The Complexity of Counting Turns in the Line-Based Dial-a-Ride Problem

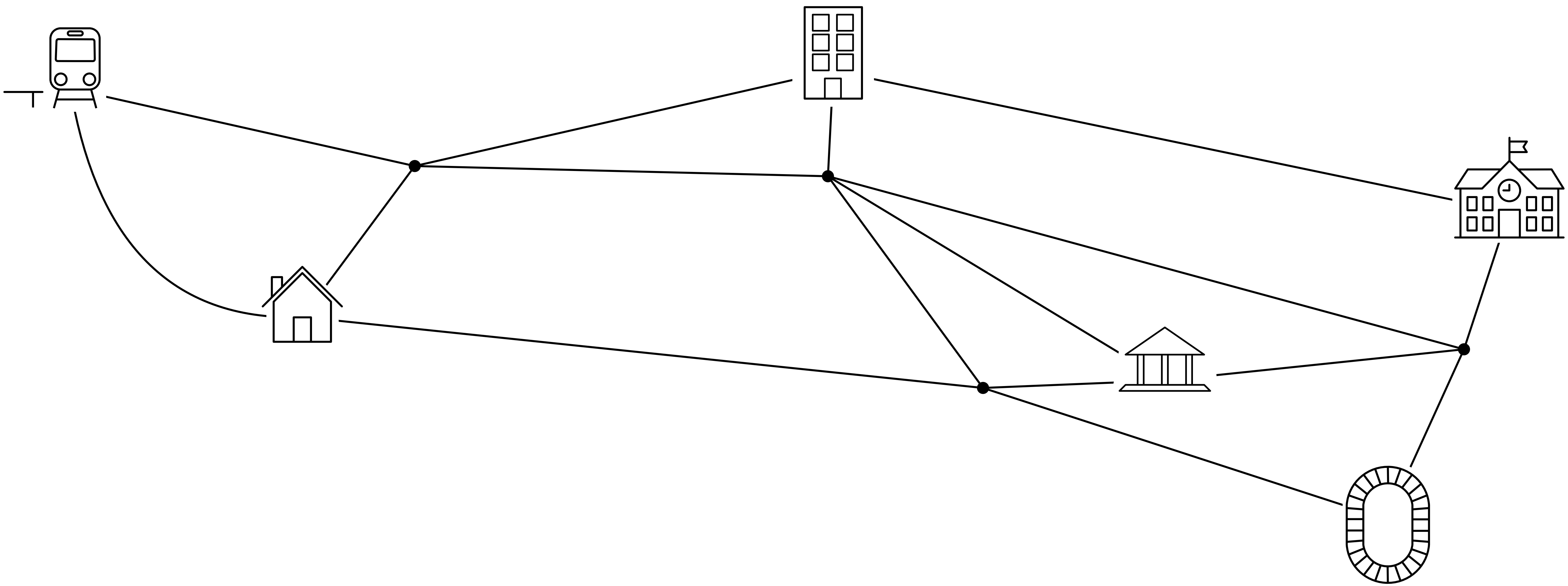
SOFSEM 25

Antonio Lauerbach Kendra Reiter Marie Schmidt

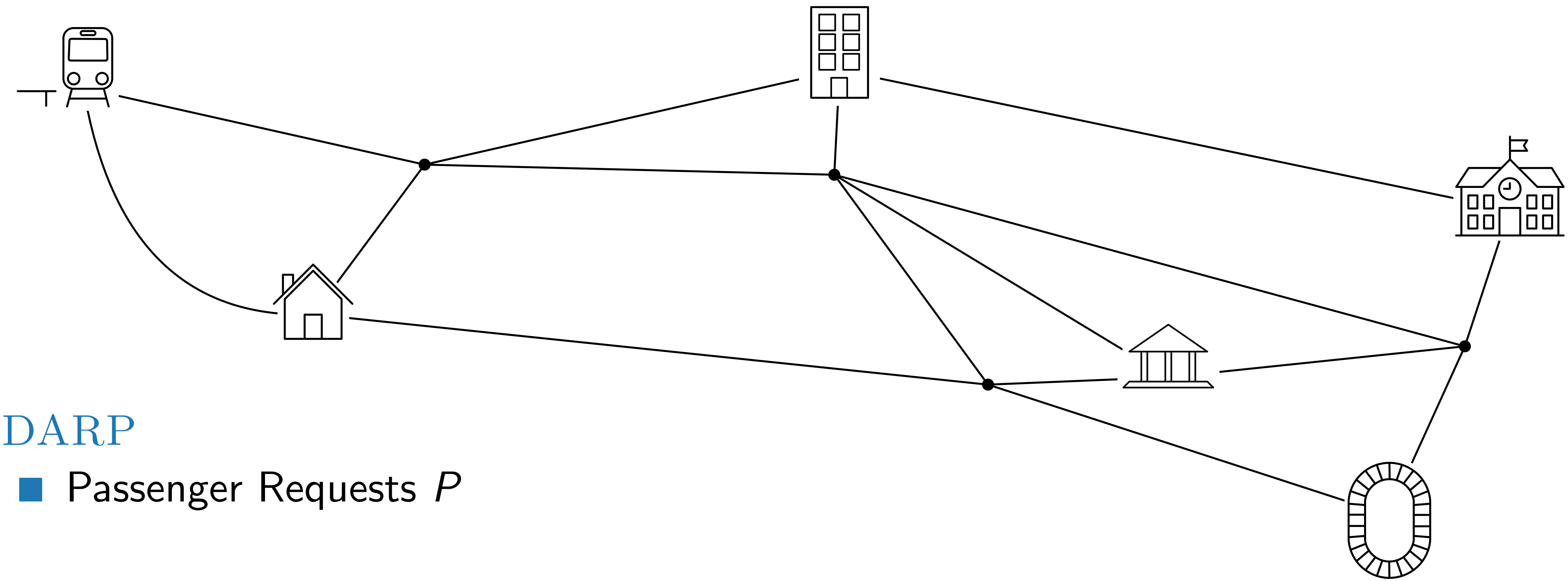
Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Dial-a-Ride-Problem (DARP)

Dial-a-Ride-Problem (DARP)



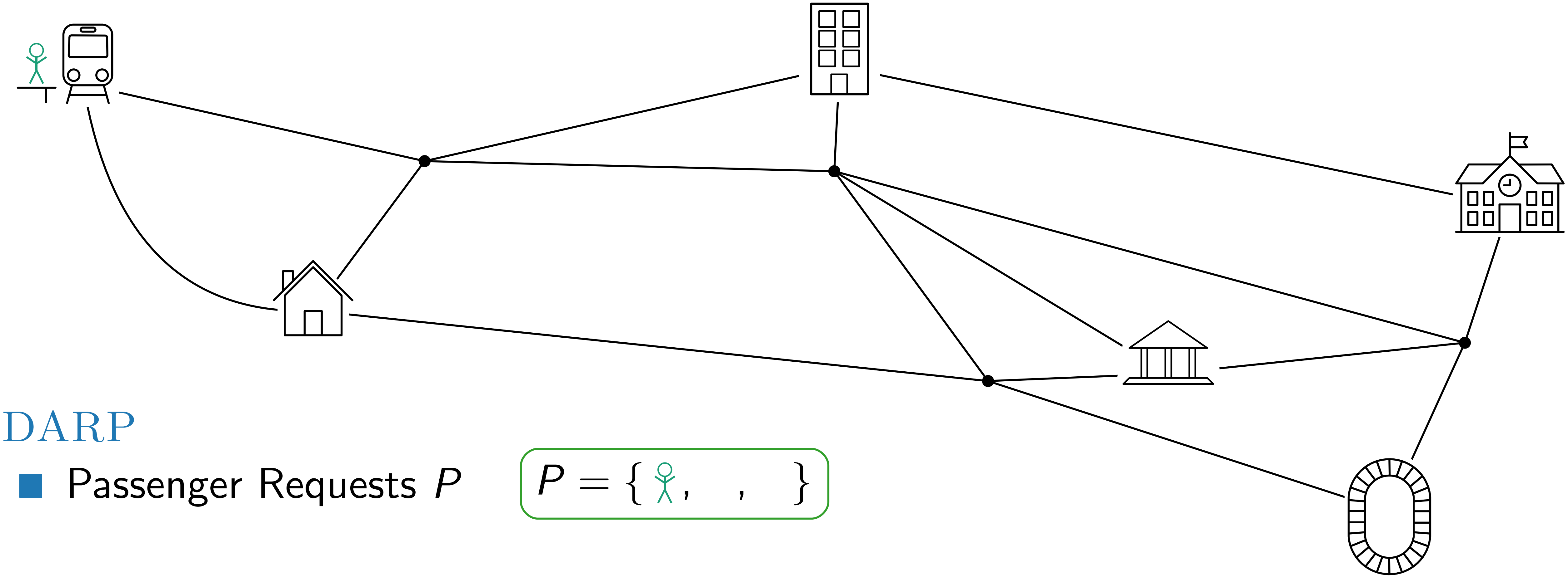
Dial-a-Ride-Problem (DARP)



DARP

- Passenger Requests P

Dial-a-Ride-Problem (DARP)

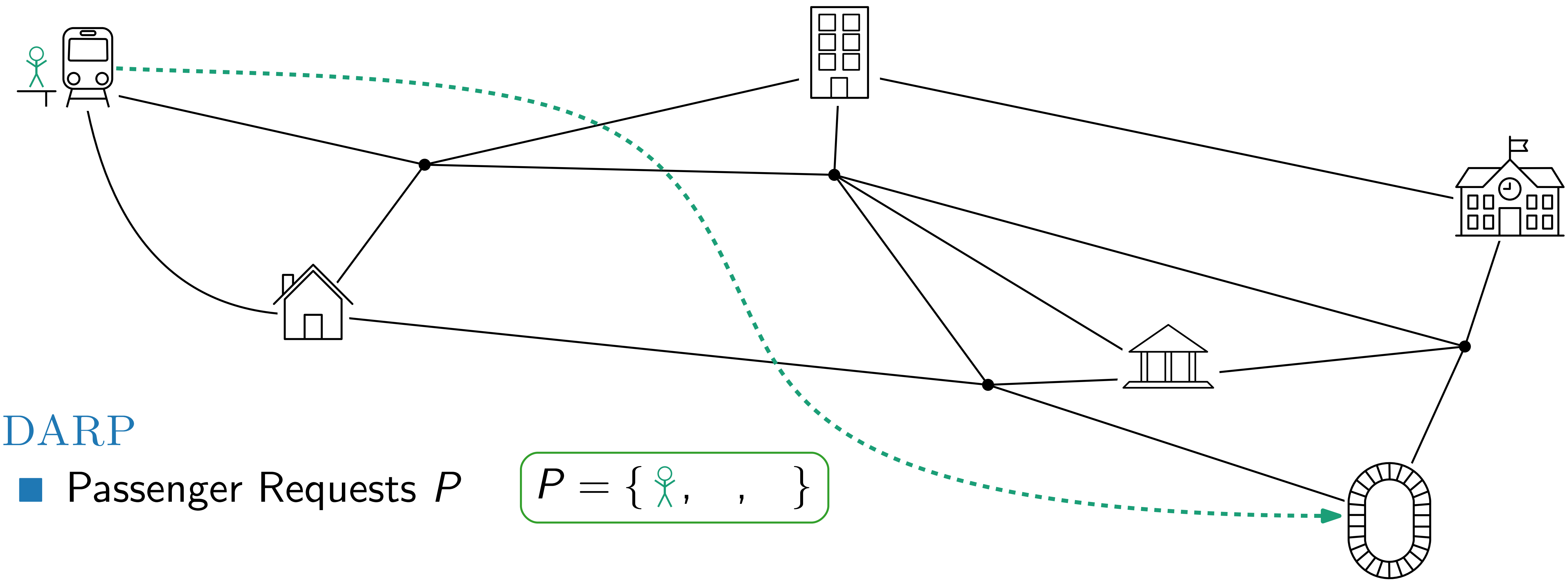


DARP

- Passenger Requests P

$P = \{\ \ ,\ \ ,\ \ \}$

Dial-a-Ride-Problem (DARP)

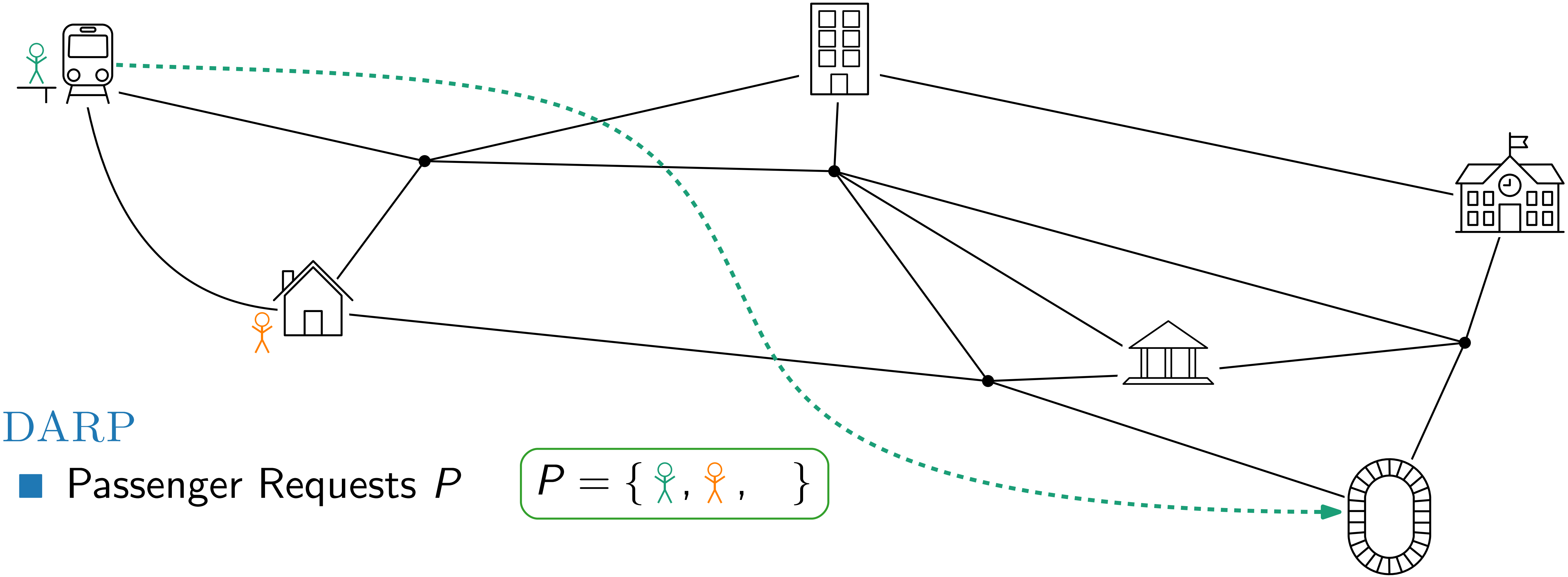


DARP

- Passenger Requests P

$P = \{\ \ ,\ \ ,\ \ \}$

Dial-a-Ride-Problem (DARP)

