Abstract
- Revisit several existing traffic control algorithms and analyze their efficiency.
- Simplify the traffic model for better understanding of efficiency.
- Every vehicle travels eastward or southward only.
- Result: Efficiency largely depends on the length of a longest route and the number of cars.

Preliminaries
- Streets are laid in full grid and go only to south or east. (See Figure 1.)
- Streets have unit length but can accommodate unlimited number of cars.
- Traffic light at every intersection.
- All cars enter at time 0.
- Cars hop at most one street segment per unit time. No acceleration.
- If a street is dispatched, every car on that street moves to next street.

Notation 1 \( L \) denotes the length of a longest route in the instance.

Notation 2 \( n \) denotes the number of vehicles in the instance.

Algorithm 1: Alternate Dispatch, Ignoring Cars

Idea: If time is odd, dispatch horizontal streets. If time is even, dispatch vertical streets.
- Natural: Regular traffic lights
- One of the few possible algorithms that do not require traffic sensors.

Upper Bound: \( 2L \)
Let \( L' \) be the maximum number of street segments in one direction over all the routes, which is at most \( L \). Algorithm 1 will run for at most \( 2L' \).

Lower Bound: \( 2L \)
All routes go from top to bottom. (See Figure 2 for \( n = 3 \).)

Algorithm 2: Main Streets Always Go

Idea: If a car is not in conflict, let it move. Otherwise, always block the car on vertical streets. Cars on horizontal streets always have a priority.
- Plausible: Sometimes a certain street has more traffic volume. For example, cars on 34th Street should always yield to cars on Walnut Street.
- Resembles a stop-sign strategy.

Upper Bound: \( L + n - 1 \)
- Any vehicle can be blocked for at most \( n - 1 \) times. Otherwise, there must be at least \( n \) other vehicles between a vehicle’s original and final positions, which is impossible.
- Algorithm 2 must also run for at least \( L \) time units.
- Therefore, after \( L + n - 1 \) time units, every vehicle must have finished its journey.

Lower Bound: \( L + n - 1 \)
Figure 3 shows an instance for \( n = 6 \) where vehicle 6 is blocked exactly \( n - 1 \) times, but the maximum finish time is \( n/2 + n - 1 \). To obtain the bound, simply augment the route of vehicle \( n \) (6 for the Figure) for \( n/2 - 1 \) more segments. (Note: \( L = n - 1 \).)

Algorithm 3: Alternate Dispatch, a Smarter Version

Idea: For each individual intersection, if a car is not in conflict, let it move. Otherwise, alternately dispatch street segments using parity bit. If bit is 0, dispatch horizontal street and flip bit; if bit is 1, dispatch vertical street and flip bit.
- Every traffic light starts with parity bit 0.
- Fairer than Algorithm 2.

Upper Bound: \( L + n - 1 \)
Proofs are identical to the upper bound of Algorithm 2.

Lower Bound: \( L + n - 1 \)
Figure 4 shows an instance for \( n = 6 \) where vehicle 6 is blocked exactly \( n - 1 \) times, but the maximum finish time is \( m + n - 3 \), where \( m = n/2 \). To obtain the bound, simply augment the route of vehicle \( n \) for \( n/2 - 1 \) more segments. (Note: \( L = 2n - 3 \).)

Conclusions: Food for Thought
- Algorithm 1 is better when all cars travel in short distances. Otherwise, more cars lead to more congestion.
- Algorithm 2 performs badly when many cars use “side streets.” Algorithm 3 outperforms Algorithm 2.
- Traffic sensors are needed for Algorithms 2 and 3 to determine approaching cars.
- Drive less to save gas and time. Drive longer and get stuck in the traffic easier.
- Zigzagging might appear to avoid traffic, but this might worsen the traffic overall.