The problem in general:

- The problem of The Bus Driver Scheduling Problem (BDSP) is one of scheduling problem .The problem is to assign an employee (driver) to each bus in an effective way according to a set of restrictions. These restrictions are that the work company wants to reduce the value of the money paid, satisfy the employee, and not violate the schedules set in advance to meet the delivery date. Also, one of the restrictions is that every driver has A certain number of working hours per day and a certain number of continuous working hours throughout the day. Therefore, in the scheduling process, the driver's tasks are divided into pieces of work to work with throughout the full working day, where drivers are replaced at rest stations throughout their work period and But according to the events on the ground, this leads to non-compliance with the schedules and plans set

What is the desired goal ?

- constructing a set of legal shifts, which together cover all the trips planned for a group of vehicles . aiming at minimizing the number of vehicles
- Crew scheduling is subject to constraints that are specific to each company.(government legislation , union agreements and some operational rules that are settled between drivers and the transport company).

pieces-of-work

The daily work of each vehicle is divided into units, that start and finish at relief points

What's the solution ?

First, feasible duty is A set of pieces-of-work that satisfies all the constraints

The solution is a set of feasible duties that hopefully cover all the vehicle trips scheduled for a route or a small set of routes

problem is NP-hard

Generation of feasible duties

- Generating a large number of
  feasible duties, and by applying then a mathematical
  programming technique that selects a set of duties that cover
  all the trips with
  minimal cost .

- some systems use heuristic procedures that
  select potentially interesting duties and eliminate less efficient
  ones

Formulation of the bus driver scheduling problem

Constraints (1) impose that each piece-of-work is part of at least one duty of the solution.
Constraints (2) force the satisfaction of some operational properties of the solution (eg percentage of types of duties, number of duties or maximum working time)

The set partitioning model is a special case in which constraints (1) are equality constraints there is only one driver in each vehicle at any time. The main difficulty of this particular model is that, if a limited number of generated duties is available, we cannot usually guarantee that there is a feasible solution.

Therefore it is common to start by applying the set covering approach, even if the number of overcovers in the final solution may often be very high

$$\text{minimise} \quad \sum_{j \in P} c_j x_j$$

$$\text{subject to} \quad \sum_{j \in P} a_{ij} x_j \geqslant 1 \qquad \forall i \in I \qquad (1)$$

$$\sum_{j \in P} b_{ij} x_j \geqslant u_j \qquad \forall i \in I \qquad (2)$$

$$x_j \in \{0, 1\} \qquad \forall j \in P \qquad (3)$$

where:

$I = \{i$: piece-of-work$\}$;
$P = \{j$: candidate feasible duty$\}$;
c_j = cost of feasible duty j;
$x_j = 1$, if duty j is in the solution,
$= 0$, otherwise;
$a_{ij} = 1$, if the piece-of-work i belongs to duty j,
$= 0$, otherwise;
b_{ij}, u_j = real values representing operational properties of the solution.

A relaxed set partitioning model

- Overcovers →→ duty idle time
  when a piece-of-work is covered by several duties, we have a new decision process in which we have to decide which duty is going to be effective for that piece-of-work and which duties are going to be idle
  For this reason the transport companies we have been working with have serious difficulties in the implementation of covering solutions

$$\begin{aligned}
\text{minimise} \quad & \sum_{j \in P} c_j x_j + \sum_{i \in I} z_i y_i & \\
\text{subject to} \quad & \sum_{j \in P} a_{ij} x_j + y_i = 1 & \forall i \in I \\
& x_j \in \{0, 1\} & \forall j \in P \\
& y_i \in \{0, 1\} & \forall i \in I
\end{aligned}$$

where:
$I = \{i$: piece-of-work$\}$;
$P = \{j$: candidate feasible duty$\}$;
$c_j =$ cost of feasible duty j;
$z_i =$ penalty associated with not covering piece-of-work
$x_j = 1$, if duty j is in the solution,
$= 0$, otherwise;
$y_i = 1$, if piece-of-work i is not covered,
$= 0$, otherwise;
$a_{ij} = 1$, if piece-of-work i belongs to duty j,
$= 0$, otherwise.