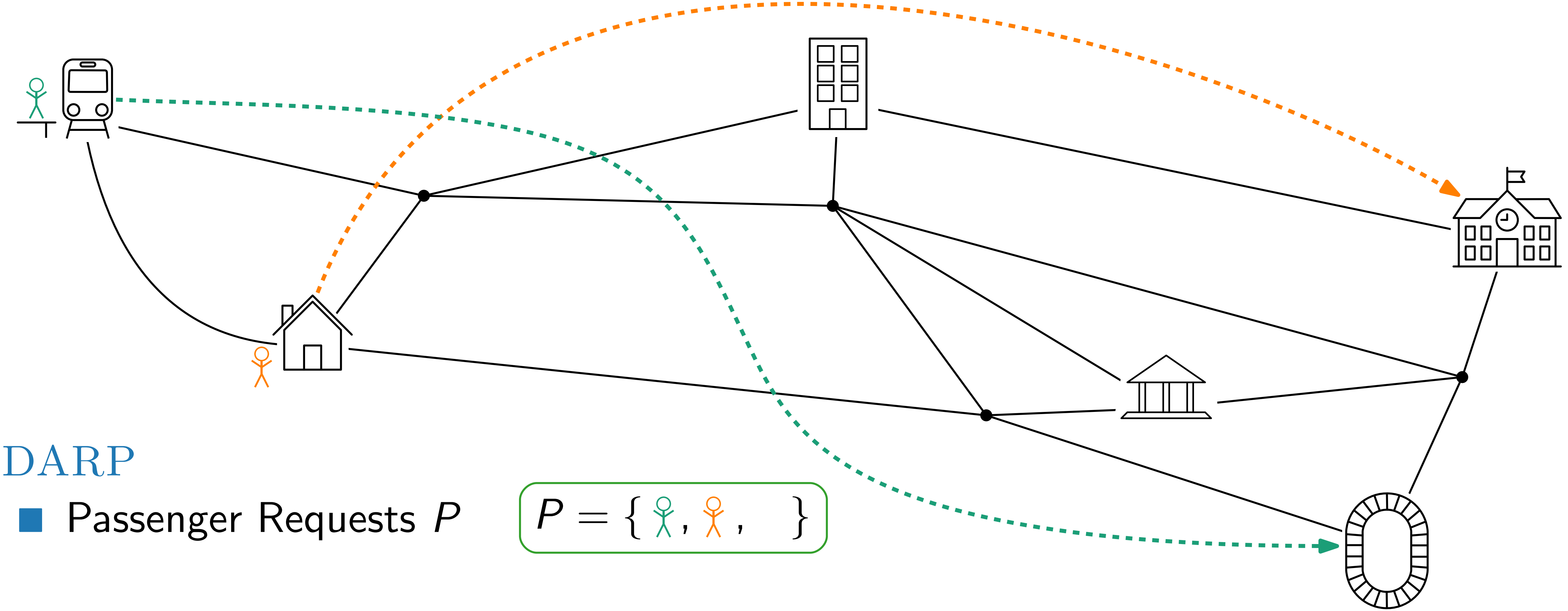


DARP

- Passenger Requests P

$$P = \{ \; , \; , \quad \}$$

Dial-a-Ride-Problem (DARP)

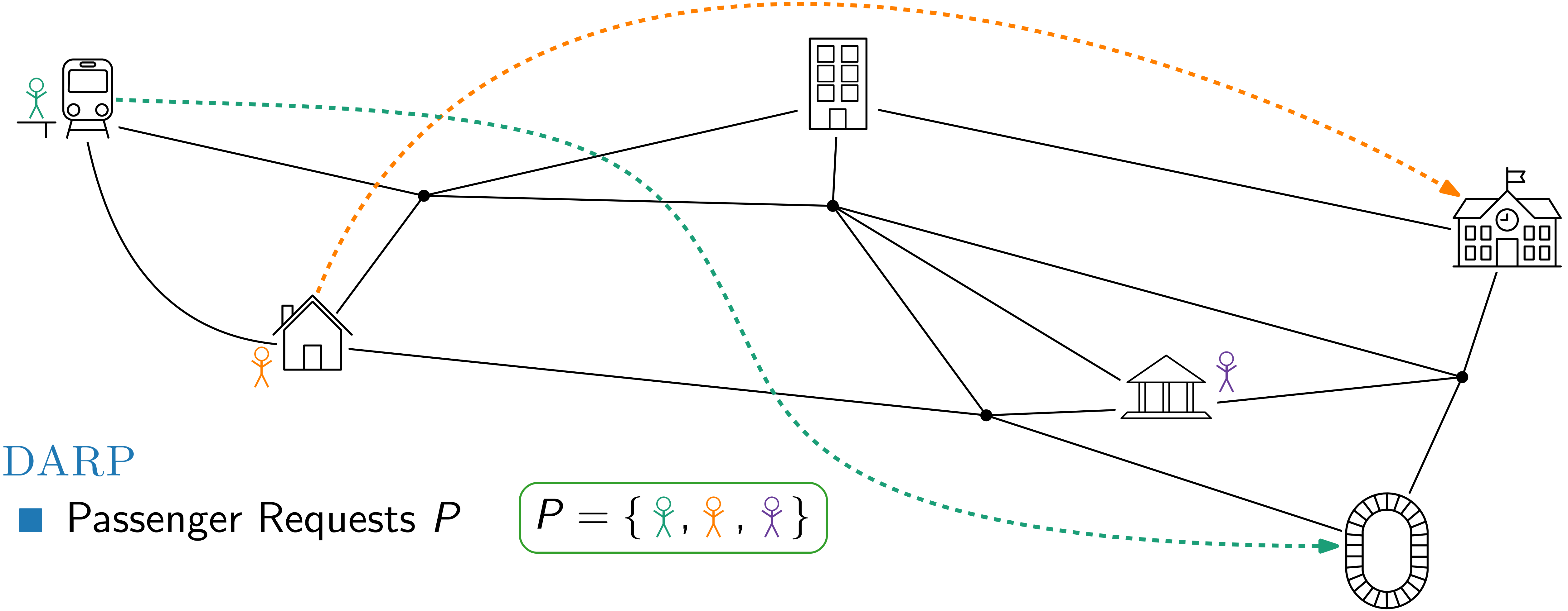


DARP

- Passenger Requests P

$P = \{ \quad , \quad , \quad \}$

Dial-a-Ride-Problem (DARP)

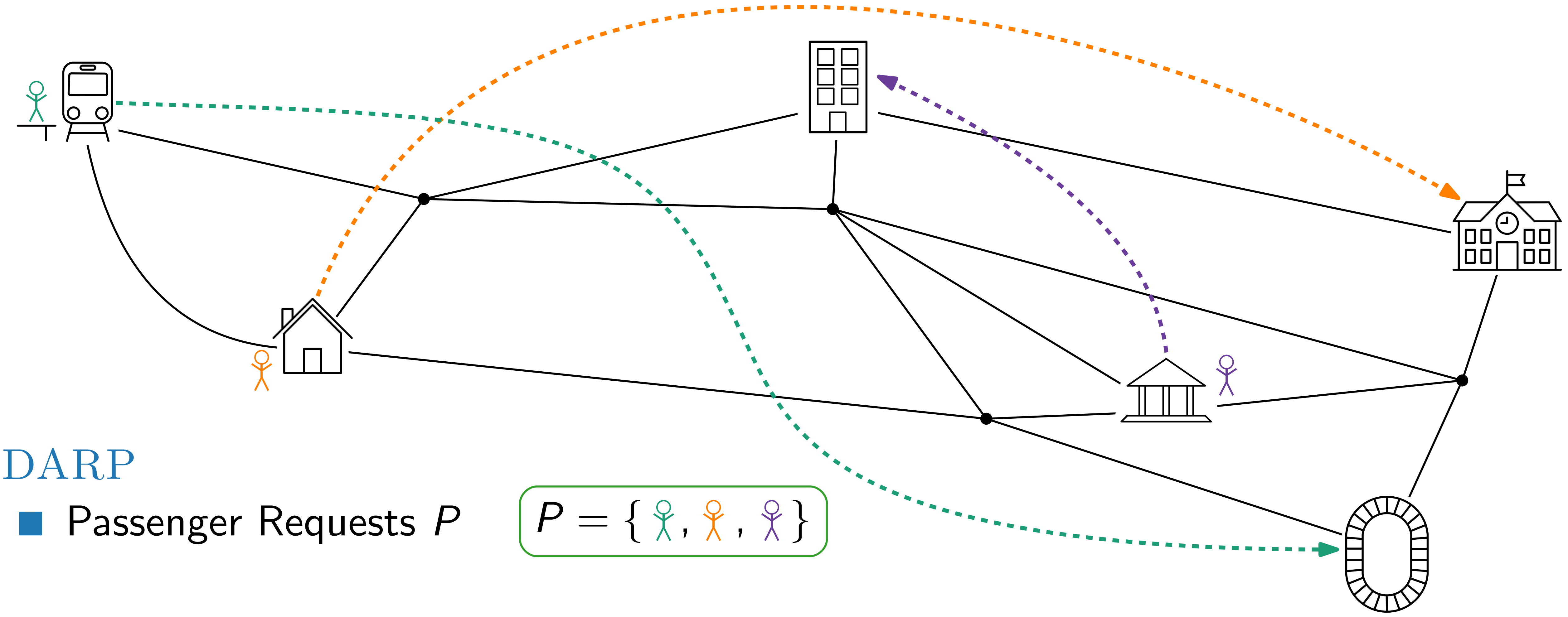


DARP

- Passenger Requests P

$$P = \{\ ,\ ,\ \}$$

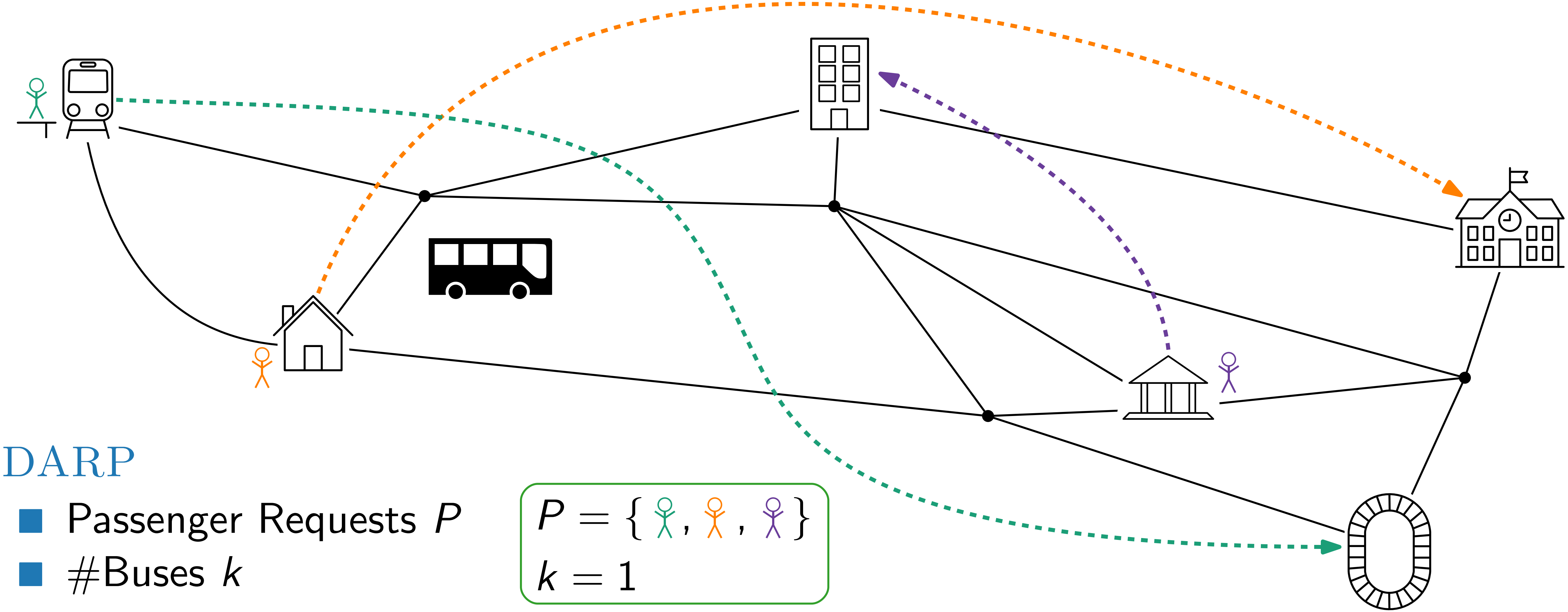
Dial-a-Ride-Problem (DARP)



DARP

- Passenger Requests P

Dial-a-Ride-Problem (DARP)

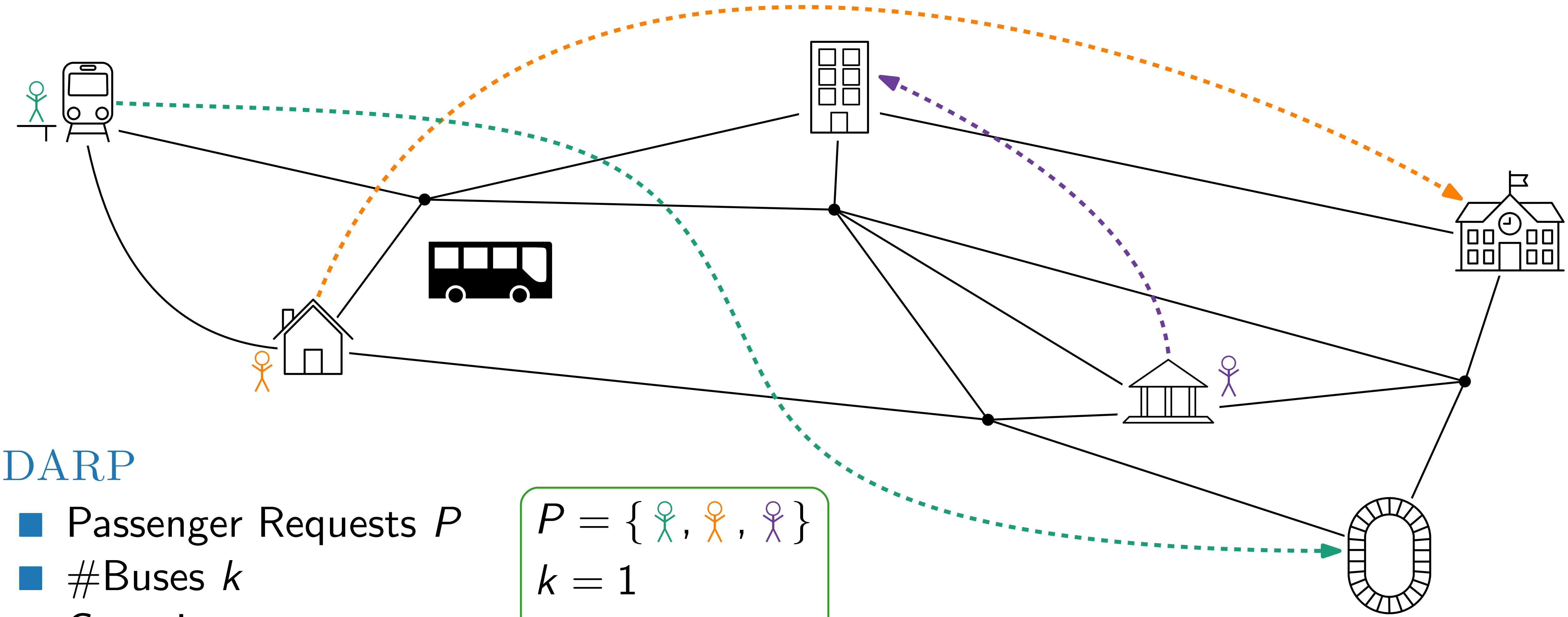


DARP

- Passenger Requests P
- #Buses k

$P = \{$ 🧍, 🧍, 🧍 $\}$
$k = 1$

Dial-a-Ride-Problem (DARP)



