# Übung 04

## Ressourcenplanung und Durchlaufterminierung

## Aufgabe 1: Montageplanung

Die Fahrzeugfertigung Zwickau GmbH ist ein traditionsreicher Standort im Automobilbau und hat sich auf die Produktion von Komponenten und die Endmontage von Elektrofahrzeugen spezialisiert. Für die Einführung einer neuen Montagelinie zur Fertigung von Batteriemodulen für das Modell "Saxon-E" müssen die einzelnen Arbeitsschritte genau geplant werden. Das Projektmanagement-Team hat die folgende Liste von Arbeitsgängen (AG), deren Dauer in Stunden (Std.) und die direkten Vorgänger identifiziert:

| Arbeitsgang | Beschreibung | Dauer (Std.) | Direkte Vorgänger |
|---|---|---|---|
| A | Materialbereitstellung Rahmen | 3 | - |
| B | Vormontage Zellhalterungen | 5 | A |
| C | Einbau Zellhalterungen in Rahmen | 4 | B |
| D | Materialbereitstellung Elektronik | 2 | - |
| E | Vormontage Steuerungseinheit | 6 | D |
| F | Integration Steuerungseinheit | 3 | C, E |
| G | Qualitätsprüfung & Endverschluss | 2 | F |

Der Projektstart (Beginn von A und D) ist zum Zeitpunkt 0. Der spätestzulässige Fertigstellungstermin für den gesamten Prozess (Ende von G) ist Stunde 25.

**Ihre Aufgaben:**

1. Erstellen Sie ein Netzplan-Diagramm für dieses Projekt.
2. Führen Sie eine Vorwärtsrechnung durch, um die frühestmöglichen Anfangszeitpunkte (FAZ) und Endzeitpunkte (FEZ) für jeden Arbeitsgang zu bestimmen.
3. Führen Sie eine Rückwärtsrechnung durch, um die spätestzulässigen Anfangszeitpunkte (SAZ) und Endzeitpunkte (SEZ) für jeden Arbeitsgang zu bestimmen. Gehen Sie davon aus, dass der SEZ des letzten Arbeitsgangs (G) dem spätestzulässigen Projektendtermin entspricht, falls er diesen nicht überschreitet. Andernfalls ist der spätestzulässige Endtermin des Projekts maßgebend für SEZ(G).
4. Berechnen Sie die Gesamtpufferzeit (GP) für jeden Arbeitsgang.
5. Identifizieren Sie den kritischen Weg im Projekt.

**Lösungshinweise:**

1. **Netzplan:** Eine grafische Darstellung der Arbeitsgänge und ihrer Abhängigkeiten. (Skizze auf Papier)

2. **Vorwärtsrechnung:**
   - A: FAZ=0, FEZ=3
   - D: FAZ=0, FEZ=2
   - B: FAZ=max(FEZ(A))=3, FEZ=3+5=8
   - E: FAZ=max(FEZ(D))=2, FEZ=2+6=8
   - C: FAZ=max(FEZ(B))=8, FEZ=8+4=12
   - F: FAZ=max(FEZ(C)=12, FEZ(E)=8)=12, FEZ=12+3=15
   - G: FAZ=max(FEZ(F))=15, FEZ=15+2=17 Die minimale Projektdauer beträgt 17 Stunden.
3. **Rückwärtsrechnung (Annahme SEZ(G) = FEZ(G) = 17, um kritischen Pfad zu finden):**
   - G: SEZ=17, SAZ=17-2=15
   - F: SEZ=min(SAZ(G))=15, SAZ=15-3=12
   - C: SEZ=min(SAZ(F))=12, SAZ=12-4=8
   - E: SEZ=min(SAZ(F))=12, SAZ=12-6=6
   - B: SEZ=min(SAZ(C))=8, SAZ=8-5=3
   - A: SEZ=min(SAZ(B))=3, SAZ=3-3=0
   - D: SEZ=min(SAZ(E))=6, SAZ=6-2=4
4. **Gesamtpuffer (GP = SAZ - FAZ):**
   - A: GP = 0 - 0 = 0
   - B: GP = 3 - 3 = 0
   - C: GP = 8 - 8 = 0
   - D: GP = 4 - 0 = 4
   - E: GP = 6 - 2 = 4
   - F: GP = 12 - 12 = 0
   - G: GP = 15 - 15 = 0
5. **Kritischer Weg (GP=0):** A ⟶ B ⟶ C ⟶ F ⟶ G. Die Dauer dieses Pfades ist 3+5+4+3+2 = 17 Stunden.

