $K = P = T$.

HITTINGSET

Def: Universum U , Mengen $F_1, \dots, F_r$, Zahl k ; H , $|H| \leq k$, trifft jede F_i ?

Bsp: $U = \{1..5\}$, $F = \{1, 2\}, \{2, 3\}, \{4, 5\}$; $H = \{2, 4\}$, $k=2$ ja.

Reduktion: von 3-SAT (HA 12.2 / Probeklausur).

ETH: $|U| = O(n)$, $r = O(m)$, $k = n$.

NAE-4-SAT

Def: Klauseln ≤ 4 Literale; Belegung, sodass jede Klausel wahres UND falsches Literal hat (“not all equal”)?

Bsp: $(x \vee y \vee z)$: $x=1, y=0$ – ja. (x) allein: nein (nie beides).

Reduktion: von 3-SAT (SS25-Klausur).

ETH: $n' = O(n)$, $m' = m$.

HALT_{TM} (nur NP-schwer!)

Def: Hält DTM M auf Wort w ?

Bsp: M = Endlosschleife auf jeder Eingabe: nein für jedes w .

Reduktion: von 3-SAT (HA 11.2).

Konstruktion:

– M_φ : probiere alle 2^n Belegungen; falls eine erfüllt, halte; sonst Endlosschleife

– $w = \langle \varphi \rangle$

⇒: Erfüllbar $\rightarrow M_\varphi$ findet Belegung und hält.

⇐: M_φ hält nur im Erfolgsfall $\rightarrow$ erfüllbar.

GÜTE: Δ TSP1

Algorithmus: MST $T \rightarrow$ Kanten verdoppeln $\rightarrow$ Eulerkreis $\rightarrow$ Abkürzen.

Güte: 2.

Beweis:

– Kante aus optimaler Tour entfernen $\rightarrow$ Spannbaum $\rightarrow w(T) \leq \text{OPT}$

- Verdoppeln: Eulerkreis der Länge $2w(T) \leq 2\text{OPT}$
- Abkürzen verlängert nicht (Δ -Ungleichung)

GÜTE: CHRISTOFIDES (ΔTSP2)

Algorithmus: MST $T \rightarrow X = \text{Knoten ungeraden Grades} \rightarrow \text{min. perfektes Matching } M \text{ auf } X \rightarrow \text{Eulerkreis in } T + M \rightarrow \text{Abkürzen.}$

Güte: $\frac{3}{2}$.

Beweis:

- $w(T) \leq \text{OPT}$ (Tour minus Kante = Spannbaum)
- $|X|$ gerade; optimale Tour auf X abkürzen $\rightarrow$ Kreis C mit $d(C) \leq \text{OPT}$
- C zerfällt in zwei perfekte Matchings $M_1, M_2 \rightarrow w(M) \leq \min \leq \frac{1}{2}\text{OPT}$
- Eulerkreis in $T + M$: $w(T) + w(M) \leq \frac{3}{2}\text{OPT}$; Abkürzen verlängert nicht

Scharf: Leitergraph (Präsenz 12.2): Zick-Zack-MST $(n-1) + \text{Matchingkante } \{1, n\} \text{ der Länge } \frac{n}{2} \rightarrow \text{Rate } \frac{n-1+n/2}{n} \rightarrow \frac{3}{2}$.

GÜTE: GREEDY-KNAPSACK (Widerlegung)

Algorithmus: Absteigend nach Dichte p_i/w_i ; packe jedes noch passende Item.

Güte: Unbeschränkt – keine konstante Güte.

Beweis:

- Instanz $(w, p) = (1, 1), (B, B-1)$, Kapazität B
- Dichten: $1 > \frac{B-1}{B} \rightarrow$ Greedy packt Item 1; Item 2 passt nicht mehr
- $GA = 1$, $\text{OPT} = B - 1 \rightarrow \text{Rate } B - 1 \rightarrow \infty$

GÜTE: MODIFIEDGREEDY (Knapsack)

Algorithmus: $\text{MGA} = \max\{\text{Greedy-Lösung, profitreichstes Einzelitem}\}$.

Güte: 2, d.h. $\text{OPT} \leq 2\text{MGA}$.

Beweis:

- $k+1$ = erstes Item, das Greedy nicht mehr packt
- fraktionale Relaxierung: $\text{OPT} \leq \text{OPT}_f \leq p_1 + \dots + p_k + p_{k+1}$
- $p_1 + \dots + p_k \leq GA$ und $p_{k+1} \leq p_{\max}$
- $\text{OPT} \leq GA + p_{\max} \leq 2 \max\{GA, p_{\max}\} = 2\text{MGA}$

GÜTE: LISTSCHEDULING ($P \parallel C_{\max}$)

Algorithmus: Jobs der Reihe nach auf die aktuell leerste Maschine.

Güte: $2 - \frac{1}{m}$.

Beweis:

- J_k = letzter Job auf der vollsten Maschine (Last L); bei Platzierung war sie die leerste
- alle Maschinen hatten Last $\geq L - p_k \rightarrow \sum_i p_i \geq m(L - p_k) + p_k$
- $\text{OPT} \geq \frac{1}{m} \sum_i p_i \geq L - (1 - \frac{1}{m})p_k$
- $p_k \leq \text{OPT}$ einsetzen $\rightarrow L \leq (2 - \frac{1}{m})\text{OPT}$

Scharf: (Skript!) $m(m-1)$ Einser + 1 Job der Größe m : $LS = 2m - 1$, $\text{OPT} = m$.

GÜTE: LPT

Algorithmus: Jobs absteigend sortieren, dann ListScheduling.

Güte: $\frac{4}{3} - \frac{1}{3m}$.

Beweis:

- wie LS: $\text{LPT} \leq \text{OPT} + (1 - \frac{1}{m})p_n$ (p_n = Job, der den Makespan setzt)
- Fall $p_n \leq \frac{1}{3}\text{OPT}$: einsetzen $\rightarrow (\frac{4}{3} - \frac{1}{3m})\text{OPT}$
- Fall $p_n > \frac{1}{3}\text{OPT}$: alle Jobs $> \frac{1}{3}\text{OPT} \rightarrow$ optimal höchstens 2 Jobs pro Maschine $\rightarrow$ LPT ist optimal

GÜTE: 2ApproxVC

Algorithmus: Kanten durchgehen; sind beide Endpunkte neu, nimm beide in C auf.

Güte: 2, d.h. $|C| \leq 2|C^*|$.

Beweis:

- A = aufgenommene Kanten $\rightarrow |C| = 2|A|$

- A ist Matching: gemeinsamer Endpunkt wäre schon in C gewesen
- C^* braucht pro Kante aus A einen *eigenen* Knoten $\rightarrow |C^*| \geq |A|$
- $|C| = 2|A| \leq 2|C^*|$

GÜTE: MAX-3-SAT

Algorithmus: Werte β_0 (alles falsch) und β_1 (alles wahr) aus; gib die bessere zurück.

Güte: 2, d.h. $v(A(\varphi)) \geq \frac{1}{2}v(\text{OPT})$.

Beweis:

- jede Klausel hat ≥ 1 Literal; negatives Literal $\rightarrow \beta_0$ erfüllt, positives $\rightarrow \beta_1$
- also $v(\beta_0) + v(\beta_1) \geq m$
- $\max \geq \text{Durchschnitt} \geq \frac{m}{2} \geq \frac{1}{2}v(\text{OPT})$

Scharf: $(x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)$: $A = 1$, $\text{OPT} = 2$ (via $x_1 = 1, x_2 = 0$).