DARP

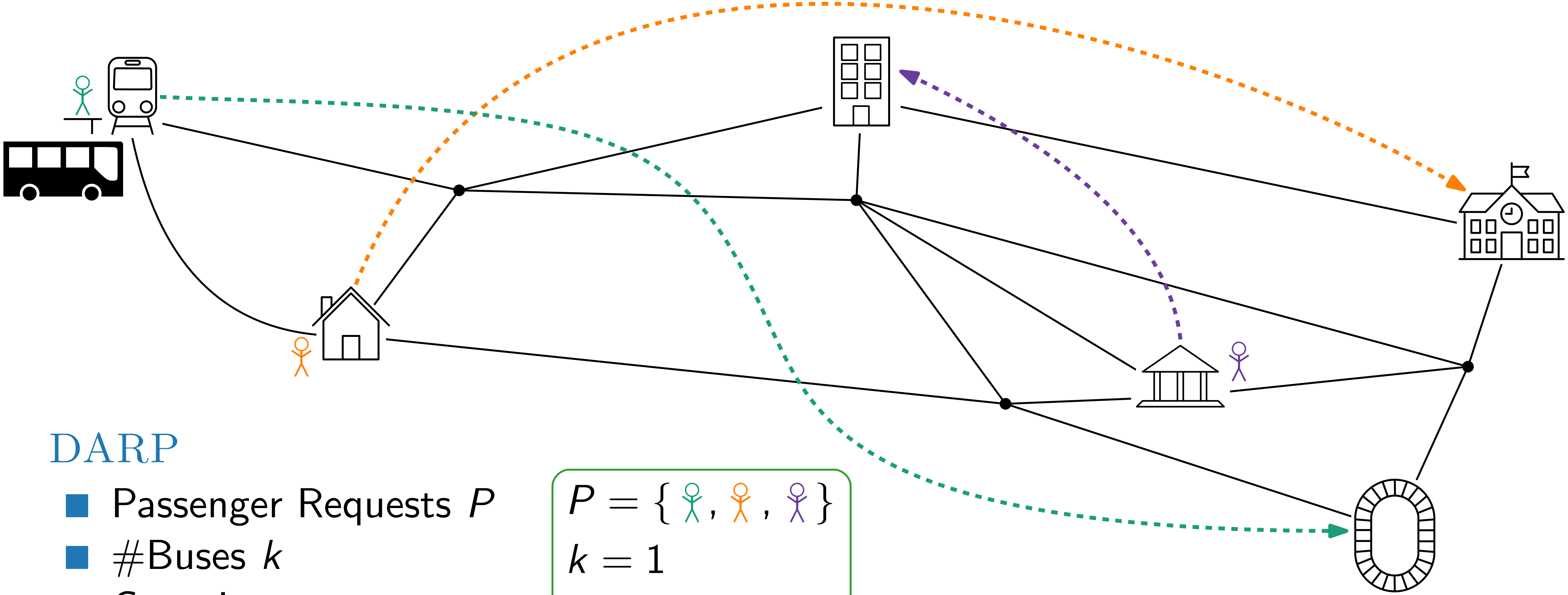
- Passenger Requests P
- #Buses k
- Capacity c

$P = \{\ ,\ ,\ \}$

$k = 1$

$c = 3$

Dial-a-Ride-Problem (DARP)



DARP

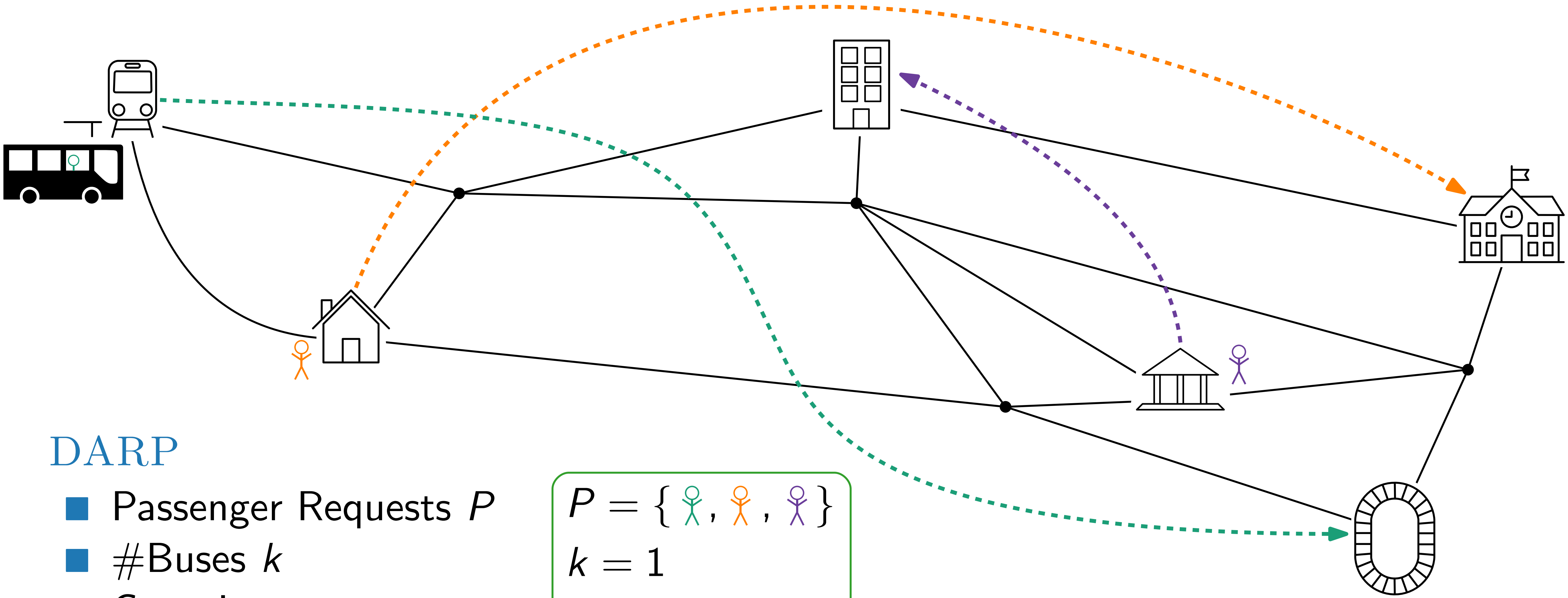
- Passenger Requests P
- #Buses k
- Capacity c

$P = \{$, , $\}$

$k = 1$

$c = 3$

Dial-a-Ride-Problem (DARP)



DARP

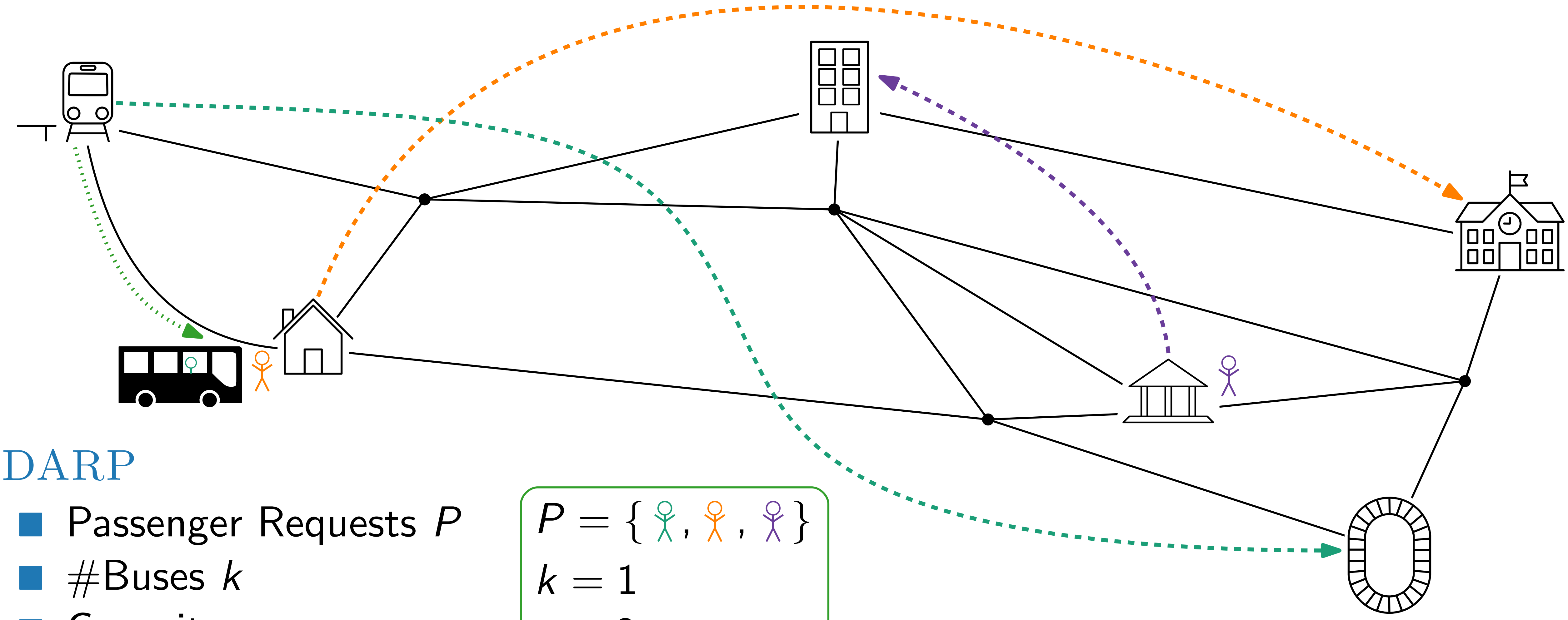
- Passenger Requests P
- #Buses k
- Capacity c

$P = \{$ 🧍, 🧍, 🧍 $\}$

$k = 1$

$c = 3$

Dial-a-Ride-Problem (DARP)



DARP

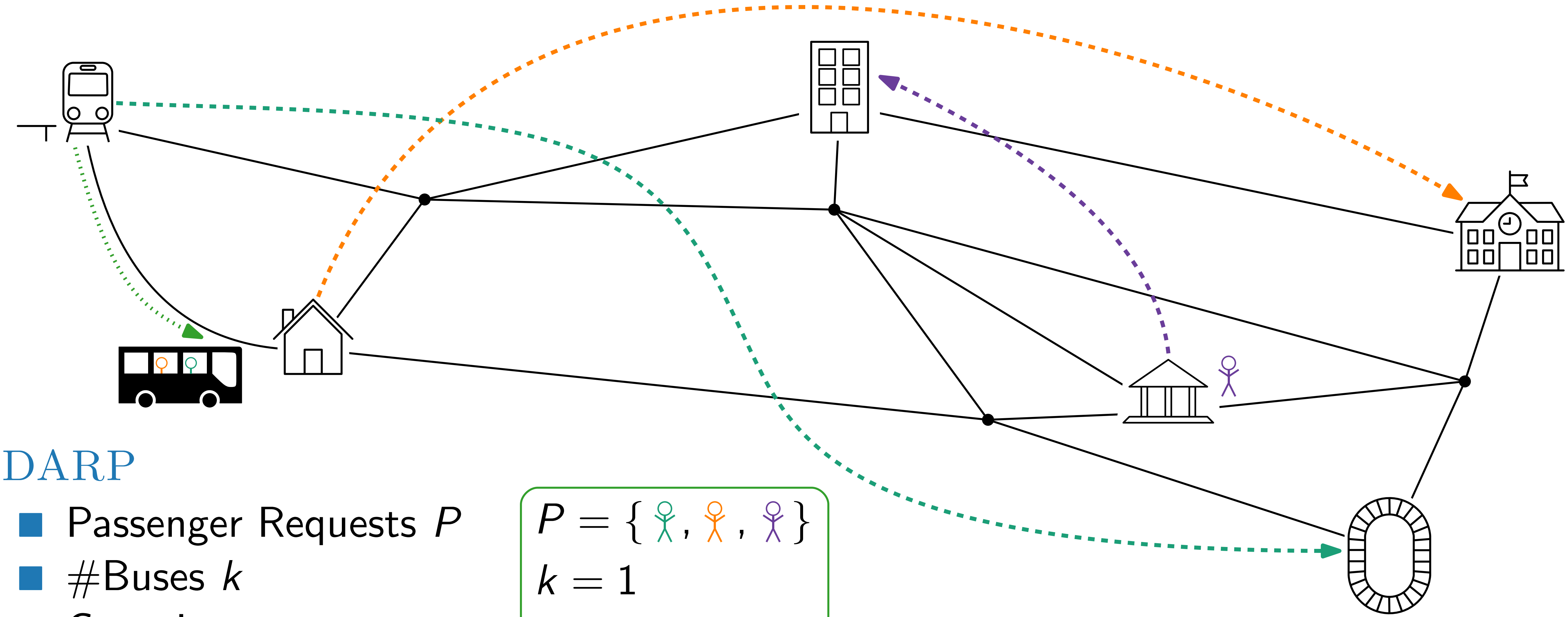
- Passenger Requests P
- #Buses k
- Capacity c

$P = \{ \text{🧍}, \text{🧍}, \text{🧍} \}$

$k = 1$

$c = 3$

Dial-a-Ride-Problem (DARP)