**Hinweis:** Wenn der gegebene spätestzulässige Projektendtermin (hier 25 Stunden) für die Rückwärtsrechnung verwendet worden wäre (d.h. SEZ(G) = 25), dann wären alle Pufferzeiten positiv gewesen (z.B. GP(G) = SAZ(G) - FAZ(G) = (25-2) - 15 = 23 - 15 = 8). In diesem Fall gäbe es keinen Arbeitsgang mit Pufferzeit Null. Der "kritische Weg" wird typischerweise als der Pfad mit Pufferzeit Null definiert, was impliziert, dass das Projekt so früh wie möglich abgeschlossen wird.

# Aufgabe 2: Kapazitätsabgleich

Die Duisburg Flugzeugwerke GmbH ist spezialisiert auf die Umrüstung von Passagierflugzeugen in Frachtflugzeuge (P2F - Passenger to Freighter). Bei der Umrüstung eines A320-Flugzeugs müssen mehrere neue Bodenstrukturelemente im Frachtraum installiert werden. Diese Arbeiten erfordern den Einsatz einer speziellen, hochpräzisen mobilen Nietanlage, von der im Hangar für dieses Projekt aktuell nur **eine** Einheit zur Verfügung steht (Kapazität = 1).

Das Projektmanagement-Team hat folgende Arbeitsgänge (AG), deren Dauer in Tagen, die direkten Vorgänger und den Bedarf an der mobilen Nietanlage (NA) identifiziert:

| AG | Beschreibung | Dauer (Tage) | Direkte Vorgänger | Benötigt (NA=1) | Nietanlage |
|---|---|---|---|---|---|
| A | Vorbereitung Sektion 1 | 2 | - | Nein | |
| B | Vorbereitung Sektion 2 | 1 | - | Nein | |
| C | Nietarbeiten Sektion 1 | 3 | A | Ja | |
| D | Nietarbeiten Sektion 2 | 4 | B | Ja | |
| E | Montage Hilfsstruktur | 2 | - | Nein | |
| F | Finale Niet-Verbindung | 3 | C, D, E | Ja | |
| G | Inspektion & Abschluss | 1 | F | Nein | |

Projektstart ist zum Zeitpunkt 0.

**Ihre Aufgaben:**

1. Durchlaufterminierung (ohne Kapazitätsbeschränkung):
   a. Bestimmen Sie die frühestmöglichen Anfangs- (FAZ) und Endzeitpunkte (FEZ) für alle Arbeitsgänge.
   b. Bestimmen Sie die spätestzulässigen Anfangs- (SAZ) und Endzeitpunkte (SEZ) für alle Arbeitsgänge. Nehmen Sie für die Rückwärtsrechnung an, dass SEZ(G) = FEZ(G) ist, um den kritischen Pfad zu identifizieren.
   c. Berechnen Sie die Gesamtpufferzeit (GP) und identifizieren Sie den/die kritischen Pfad(e).
   d. Wie lange dauert das Projekt minimal ohne Kapazitätsengpässe?
2. Kapazitätsorientierte Planung (mit Nietanlage Kapazität = 1):
   a. Identifizieren Sie die Arbeitsgänge, die die Nietanlage benötigen.
   b. Erstellen Sie einen neuen Zeitplan (Start- und Endtermine für alle Arbeitsgänge), der die Kapazitätsbeschränkung der Nietanlage (maximal ein Arbeitsgang gleichzeitig) berücksichtigt. Versuchen Sie, die Arbeitsgänge so auf der Nietanlage einzuplanen, dass die neue Gesamtprojektdauer minimiert wird. Verwenden Sie die FAZ-Werte aus der unbeschränkten Planung als eine mögliche Priorität (frühester Start zuerst). Bei Gleichheit kann die Dauer oder eine andere logische Überlegung entscheiden.
   c. Stellen Sie den Belegungsplan der Nietanlage und den resultierenden Projektplan skizzenhaft dar (z.B. als einfache Zeitachse oder Tabelle).
   d. Wie lange dauert das Projekt nun unter Berücksichtigung des Engpasses? Was ist der neue kritische Pfad?