Genetic algorithm

- A Genetic Algorithm (GA) may be described as a mechanism that imitates the genetic evolution of species

- One of the main differences between GAs and other local search heuristics (eg tabu search, simulated annealing) is that GAs search is based in a population of solutions not in a single solution

- A traditional GA usually uses a binary coding alphabet and the crossover and mutation operators do not include any knowledge about the structure and domain of the problem

- a hybrid GA we incorporate problem specific knowledge in the operators or in the coding scheme

the bus driver scheduling problem is basically composed of two types of information
the set of candidate duties and the set of pieces-of-work

the bus driver scheduling problem is basically composed of two types of information, the set of candidate duties and the set of pieces-of-work. In general there are much more duties than pieces of-work, but each solution is composed of a relatively small number of duties For example if for a given problem we have 15 candidate duties and a solution is composed by duties 1, 5, 8, 11 and 12, the corresponding chromosome will have length 15 and the following structure: 100010010011000.associating the two fundamental types of information contained in a solution: the duties and the set of pieces-of-work. Each piece-of-work corresponds to a gene of the chromosome, and each gene is characterised by the duty that covers the piece-of-work in that particular solution. The genes are ordered by bus and for each bus they are ordered by time One advantage of this coding scheme comes from the fact that a binary representation is used, which makes the data structures extremely simple, economical and easy to manipulate by the standard operators The cost of a duty is computed as a function of the duration, size of the breaks, extra work but The way the penalty function is calculated depends also on the particular company . In our genetic algorithm, the most important (minimisation) criteria used are the following: (i) total cost of duties, (ii) number of single leftovers, (iii) number of grouped leftovers (if we assume that the successive uncovered pieces-of-work are grouped as a single leftover), (iv) total duration of leftovers, (v) deviation from a target mean duration of duties and leftovers .

Generation of the initial population :

- For each chromosome a list of the available duties is created. A duty is available if none of the pieces-of-work it covers is already covered by any duty in that solution .An available duty is randomly selected and inserted in the chromosome and this process is repeated until there are no more available duties. Naturally, every time a duty is added to the chromosome, the list of available duties is updated.

The fitness function:

- the sum of the individual costs of each duty in the solution. The cost of a duty is computed as a function of the duration, size of the breaks, extra work, etc. In relaxed partitioning solutions, leftovers must also be penalised

In real companies there are a set of objectives and constraints that cannot be included in the cost of each duty

Parent selection scheme

- assigns a probability for reproduction to each individual in a population, we have used the proportionate selection (roulette wheel) and the tournament selection (binary tournament).
- The roulette wheel contains one slot for each population element. The size of each slot is directly proportional to its respective ps(i), and therefore population members with higher fitness values are likely to be selected more often than those with lower fitness values.
- The tournament selection method is much simpler and more efficient than the proportionate selection method since it does not require calculating the fitness values of each individual

Crossover

- The crossover operator is applied to pairs of selected chromosomes in order to generate one, two or more ‘children’

The procedure is divided into two phases.

- In the first phase, we adopt an approach similar to the one used to generate the members of the initial population

- In the second phase we try to reduce the leftovers that may still exist. For each uncovered piece-of-work, randomly chosen, the subset of available generated duties that cover that piece-of-work is constructed. One of these duties is selected according to some rule. Hence, while there is still a not covered piece-of-work, an available duty from the set of all the generated duties is selected and added to the chromosome

Mutation:

- basic_mutation is based on the same philosophy we have used for the crossover operator and works as TG Dias et al— GAs for the bus driver scheduling problem

- A small percentage of duties is removed from the selected chromosome and a list with all the duties that have become available is built

- The second mutation operator, called improve_mutation, is a knowledge-based operator and it is used to directly improve a given solution through small changes in its neighbourhood

Population replacement scheme:

A fundamental decision in a genetic algorithm implementation is the choice of the strategy for the replacement of populations
In a steady-state replacement scheme, a percentage of the population (given by the generation gap parameter) is replaced by the new offspring

We have tried different values for the generation gap, and steady-state replacement was the strategy normally adopted with a generation gap of at most 0.25 (this means that no more than 25% of the population is replaced in each generation).