DARP

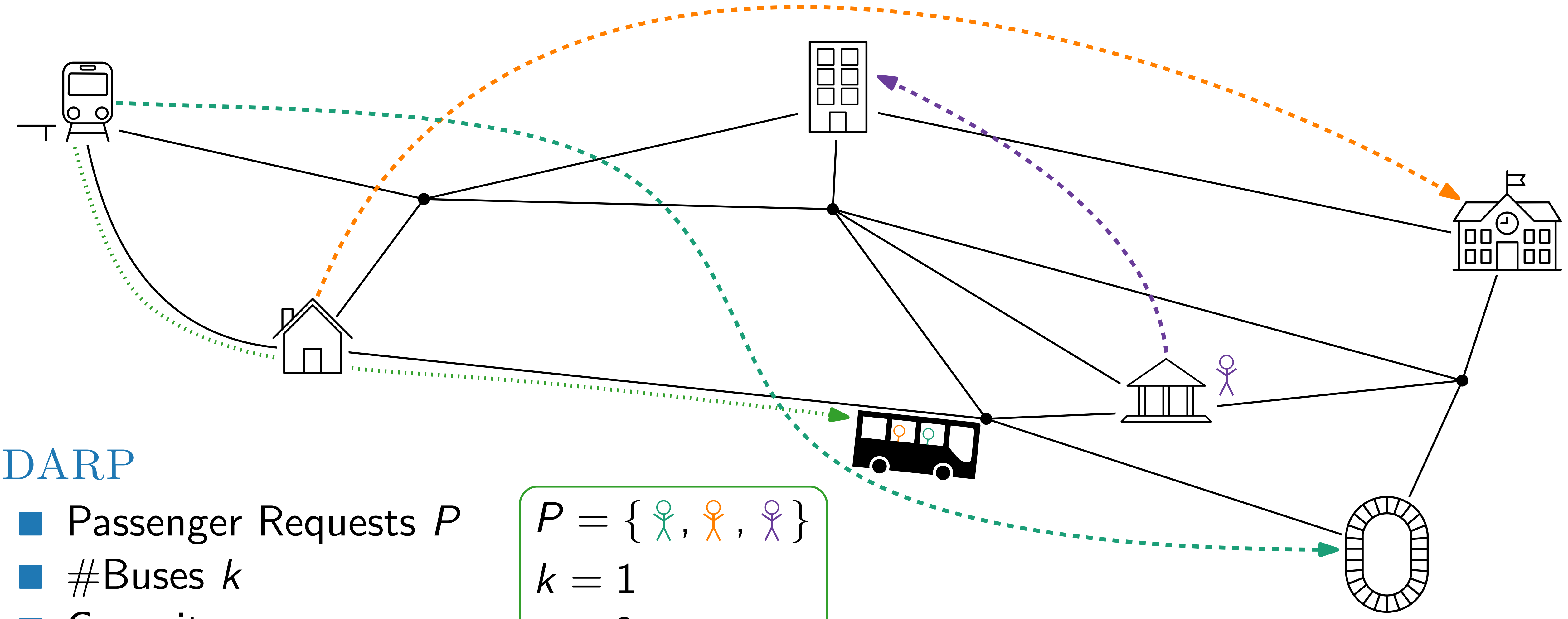
- Passenger Requests P
- #Buses k
- Capacity c

$P = \{ \text{(green passenger)}, \text{(orange passenger)}, \text{(purple passenger)} \}$

$k = 1$

$c = 3$

Dial-a-Ride-Problem (DARP)



DARP

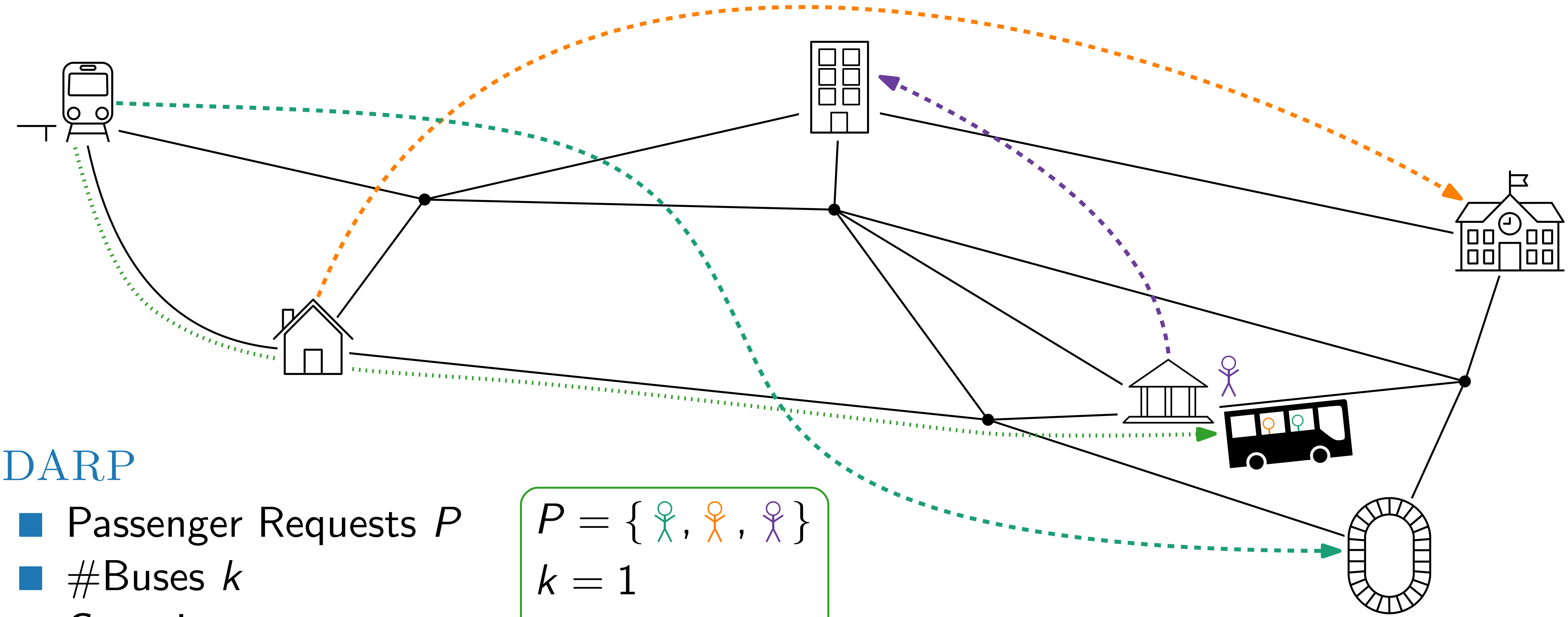
- Passenger Requests P
- #Buses k
- Capacity c

$P = \{ \text{🧍}, \text{🧍}, \text{🧍} \}$

$k = 1$

$c = 3$

Dial-a-Ride-Problem (DARP)



DARP

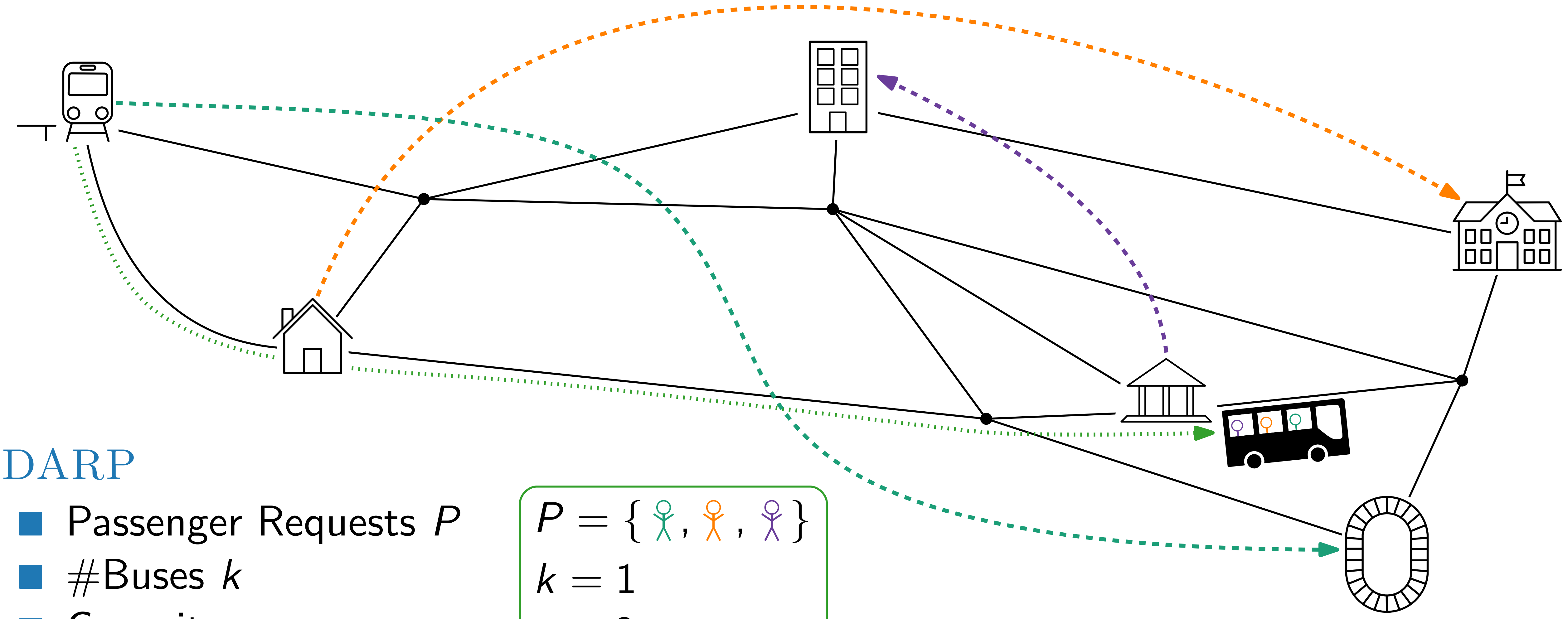
- Passenger Requests P
- #Buses k
- Capacity c

$P = \{ , , \}$

$k = 1$

$c = 3$

Dial-a-Ride-Problem (DARP)

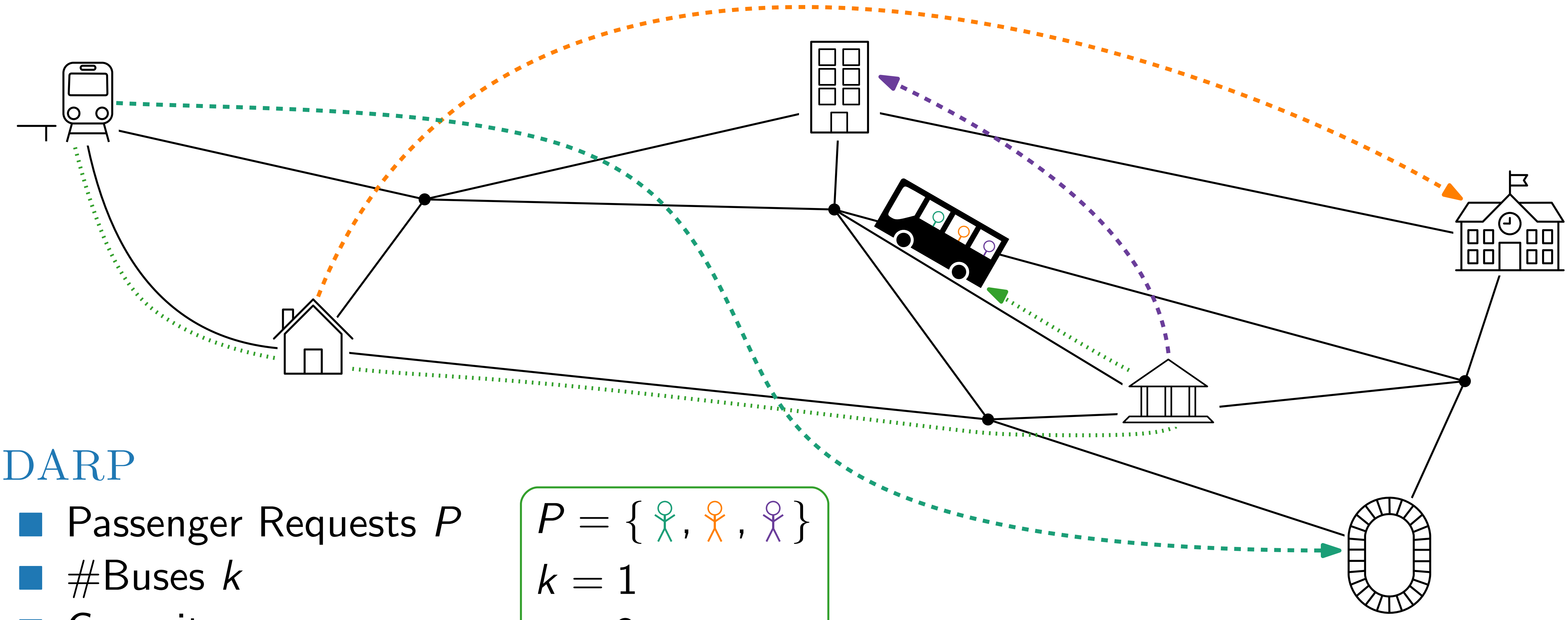


DARP

- Passenger Requests P
- #Buses k
- Capacity c

$P = \{$ 🯅 , 🯅 , 🯅 $\}$
$k = 1$
$c = 3$

Dial-a-Ride-Problem (DARP)



