Book Scanning Problem: Mathematical Formulation and Python Implementation

Sets and Parameters

- **Mathematical Formulation:**

$$
\begin{aligned}
&\text{Let } B \text{ be the set of books and } L \text{ the set of libraries.}\\
&\text{Let } D \text{ be the total number of days available for scanning.}\\
&\text{For each book } b \in B, \text{ let } s_b \text{ be the score awarded when book } b \text{ is scanned.}\\
&\text{For each library } l \in L:\\
&\quad B_l \subseteq B \text{ is the set of books available in library } l,\\
&\quad \sigma_l \text{ is the number of days to complete the signup process for library } l,\\
&\quad \delta_l \text{ is the number of books library } l \text{ can scan per day after signup.}
\end{aligned}
$$

- **Code Implementation:**

```
# B: set of books (assumed as indices or IDs)
# L: set of libraries (assumed as indices or IDs)
# D: total days (integer)
# book_scores: dictionary mapping b to s_b
# libraries: dictionary where libraries[l] has 'books' (B_l), 'signup' (
    sigma_l), 'ship' (delta_l)
```

Decision Variables

- **Mathematical Formulation:**

$$
\begin{aligned}
&y_l \in \{0,1\}, \quad \forall l \in L \quad \text{(1 if library } l \text{ is signed up, 0 otherwise)}\\
&z_{l,b} \in \{0,1\}, \quad \forall l \in L, \forall b \in B_l \quad \text{(1 if book } b \text{ is scanned from library } l\text{)}\\
&u_b \in \{0,1\}, \quad \forall b \in B \quad \text{(1 if book } b \text{ is scanned from any library, 0 otherwise)}\\
&p_{l1,l2} \in \{0,1\}, \quad \forall l1, l2 \in L,\, l1 \neq l2 \quad \text{(1 if library } l1 \text{ is processed before } l2\text{)}\\
&t_l \in \{0,1,\ldots,D-1\}, \quad \forall l \in L \quad \text{(day when signup for library } l \text{ starts)}
\end{aligned}
$$

- **Code Implementation:**

```
from ortools.linear_solver import pywraplp
solver = pywraplp.Solver.CreateSolver('SCIP')

y = {l: solver.IntVar(0, 1, f'y[{l}]') for l in L}
z = {(l, b): solver.Relevant_z(0, 1, f'z[{l},{b}]')
     for l in L for b in libraries[l]['books'] if b in book_scores}
u = {b: solver.IntVar(0, 1, f'u[{b}]') for b in B}
p = {(l1, l2): solver.IntVar(0, 1, f'p[{l1},{l2}]')
     for l1 in L for l2 in L if l1 != l2}
t = {l: solver.IntVar(0, D - 1, f't[{l}]') for l in L}
```

Objective Function

- **Mathematical Formulation:**

$$\text{Maximize} \sum_{b \in B} s_b u_b$$

- **Meaning:** The goal is to maximize the total score of scanned books. A book contributes its score s_b only if it is scanned ($u_b = 1$).
- **Code Implementation:**

```
objective = solver.Objective()
for b in B:
    if b in book_scores:
        objective.SetCoefficient(u[b], book_scores[b])
objective.SetMaximization()
```

Constraints

1. Each Book Scanned at Most Once

- **Mathematical Formulation:**

$$\sum_{l \in L} z_{l,b} \leq 1 \quad \forall b \in B$$

- **Meaning:** Each book can be scanned by at most one library, ensuring no duplicate scanning.
- **Code Implementation:**

```
for b in B:
    relevant_z = [z[(l, b)] for l in L
                  if b in libraries[l]['books'] and (l, b) in z]
    if relevant_z:
        solver.Add(solver.Sum(relevant_z) <= 1)
```

2. Link u_b to $z_{l,b}$

- **Mathematical Formulation:**

$$u_b \geq z_{l,b} \quad \forall l \in L, \forall b \in B_l$$

- **Meaning:** If a book is scanned from any library (i.e., if any $z_{l,b} = 1$), then u_b must be set to 1, indicating the book is scanned.
- **Code Implementation:**

```
for l in L:
    for b in libraries[l]['books']:
        if (l, b) in z:
            solver.Add(u[b] >= z[(l, b)])
```

3. Only Scan from Signed-Up Libraries

- **Mathematical Formulation:**

$$z_{l,b} \leq y_l \quad \forall l \in L, \forall b \in B_l$$

- **Meaning:** A library can only scan its books if it has been signed up (i.e., $y_l = 1$).