**Lösungshinweise:**

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.patches as mpatches

# Definition der Arbeitsgänge für Aufgabe 2
tasks_data_t2 = [
```

```
    {'AG': 'A', 'Beschreibung': 'Vorbereitung Sektion 1', 'Dauer': 2,
'Vorgänger': [], 'Bedarf_NA': 0},
    {'AG': 'B', 'Beschreibung': 'Vorbereitung Sektion 2', 'Dauer': 1,
'Vorgänger': [], 'Bedarf_NA': 0},
    {'AG': 'C', 'Beschreibung': 'Nietarbeiten Sektion 1', 'Dauer': 3,
'Vorgänger': ['A'], 'Bedarf_NA': 1},
    {'AG': 'D', 'Beschreibung': 'Nietarbeiten Sektion 2', 'Dauer': 4,
'Vorgänger': ['B'], 'Bedarf_NA': 1},
    {'AG': 'E', 'Beschreibung': 'Montage Hilfsstruktur', 'Dauer': 2,
'Vorgänger': [], 'Bedarf_NA': 0},
    {'AG': 'F', 'Beschreibung': 'Finale Niet-Verbindung', 'Dauer': 3,
'Vorgänger': ['C', 'D', 'E'], 'Bedarf_NA': 1},
    {'AG': 'G', 'Beschreibung': 'Inspektion & Abschluss', 'Dauer': 1,
'Vorgänger': ['F'], 'Bedarf_NA': 0}
]

def calculate_cpm(task_list, fixed_starts=None):
    tasks_dict_cpm = {task['AG']: task.copy() for task in task_list}
    if fixed_starts is None:
        fixed_starts = {}

    for task_id in tasks_dict_cpm:
        tasks_dict_cpm[task_id]['FAZ'] = 0
        tasks_dict_cpm[task_id]['FEZ'] = 0
        tasks_dict_cpm[task_id]['SAZ'] = 0
        tasks_dict_cpm[task_id]['SEZ'] = 0
        tasks_dict_cpm[task_id]['GP'] = 0
        tasks_dict_cpm[task_id]['Kritisch'] = False

    # Angepasste Reihenfolge für dieses Beispiel (oder topologische
Sortierung verwenden)
    processing_order_fwd_cpm = ['A', 'B', 'E', 'C', 'D', 'F', 'G']
    # Sicherstellen, dass alle Tasks in der Liste sind
    current_tasks_ids = [t['AG'] for t in task_list]
    processing_order_fwd_cpm = [t for t in processing_order_fwd_cpm if t in
current_tasks_ids] \
                               + [t for t in current_tasks_ids if t not in
processing_order_fwd_cpm]