DARP

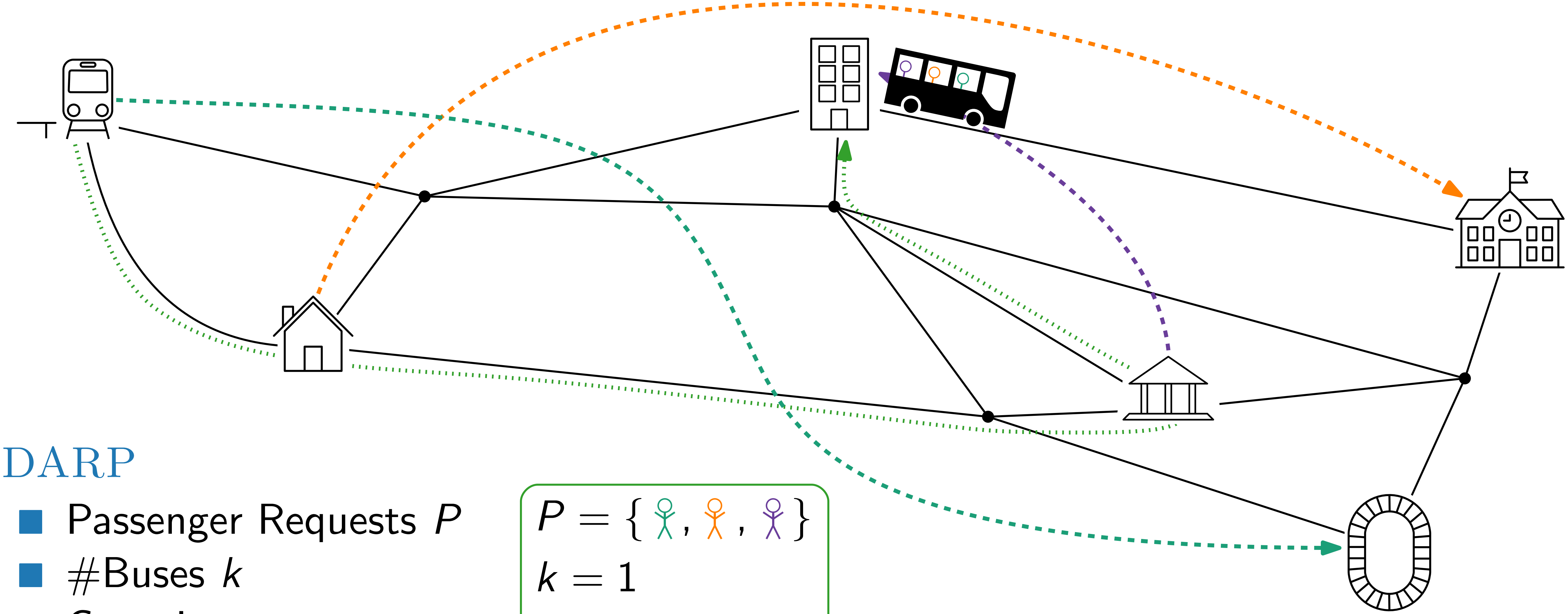
- Passenger Requests P
- #Buses k
- Capacity c

$P = \{ \text{passenger}, \text{passenger}, \text{passenger} \}$

$k = 1$

$c = 3$

Dial-a-Ride-Problem (DARP)



DARP

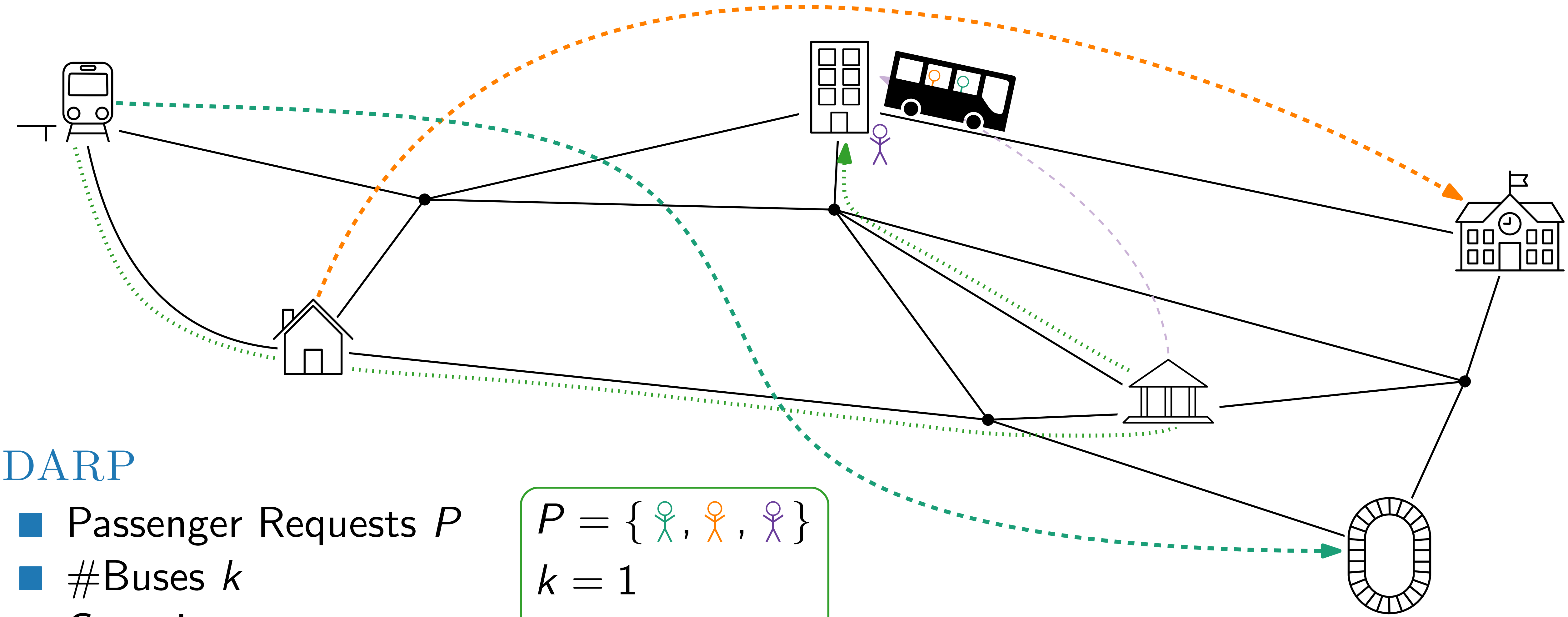
- Passenger Requests P
- #Buses k
- Capacity c

$P = \{ \; , \; , \; \}$

$k = 1$

$c = 3$

Dial-a-Ride-Problem (DARP)



DARP

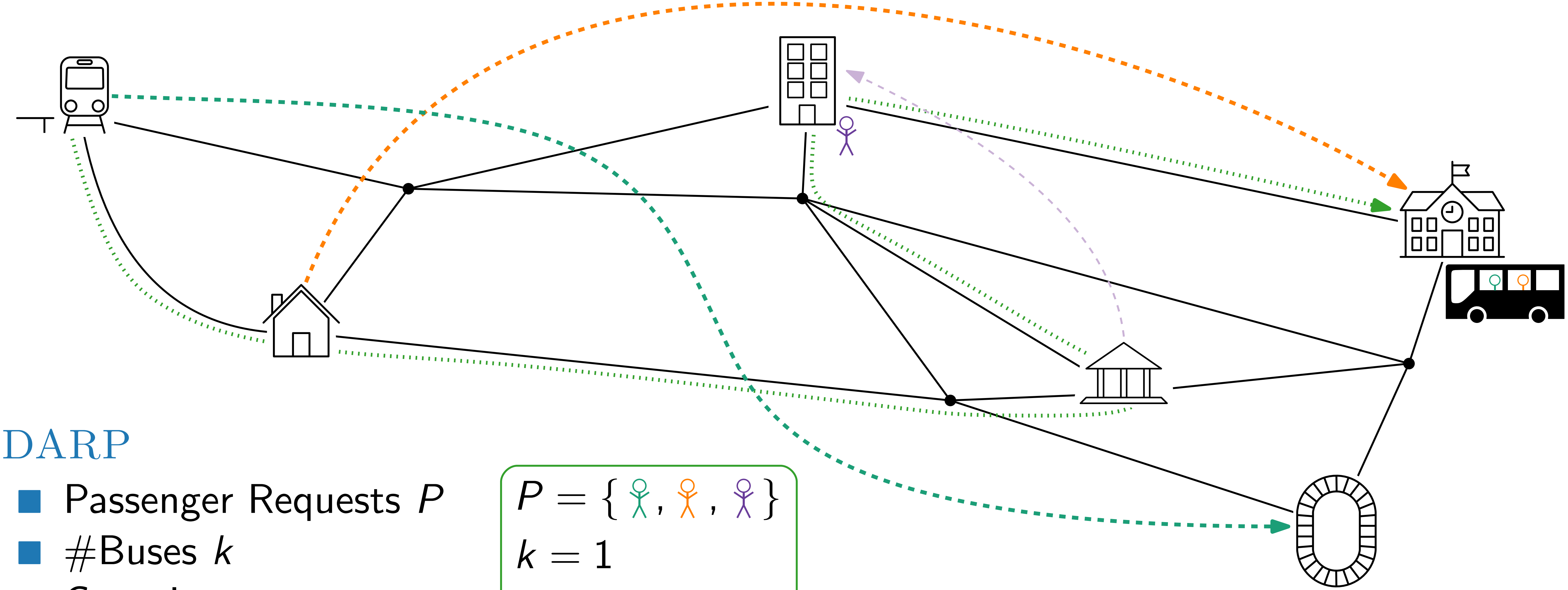
- Passenger Requests P
- #Buses k
- Capacity c

$P = \{ , , \}$

$k = 1$

$c = 3$

Dial-a-Ride-Problem (DARP)

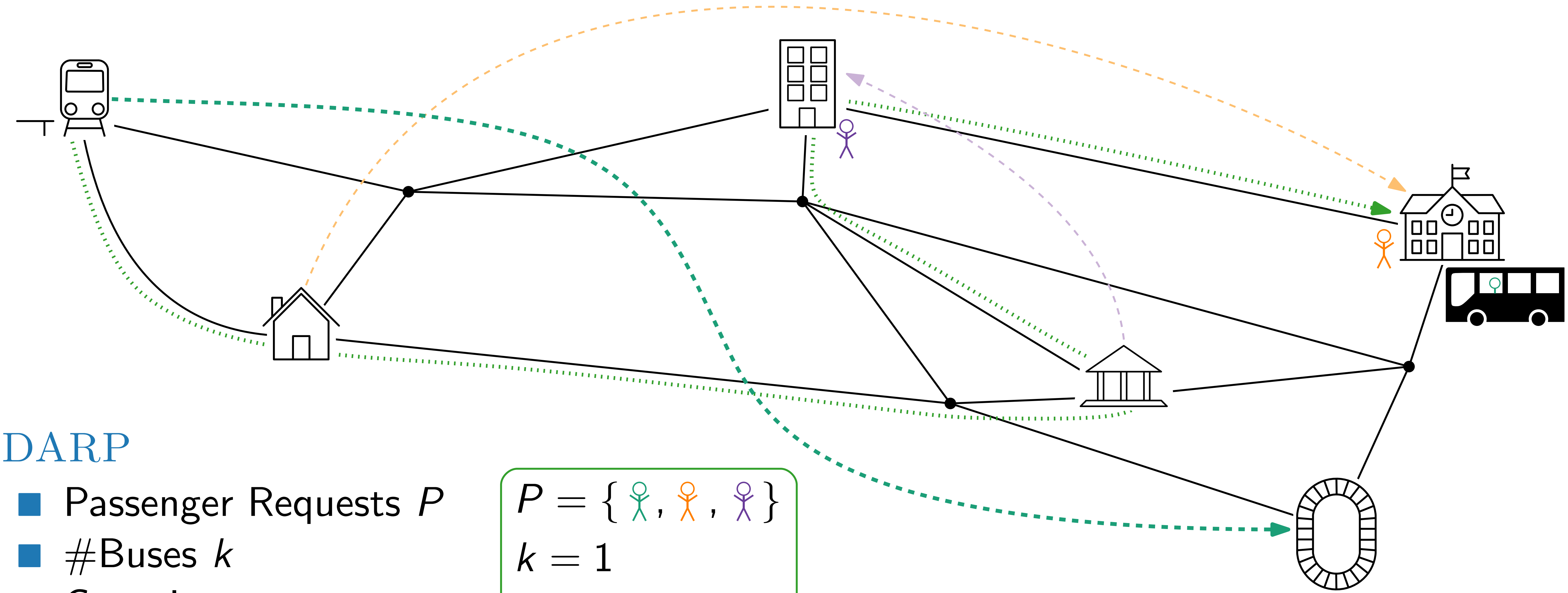


DARP

- Passenger Requests P
- #Buses k
- Capacity c

$P = \{$ ⚲, ⚲, ⚲ $\}$
$k = 1$
$c = 3$

Dial-a-Ride-Problem (DARP)



DARP

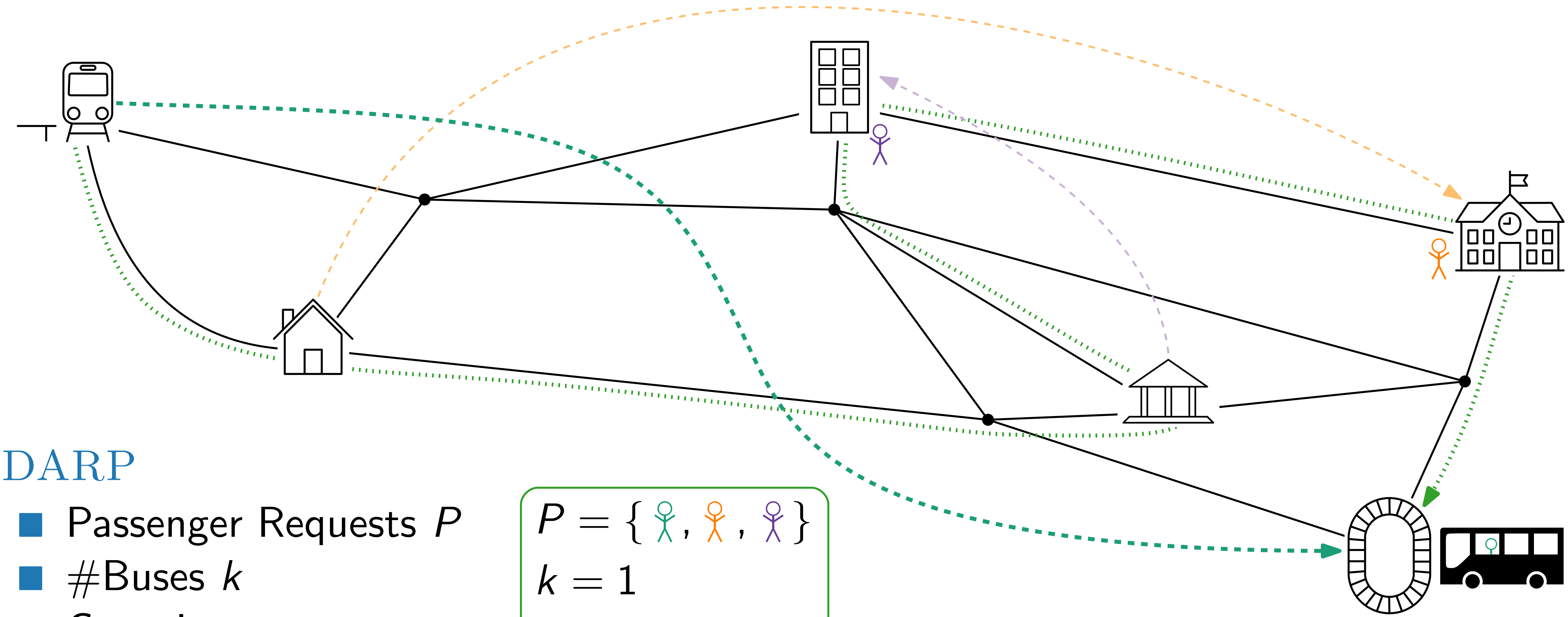
- Passenger Requests P
- #Buses k
- Capacity c

