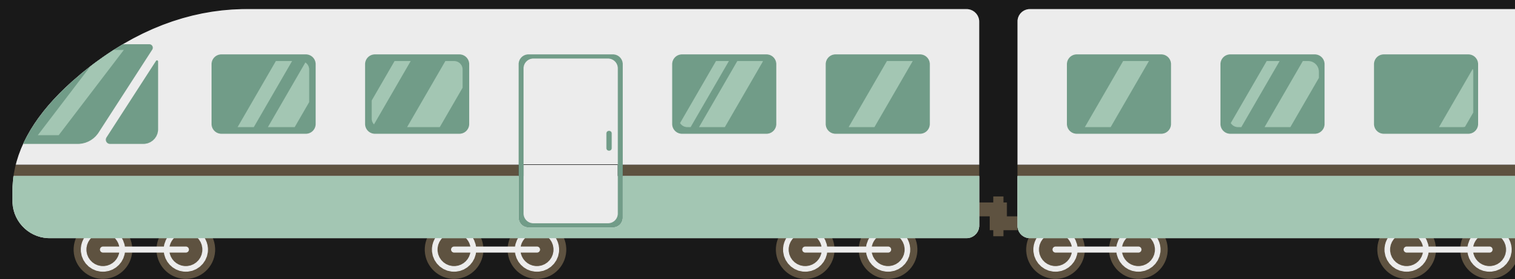
Optimizing Railway Traffic: An Operations Research Approach



Introduction

- Railway traffic management is a complex and challenging task that requires careful planning and execution. Operations research techniques can be used to optimize railway operations, improve efficiency, and reduce costs.
- Operations research involves the use of mathematical models, algorithms, and optimization techniques to solve complex problems in various fields, including transportation.

Challenges in Railway Traffic Management

Railway traffic management faces many challenges, including congestion, delays, and safety concerns. These challenges can lead to increased costs, reduced efficiency, and decreased customer satisfaction.

Operations research techniques can help address these challenges by optimizing train schedules, improving routing and signaling systems, and reducing downtime and maintenance costs.

Optimizing Train Schedules

One of the key applications of operations research in railway traffic management is optimizing train schedules. This involves determining the optimal sequence and timing of train movements to minimize delays and improve efficiency.

Mathematical models can be used to analyze train schedules, identify bottlenecks and constraints, and develop optimized schedules that balance competing priorities such as speed, reliability, and safety.

Problem Statement

The problem we are dealing with is about managing the flow of trains. In order to avoid congestion in critical rail zones, the departure of trains is delayed.
Limitations of rail capacity are expressed in terms of regulated periods, i.e. an interval with an hourly capacity rate. Regulated rail traffic zones have one or several regulated periods, i.e. their capacity is limited during these periods. The trains have an expected departure time (ETDT), which is specified in hours, minutes, and seconds and then converted into the total number of minutes. An enter event specifies that a given train will enter a given zone at an expected time (called expected time over).

OBJECTIVE :

"To minimize the total sum of train delays."

CPlex Code

```
using CP;

int No_of_Trains = ...;
range Trains = 1 .. No_of_Trains;

{string} ZoneNames = ...;

// times are specified in hours, minutes, seconds

tuple Time {
  int hours;
 int minutes;
 int seconds;
};
```

CPlex Code

```
// limitations of trains capacity are expressed in terms of regulated
// periods, i.e. an interval with an hourly capacity rate

tuple Period
{
Time start;
Time end;
int rate;
};

{Period} periods[ZoneNames] = ...;

// an enter event specifies that a given train will enter a given Zone
// at an expected time
tuple Enter {
int train;
string Zone;
Time eto;
};
```

CPlex Code

```
int No_of_Enters = ...;
range Enters = 1 .. No_of_Enters;
Enter Data[Enters] = ...;

// train delays limited to 2 hours
int maxDelay = 120;

// capacity of the resource will be made available by time steps of 10 minutes
int timeStep = 10;

// train delays are expressed by integer variables
dvar int delay[Trains] in 0 .. maxDelay;

// each enter event is modelled by an activity of duration 1
dvar interval activity[Enters] size 1;

// each Zone is modelled by a resource
cumulFunction resource[i in ZoneNames] = sum(en in Enters : Data[en].Zone == i)
pulse(activity[en], 1);
dexpr int totalDelay = sum(i in Trains) delay[i];
```

CPlex Code

```
minimize totalDelay;
constraints {
// the capacity rate is adapted to intervals of 10 minutes;
// the time scale of a resource is divided by the time step
forall (i in ZoneNames)
 forall (p in periods[i])
alwaysIn(resource[i],(p.start.hours * 60 + p.start.minutes) div timeStep,(p.end.hours * 60 +
p.end.minutes) div timeStep,0,(p.rate * timeStep + 59) div 60);

// a train enters a Zone at its expected time-over plus its delay;
// since the time scale of a resource is divided by the time step,
// we do the same for the start time of the activity
forall (i in Enters)
startOf(activity[i]) == (delay[Data[i].train] + Data[i].eto.hours * 60 + Data[i].eto.minutes) div timeStep;
//This constraint ensures that trains enter each zone at their expected time of arrival plus their delay.
forall(i in ZoneNames)
resource[i] <= No_of_Trains;
}
execute {
writeln("total number of delayed trains = " + totalDelay);
}
```

CPlex Code

Dat file

```
ZoneNames = {"Central_railway", "Konkan_railway", "Northern_railway",
"NorthCentral_railway", "NE_railway", "NEFrontier_railway", "NW_railway",
"Eastern_railway", "EastCentral_railway", "EastCoast_railway",
"Southern_railway", "Southcentral_railway", "Southeastern_railway",
"Southcoast_railway", "SoutheastCentral_railway", "Western_railway",
"WestCentral_railway"};

//taken from various railway zones in india

No_of_Trains = 200;
No_of_Enters = 324;

Data = [ ] // large dataset used
```

CPlex output

solution with objective 9

total number of delayed minutes = 9

delay = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 2 2 0 1 0 0 0 0 0 0 0 0 0 0];

the output gives the delay of each train in minutes and total delay is the sum of all the delays i.e = 9. minutes for our input

The Team

VIVEK KUMAR : 21IM10040

BISWARANJAN SAMAL : 21IM10011

UJJWAL KUMAR SINGH : 21IM10039

SNEHAL MISHRA : 21IM10036

MAGARE DRONESH MAHENDRA : 21IM30013