    # Vorwärtsrechnung
    for task_id in processing_order_fwd_cpm:
        task = tasks_dict_cpm[task_id]
        if task_id in fixed_starts:
            task['FAZ'] = fixed_starts[task_id]['start']
        elif not task['Vorgänger']:
            task['FAZ'] = 0
        else:
            task['FAZ'] = max(tasks_dict_cpm[p_id]['FEZ'] for p_id in
task['Vorgänger'] if p_id in tasks_dict_cpm)

        if task_id in fixed_starts:
```

```
            task['FEZ'] = fixed_starts[task_id]['end']
        else:
            task['FEZ'] = task['FAZ'] + task['Dauer']

    last_task_id_cpm = processing_order_fwd_cpm[-1]
    tasks_dict_cpm[last_task_id_cpm]['SEZ'] =
tasks_dict_cpm[last_task_id_cpm]['FEZ']
    tasks_dict_cpm[last_task_id_cpm]['SAZ'] =
tasks_dict_cpm[last_task_id_cpm]['SEZ'] - tasks_dict_cpm[last_task_id_cpm]
['Dauer']

    # Rückwärtsrechnung
    processing_order_bwd_cpm = processing_order_fwd_cpm[::-1]
    for task_id in processing_order_bwd_cpm:
        task = tasks_dict_cpm[task_id]
        if task_id == last_task_id_cpm:
            continue

        successors_of_current_task = []
        for succ_id, succ_details in tasks_dict_cpm.items():
            if task_id in succ_details['Vorgänger']:
                successors_of_current_task.append(succ_id)

        if not successors_of_current_task:
            task['SEZ'] = tasks_dict_cpm[last_task_id_cpm]['FEZ']
        else:
            task['SEZ'] = min(tasks_dict_cpm[s_id]['SAZ'] for s_id in
successors_of_current_task)

        task['SAZ'] = task['SEZ'] - task['Dauer']

        # Wenn Startzeit fixiert war, kann SAZ nicht früher sein als FAZ
        if task_id in fixed_starts:
            task['SAZ'] = max(task['SAZ'], task['FAZ'])
            task['SEZ'] = task['SAZ'] + task['Dauer']


    for task_id in tasks_dict_cpm:
        task = tasks_dict_cpm[task_id]
        task['GP'] = task['SAZ'] - task['FAZ']
        task['Kritisch'] = (task['GP'] == 0)

    result_df_intermediate = pd.DataFrame.from_dict(tasks_dict_cpm,
orient='index')
    result_df_intermediate = result_df_intermediate.reset_index()
    df_to_return = result_df_intermediate.drop(columns=['AG'])
    df_to_return = df_to_return.rename(columns={'index': 'AG'})
    return df_to_return

# Neue Funktion zum Erstellen von Gantt-Diagrammen
def plot_gantt_schedule(df, title, ax, task_col='AG', start_col='FAZ',
end_col='FEZ', critical_col='Kritisch', duration_col='Dauer'):
```

```
    """Erstellt ein Gantt-Diagramm für den Projektplan."""

    # Sortiere Tasks nach AG für eine konsistente Y-Achsen-Reihenfolge oder
nach FAZ
    # Hier verwenden wir die Reihenfolge im DataFrame, die typischerweise
nach AG oder Verarbeitungsreihenfolge ist
    # oder sortieren explizit, z.B. nach FAZ für eine chronologische
Darstellung von oben nach unten.
    df_plot = df.sort_values(by=start_col).copy()

    y_labels = df_plot[task_col].tolist()
    y_pos = np.arange(len(y_labels))

    for i, task_id in enumerate(y_labels):
        task_row = df_plot[df_plot[task_col] == task_id].iloc[0]
        start_time = task_row[start_col]
        # Dauer kann direkt aus der Spalte 'Dauer' genommen werden oder als
FEZ - FAZ berechnet werden
        # Stellen wir sicher, dass wir die tatsächliche Dauer des Tasks im
Plan verwenden.
        # Für geplante Tasks (fixed_starts) ist FEZ-FAZ die korrekte Dauer
im Plan.
        duration_in_plan = task_row[end_col] - task_row[start_col]
        is_critical = task_row[critical_col]

        color = 'salmon' if is_critical else 'skyblue' # Farben für
kritisch/nicht-kritisch

        ax.barh(y_pos[i], duration_in_plan, left=start_time,
edgecolor='black', color=color, height=0.6, align='center')