$P = \{ \text{ⓘ}, \text{ⓘ}, \text{ⓘ} \}$

$k = 1$

$c = 3$

Dial-a-Ride-Problem (DARP)

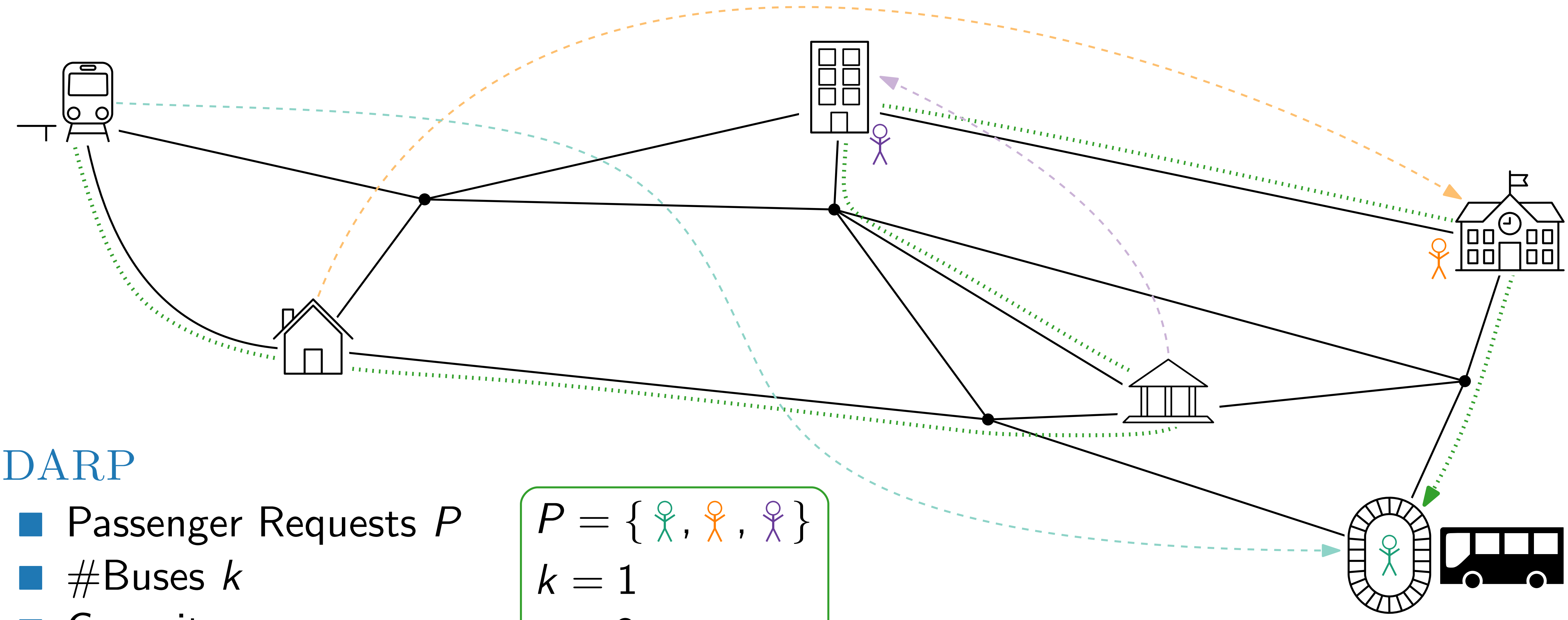


DARP

- Passenger Requests P
- #Buses k
- Capacity c

$P = \{ \text{☺}, \text{☺}, \text{☺} \}$
$k = 1$
$c = 3$

Dial-a-Ride-Problem (DARP)

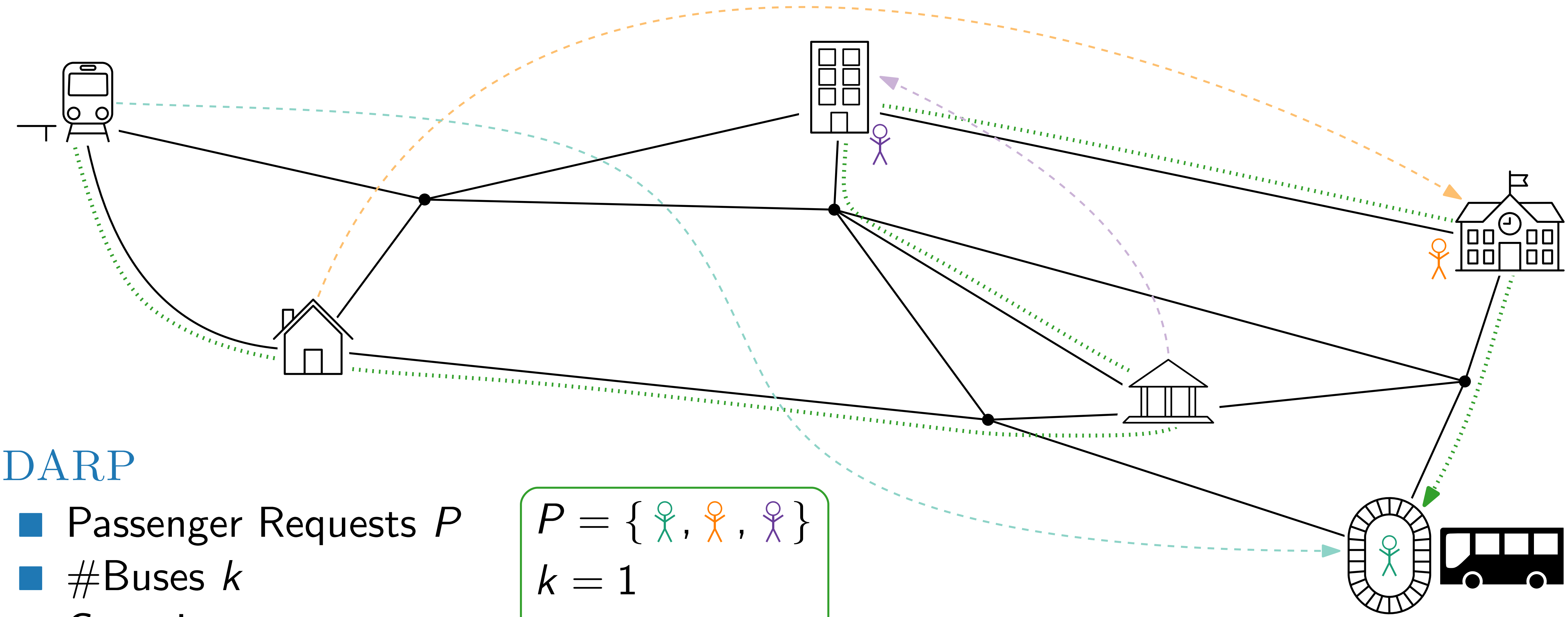


DARP

- Passenger Requests P
- #Buses k
- Capacity c

$P = \{ \ , \ , \ \}$
$k = 1$
$c = 3$

Dial-a-Ride-Problem (DARP)



DARP

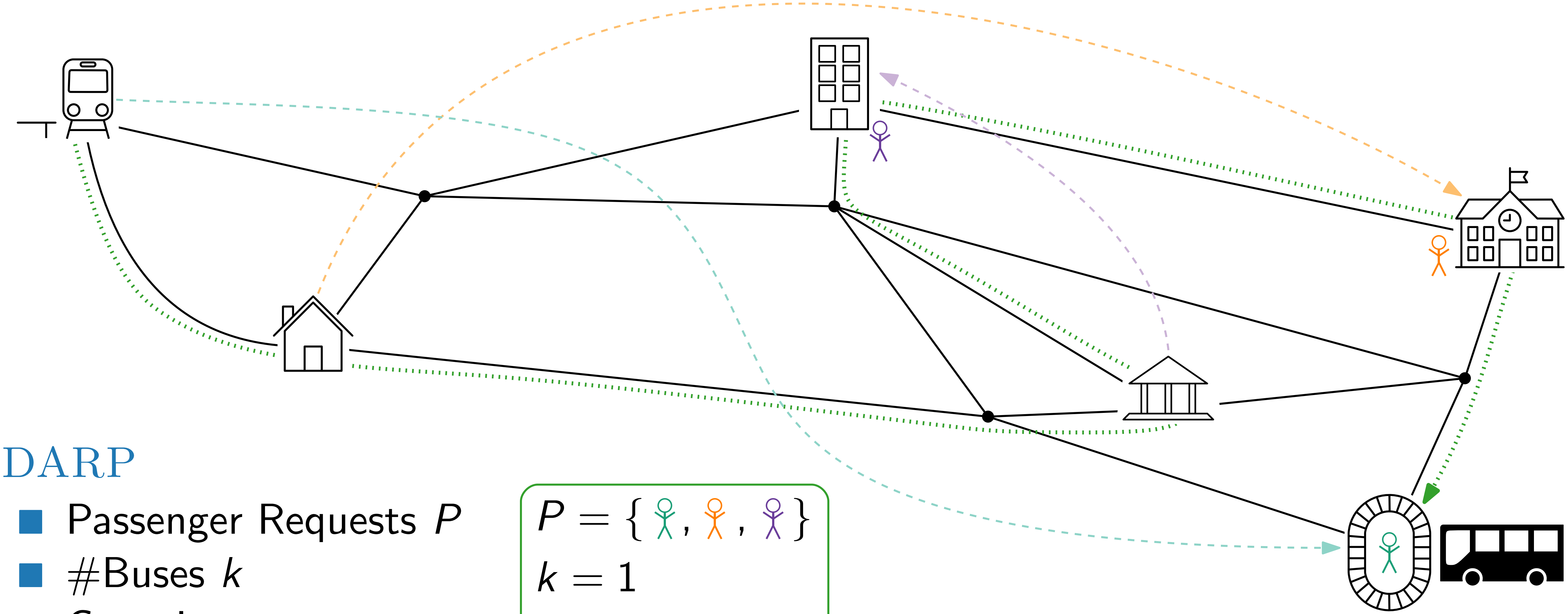
- Passenger Requests P
- #Buses k
- Capacity c
- Solution R

$P = \{$![green passenger], ![orange passenger], ![purple passenger] $\}$

$k = 1$

$c = 3$

Dial-a-Ride-Problem (DARP)



DARP

- Passenger Requests P
- #Buses k
- Capacity c
- Solution R

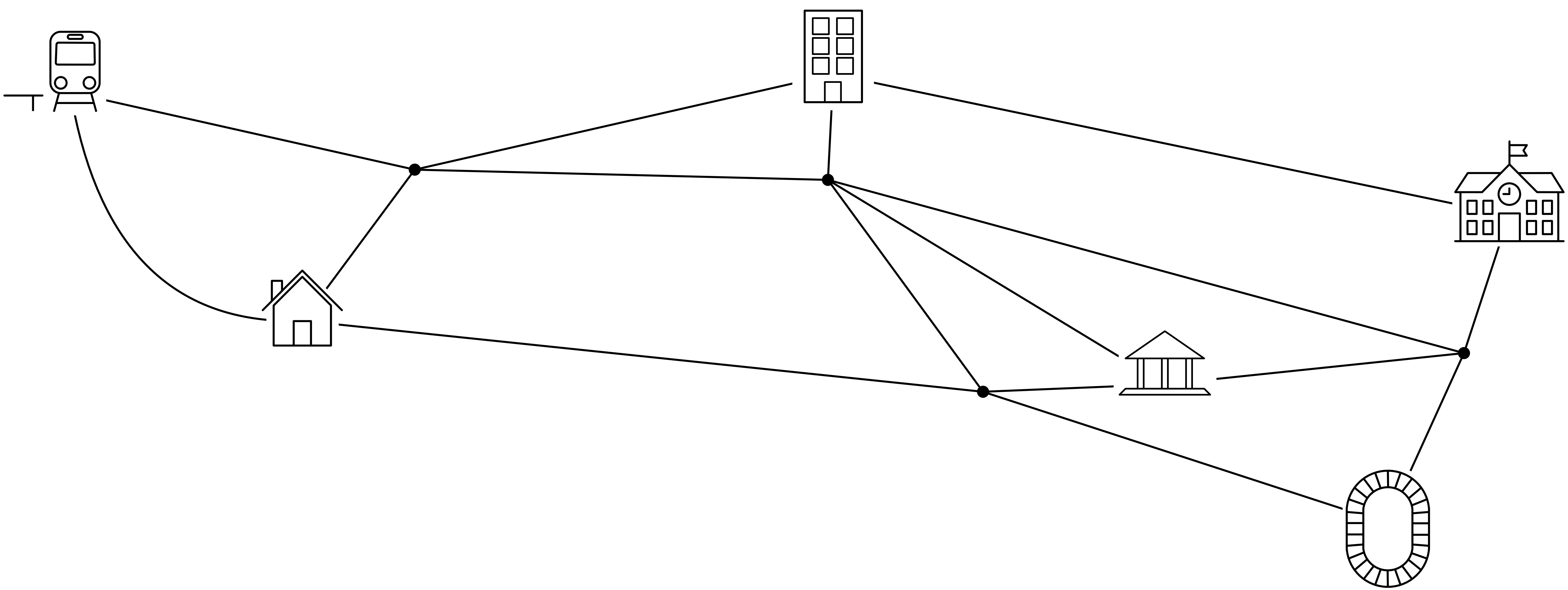
$P = \{\ ,\ ,\ \}$

$k = 1$

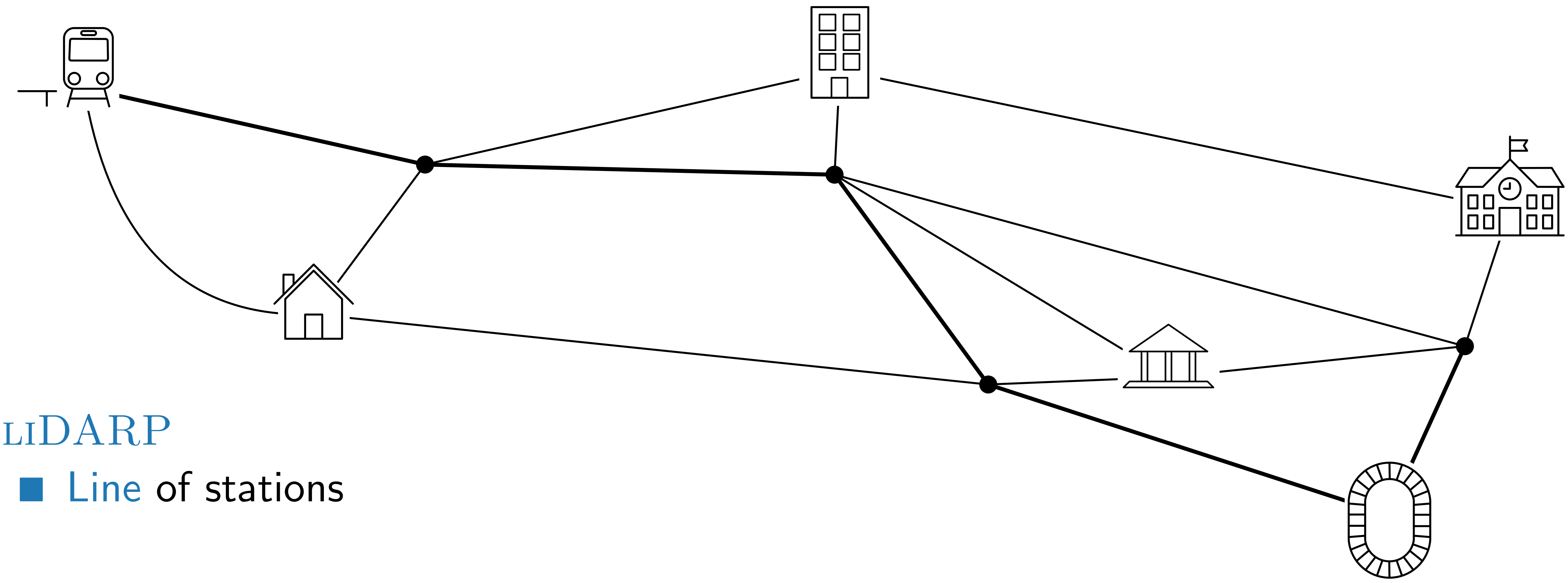
$c = 3$

$R = \{\ \}$

line-based Dial-a-Ride-Problem (liDARP)