- **Code Implementation:**

```
for l in L:
    for b in libraries[l]['books']:
        if (l, b) in z:
            solver.Add(z[(l, b)] <= y[l])
```

4. **Library Order Can't Conflict**

- **Mathematical Formulation:**

$$p_{l1,l2} + p_{l2,l1} \leq 1 \quad \forall l1, l2 \in L,\ l1 \neq l2$$

- **Meaning:** This constraint ensures that for any two libraries, both cannot be scheduled to come before each other.
- **Code Implementation:**

```
for l1 in L:
    for l2 in L:
        if l1 < l2:
            solver.Add(p[(l1, l2)] + p[(l2, l1)] <= 1)
```

5. **Timing Between Libraries**

- **Mathematical Formulation:**

$$t_{l2} \geq t_{l1} + \sigma_{l1} \cdot y_{l1} - D \cdot (1 - p_{l1,l2}) \quad \forall l1, l2 \in L,\ l1 \neq l2$$

- **Meaning:** If library $l1$ is scheduled before library $l2$ (i.e., $p_{l1,l2} = 1$), then the signup for $l2$ must start after $l1$'s signup is completed.
- **Code Implementation:**

```
for l1 in L:
    for l2 in L:
        if l1 != l2:
            solver.Add(t[l2] >= t[l1] + libraries[l1]['signup'] * y[l1]
                       - D * (1 - p[(l1, l2)]))
```

6. **Order If Both Signed Up**

- **Mathematical Formulation:**

$$p_{l1,l2} + p_{l2,l1} \geq y_{l1} + y_{l2} - 1 \quad \forall l1, l2 \in L,\ l1 \neq l2$$

- **Meaning:** If both libraries are signed up (i.e., $y_{l1} = 1$ and $y_{l2} = 1$), then one must come before the other.
- **Code Implementation:**

```
for l1 in L:
    for l2 in L:
        if l1 < l2:
            solver.Add(p[(l1, l2)] + p[(l2, l1)] >= y[l1] + y[l2] - 1)
```

7. **Signup Finishes in Time**

- **Mathematical Formulation:**

$$t_l + \sigma_l \cdot y_l \leq D \quad \forall l \in L$$

- **Meaning:** The signup process for each library must be completed within the total available time D.
- **Code Implementation:**

```
for l in L:
    solver.Add(t[l] + libraries[l]['signup'] * y[l] <= D)
```

8. **Scanning Time Limit**

- **Mathematical Formulation:**

$$\sum_{b \in B_l} z_{l,b} \leq \delta_l \cdot (D - t_l - \sigma_l) \cdot y_l \quad \forall l \in L$$

- **Meaning:** The total number of books scanned by a library cannot exceed its scanning capacity (i.e., the number of days remaining after signup multiplied by the per-day scanning limit) if it is signed up.
- **Code Implementation:**

```
for l in L:
    sum_z = solver.Sum(z[(l, b)] for b in libraries[l]['books'] if (l, b)
        in z)
    capacity = libraries[l]['ship'] * (D - t[l] - libraries[l]['signup'])
    M = len(libraries[l]['books'])
    solver.Add(sum_z <= capacity + M * (1 - y[l]))
```

9. **Don't Scan More Than Available**

- **Mathematical Formulation:**

$$\sum_{b \in B_l} z_{l,b} \leq |B_l| \cdot y_l \quad \forall l \in L$$

- **Meaning:** A library can scan at most as many books as it has available.
- **Code Implementation:**

```
for l in L:
    sum_z = solver.Sum(z[(l, b)] for b in libraries[l]['books'] if (l, b)
        in z)
    solver.Add(sum_z <= len(libraries[l]['books']) * y[l])
```

10. **u_b Needs a Scan**

- **Mathematical Formulation:**

$$u_b \leq \sum_{l \in L} z_{l,b} \quad \forall b \in B$$

- **Meaning:** A book is marked as scanned (i.e., $u_b = 1$) only if it is scanned by at least one library.
- **Code Implementation:**

```
for b in B:
    relevant_z = [z[(l, b)] for l in L
                  if b in libraries[l]['books'] and (l, b) in z]
    if relevant_z:
        solver.Add(u[b] <= solver.Sum(relevant_z))
```

Conclusion

This document presents the Book Scanning Problem as a Mixed-Integer Linear Programming (MILP) model, with its mathematical formulation and corresponding Python implementation using the OR-Tools solver. The objective is to maximize the total score of scanned books while adhering to constraints on library signups, scanning capacities, and time limits.