        # Text im Balken: Task ID und Dauer
        ax.text(start_time + duration_in_plan / 2, y_pos[i], f"{task_id}
({task_row[duration_col]})",
                va='center', ha='center', color='black', fontsize=9,
fontweight='normal')

    ax.set_yticks(y_pos)
    ax.set_yticklabels(y_labels)
    ax.invert_yaxis() # Tasks von oben nach unten in der Reihenfolge der
y_labels (sortiert nach FAZ)

    ax.set_xlabel("Zeit (Tage)")
    ax.set_title(title, fontsize=10)
    ax.grid(True, axis='x', linestyle=':', linewidth=0.7)

    # X-Achsen-Ticks für bessere Lesbarkeit
    max_time = df_plot[end_col].max()
    ax.set_xticks(np.arange(0, max_time + 2, 1)) # Ticks jeden Tag
    ax.set_xlim(0, max_time + 1)

    # Legende für kritische Pfade
```

```
    critical_patch = mpatches.Patch(color='salmon', label='Kritischer
Arbeitsgang')
    non_critical_patch = mpatches.Patch(color='skyblue', label='Nicht-
kritischer Arbeitsgang')
    ax.legend(handles=[critical_patch, non_critical_patch], loc='lower
right', fontsize=8)


# 1. Durchlaufterminierung (ohne Kapazitätsbeschränkung)
print("1. Durchlaufterminierung (ohne Kapazitätsbeschränkung):")
df_unconstrained = calculate_cpm(tasks_data_t2)
display_columns_cpm = ['AG', 'Dauer', 'FAZ', 'FEZ', 'SAZ', 'SEZ', 'GP',
'Kritisch']
print(df_unconstrained[display_columns_cpm].to_string(index=False))
duration_unconstrained = df_unconstrained.loc[df_unconstrained['AG'] ==
'G', 'FEZ'].iloc[0]
krit_weg_unconstrained_df =
df_unconstrained[df_unconstrained['Kritisch']].sort_values(by='FAZ')
print(f"\nMinimale Projektdauer (unbeschränkt): {duration_unconstrained}
Tage")
print(f"Kritischer Weg (unbeschränkt): {' ->
'.join(krit_weg_unconstrained_df['AG'].tolist())}")

# Erstellung der Gantt-Diagramme am Ende der Berechnungen
fig_gantt, ax_gantt_unconstrained = plt.subplots(figsize=(7, 5))
fig_gantt.set_facecolor('white') # Hintergrund der Figur

plot_gantt_schedule(df_unconstrained,
                    'Gantt-Diagramm: Ohne Kapazitätsbeschränkung',
                    ax_gantt_unconstrained,
                    task_col='AG', start_col='FAZ', end_col='FEZ',
critical_col='Kritisch', duration_col='Dauer')
```

```
1. Durchlaufterminierung (ohne Kapazitätsbeschränkung):
AG  Dauer  FAZ  FEZ  SAZ  SEZ  GP  Kritisch
 A      2    0    2    0    2   0      True
 B      1    0    1    0    1   0      True
 C      3    2    5    2    5   0      True
 D      4    1    5    1    5   0      True
 E      2    0    2    3    5   3     False
 F      3    5    8    5    8   0      True
 G      1    8    9    8    9   0      True