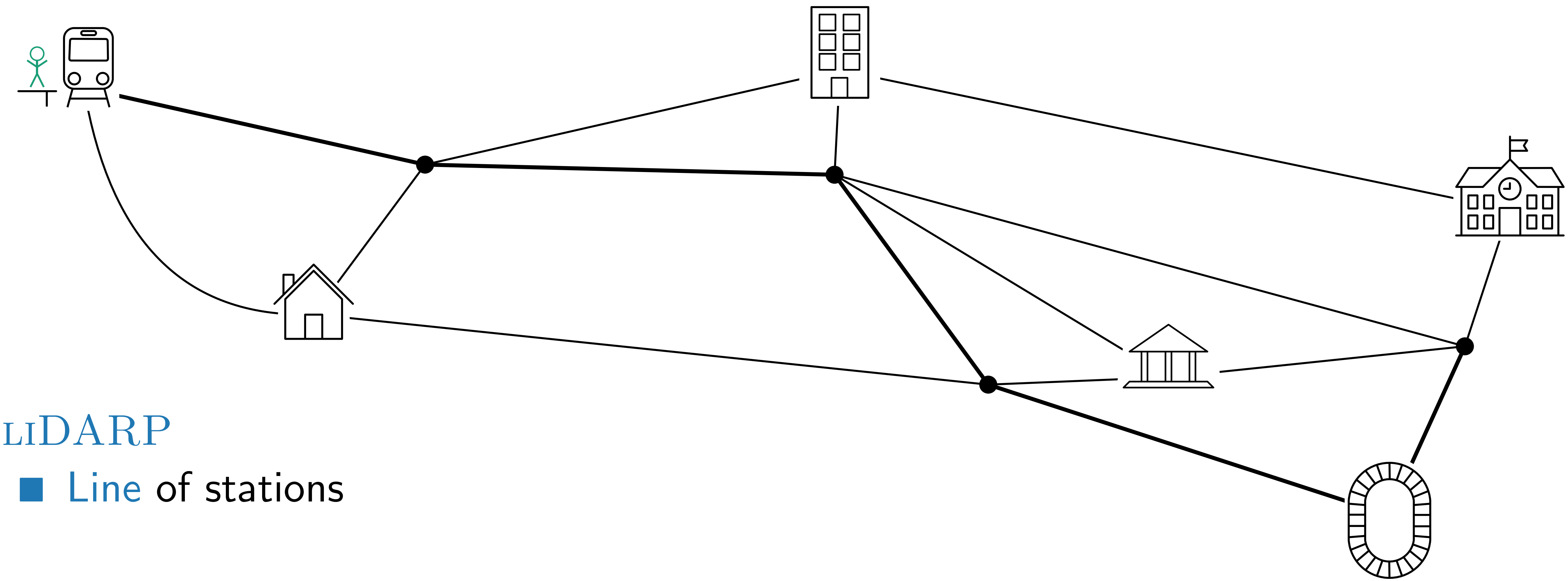
LINE-BASED DIAL-A-RIDE-PROBLEM (LIDARP)



LIDARP

- Line of stations

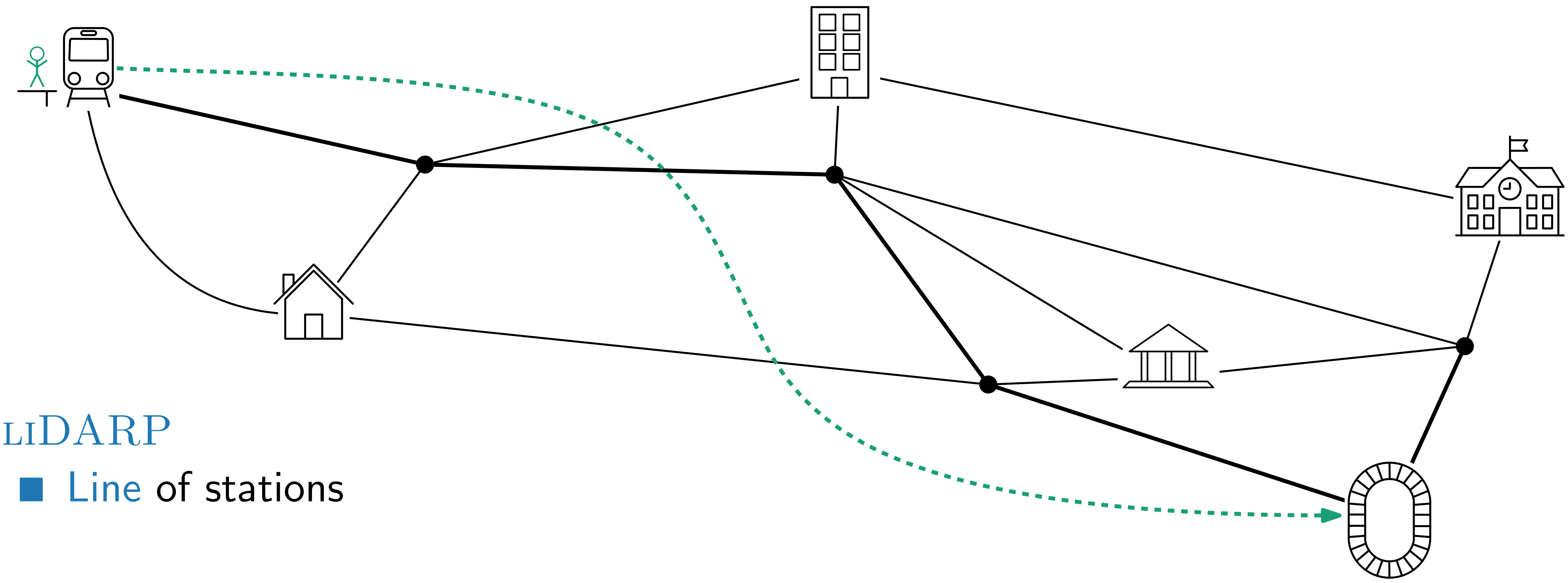
line-based Dial-a-Ride-Problem (liDARP)



liDARP

- Line of stations

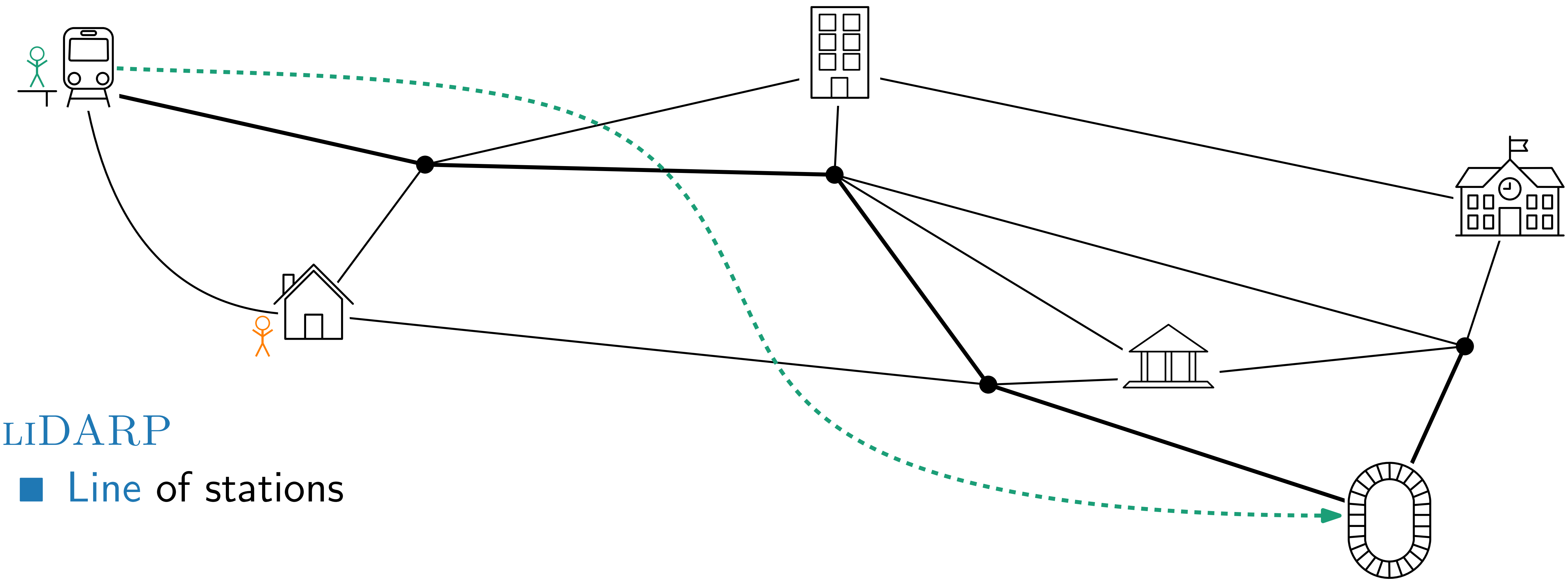
line-based Dial-a-Ride-Problem (liDARP)



liDARP

- Line of stations

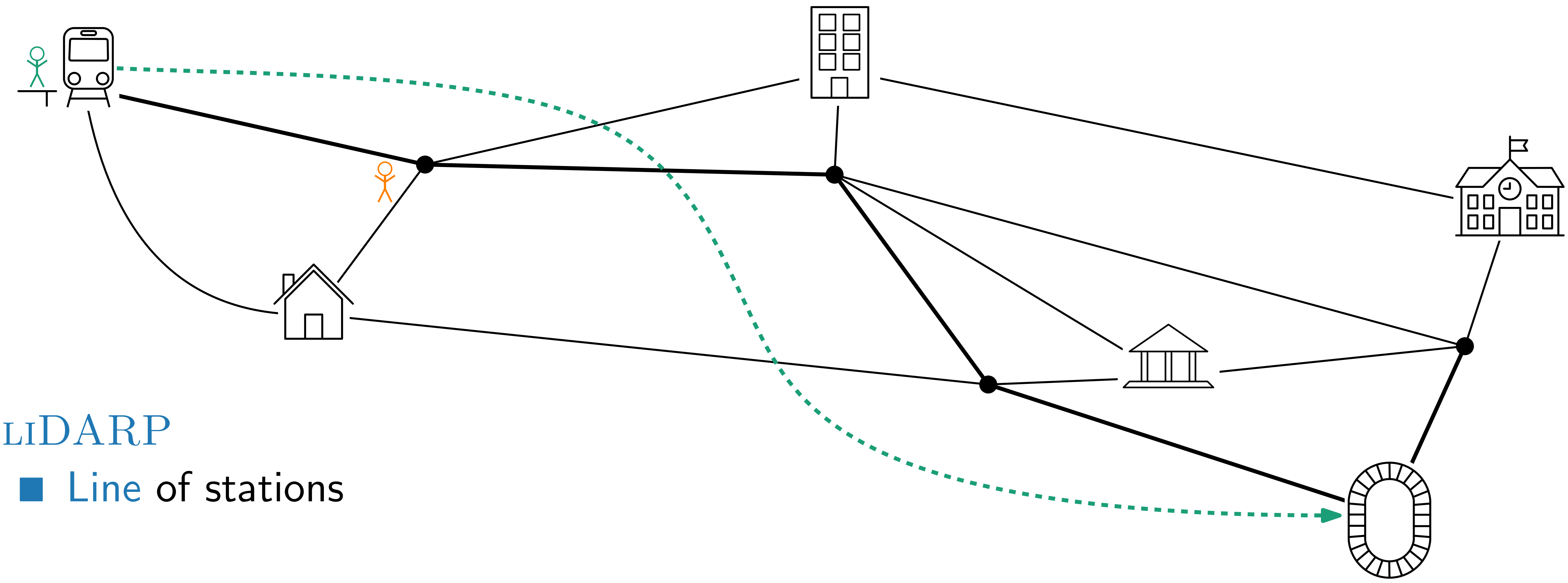
line-based Dial-a-Ride-Problem (liDARP)