Minimale Projektdauer (unbeschränkt): 9 Tage
Kritischer Weg (unbeschränkt): A -> B -> D -> C -> F -> G
```

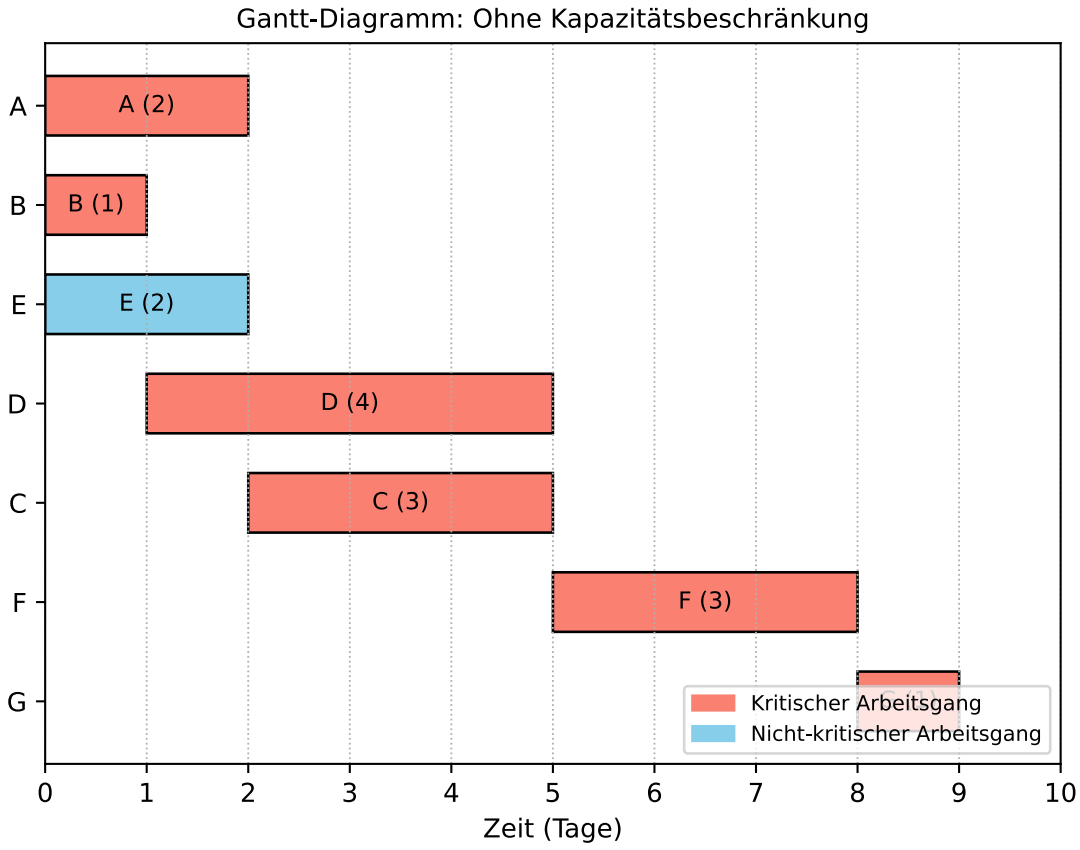


Figure 1: Netzplanberechnung und Kapazitätsabgleich für Flugzeugumbau

**Erläuterung der Ergebnisse:**

1. **Durchlaufterminierung (ohne Kapazitätsbeschränkung):**
   - Die Studierenden erstellen eine Tabelle mit FAZ, FEZ, SAZ, SEZ, GP für alle Arbeitsgänge.
   - Beispielhafte unbeschränkte Werte (manuell zu berechnen):
     - A: FAZ=0, FEZ=2
     - B: FAZ=0, FEZ=1
     - E: FAZ=0, FEZ=2
     - C: FAZ=max(FEZ(A))=2, FEZ=2+3=5
     - D: FAZ=max(FEZ(B))=1, FEZ=1+4=5
     - F: FAZ=max(FEZ(C)=5, FEZ(D)=5, FEZ(E)=2)=5, FEZ=5+3=8
     - G: FAZ=max(FEZ(F))=8, FEZ=8+1=9
   - Minimale Projektdauer (unbeschränkt): 9 Tage.
   - Kritischer Pfad (unbeschränkt, nach manueller Berechnung, wenn SEZ(G)=FEZ(G)=9):
     - SEZ(G)=9, SAZ(G)=8
     - SEZ(F)=SAZ(G)=8, SAZ(F)=5
     - SEZ(C)=SAZ(F)=5, SAZ(C)=2
     - SEZ(D)=SAZ(F)=5, SAZ(D)=1
     - SEZ(E)=SAZ(F)=5, SAZ(E)=3
     - SEZ(A)=SAZ(C)=2, SAZ(A)=0
     - SEZ(B)=SAZ(D)=1, SAZ(B)=0
     - GP(A)=0, GP(B)=0, GP(C)=0, GP(D)=0, GP(E)=3, GP(F)=0, GP(G)=0.

- Kritische Pfade: A-C-F-G und B-D-F-G.

2. **Kapazitätsorientierte Planung (Nietanlage Kapazität = 1):**
   - Arbeitsgänge mit NA-Bedarf: C, D, F.
   - Unbeschränkte FAZ-Werte: FAZ(C)=2, FAZ(D)=1, FAZ(F)=5.
   - Priorisierung für NA (z.B. nach FAZ): D (startet früher oder gleichzeitig mit C), dann C, dann F.
     - **AG D (Dauer 4):** Kann frühestens an Tag 1 starten (nach B). FAZ(B)=1. NA ist frei.
       - Start D: Tag 1, Ende D: Tag 1+4=5. NA belegt bis Ende Tag 5.
     - **AG C (Dauer 3):** Kann frühestens an Tag 2 starten (nach A). FAZ(A)=2. NA ist aber erst ab Tag 5 frei (nach D).
       - Start C: max(FAZ(A)=2, NA_frei=5) = Tag 5. Ende C: Tag 5+3=8. NA belegt bis Ende Tag 8.
     - **AG F (Dauer 3):** Benötigt C, D, E.
       - FEZ(C) ist jetzt Tag 8 (verschoben).
       - FEZ(D) ist Tag 5.
       - FEZ(E) ist Tag 2 (unverändert).
       - Frühestmöglicher Start für F (Vorgänger): max(8, 5, 2) = Tag 8.
       - NA ist frei ab Ende Tag 8 (nach C).
       - Start F: max(Vorbedingung_Start=8, NA_frei=8) = Tag 8. Ende F: Tag 8+3=11.
   - Neuer Projektplan:
     - A: FAZ=0, FEZ=2
     - B: FAZ=0, FEZ=1
     - E: FAZ=0, FEZ=2
     - D: FAZ=1, FEZ=5 (durch NA geplant)
     - C: FAZ=5, FEZ=8 (durch NA geplant, startet nach D auf NA und nach A)
     - F: FAZ=max(FEZ(C)=8, FEZ(D)=5, FEZ(E)=2)=8, FEZ=8+3=11 (durch NA geplant)
     - G: FAZ=max(FEZ(F))=11, FEZ=11+1=12
   - Neue minimale Projektdauer (beschränkt): 12 Tage.
   - Der kritische Pfad im beschränkten Plan ergibt sich aus der Abarbeitung der Nietanlagen-Tasks und deren Abhängigkeiten. Er ist **B → D → C → F → G**. Die Projektdauer beträgt somit 12 Tage. Die Arbeitsgänge D, C, und F werden nacheinander auf der Nietanlage ausgeführt und sind kritisch. B ist kritisch als direkter Vorgänger von D. G ist kritisch als direkter Nachfolger von F.
   - Task A (Dauer 2 Tage, Vorgänger von C) ist im beschränkten Plan **nicht kritisch**:
     - FAZ(A) = 0, FEZ(A) = 2.
     - Da C (Nachfolger von A) im beschränkten Plan an Tag 5 beginnt (FAZ(C)=5, SAZ(C)=5), muss A spätestens an Tag 5 beendet sein (SEZ(A)=5).
     - Der spätestzulässige Startzeitpunkt für A ist SAZ(A) = SEZ(A) - Dauer(A) = 5 - 2 = 3.
     - Der Gesamtpuffer für Task A ist GP(A) = SAZ(A) - FAZ(A) = 3 - 0 = 3 Tage.