liDARP

- Line of stations

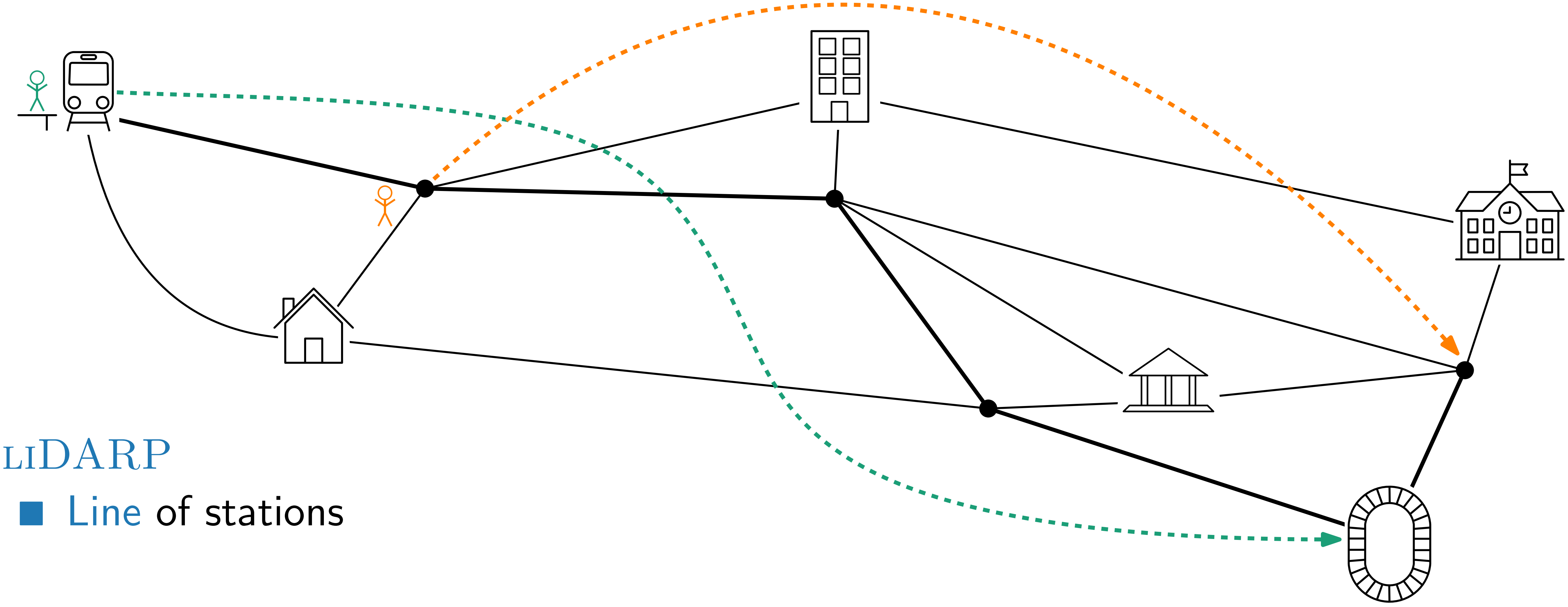
line-based Dial-a-Ride-Problem (liDARP)



liDARP

- **Line** of stations

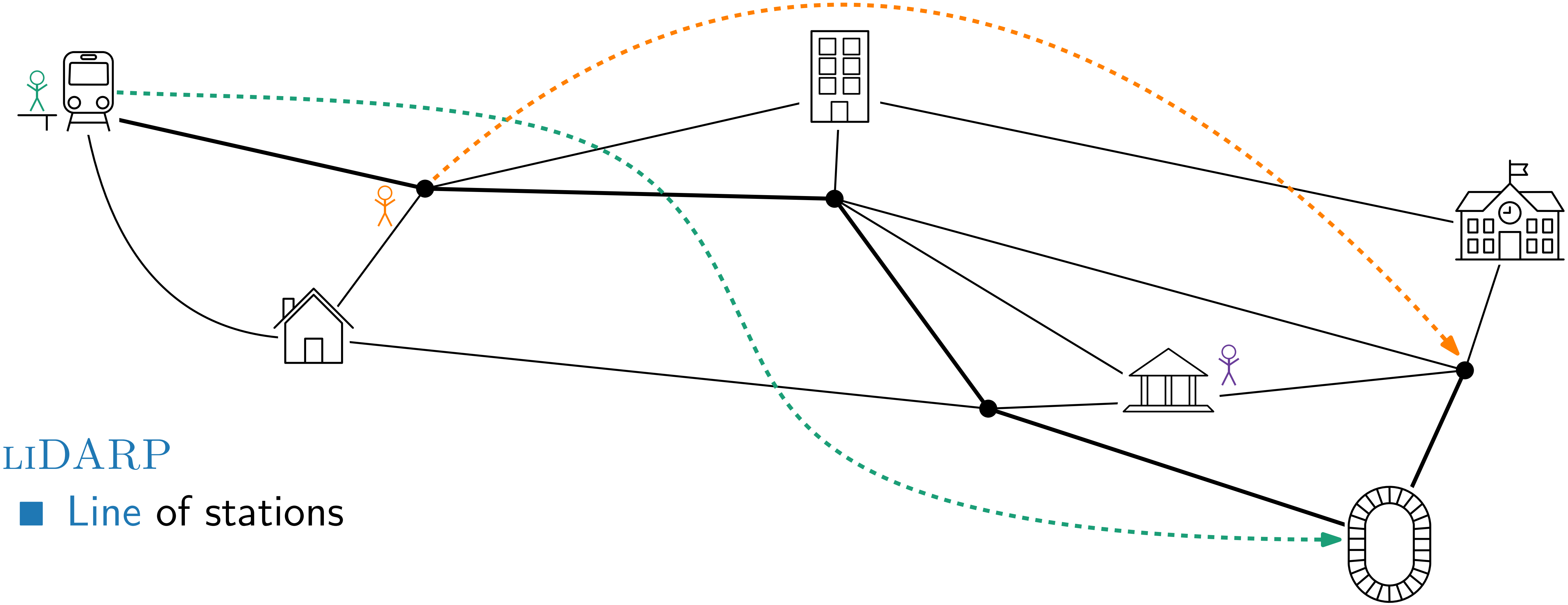
line-based Dial-a-Ride-Problem (liDARP)



liDARP

- Line of stations

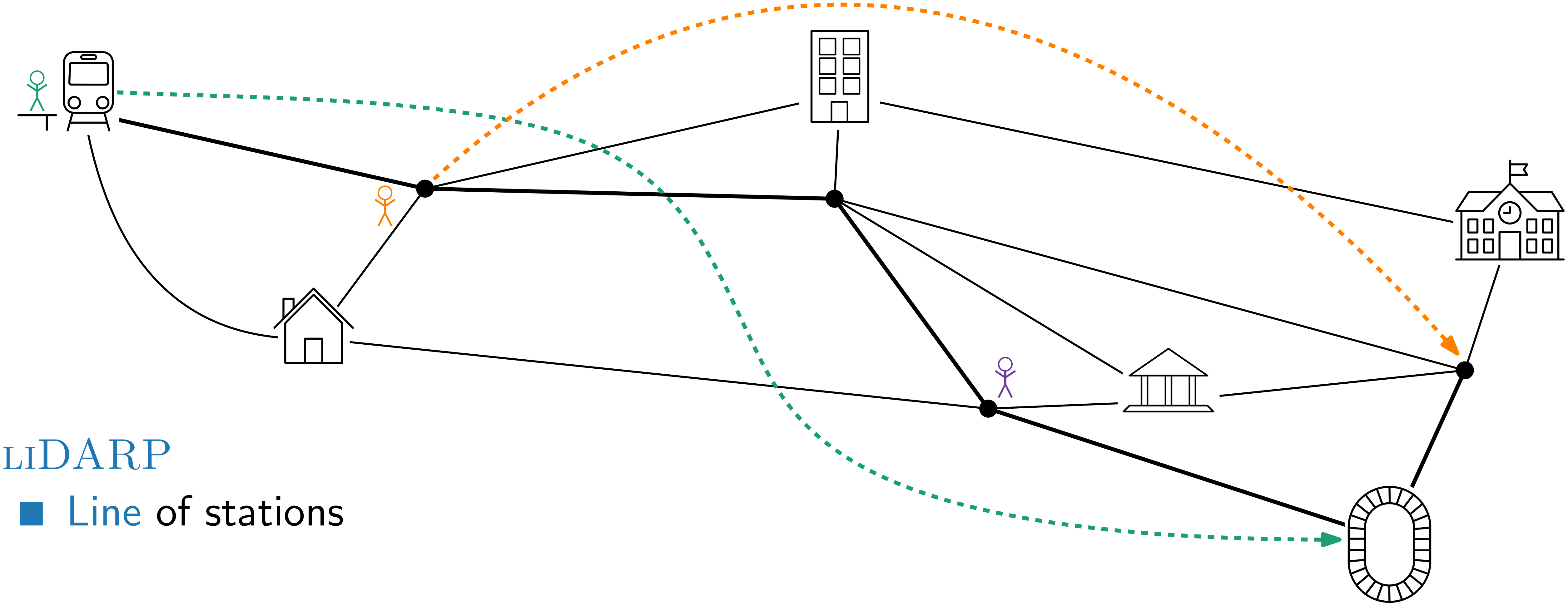
line-based Dial-a-Ride-Problem (liDARP)



liDARP

- Line of stations

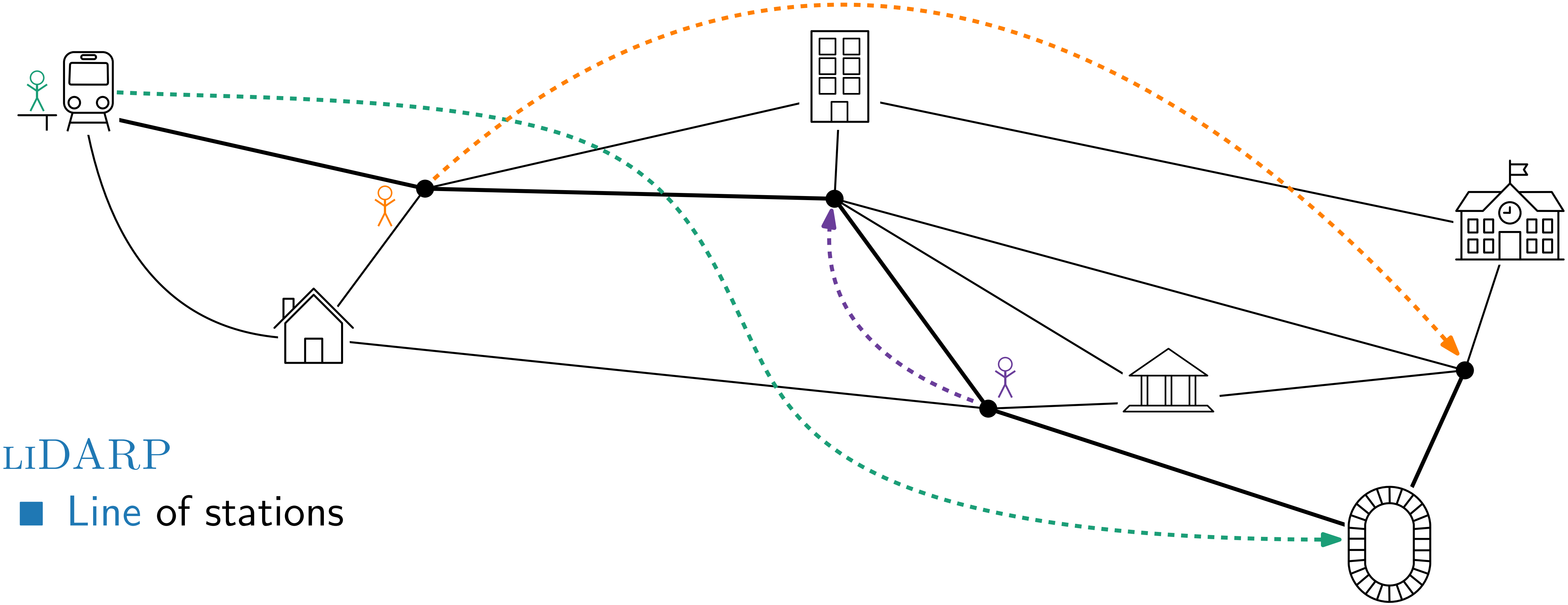
line-based Dial-a-Ride-Problem (liDARP)



liDARP

- Line of stations

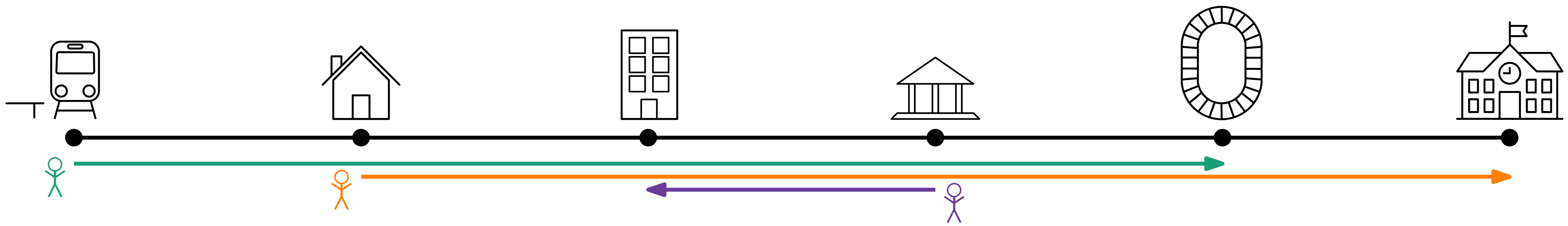
line-based Dial-a-Ride-Problem (liDARP)



liDARP

- Line of stations

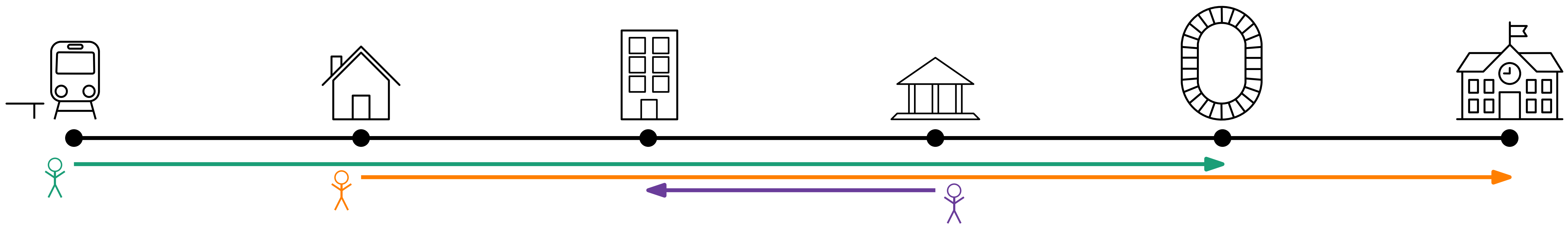
line-based Dial-a-Ride-Problem (liDARP)



liDARP

- Line of stations

line-based Dial-a-Ride-Problem (liDARP)



liDARP

- Line of stations
- Directionality: passengers always transported towards destination

line-based Dial-a-Ride-Problem (liDARP)

liDARP

- Line of stations
- Directionality: passengers always transported towards destination

line-based Dial-a-Ride-Problem (liDARP)

liDARP

- Line of stations
- Directionality: passengers always transported towards destination

line-based Dial-a-Ride-Problem (liDARP)

liDARP

- Line of stations
- Directionality: passengers always transported towards destination

line-based Dial-a-Ride-Problem (liDARP)

liDARP

- Line of stations
- Directionality: passengers always transported towards destination

line-based Dial-a-Ride-Problem (liDARP)

liDARP

- Line of stations
- Directionality: passengers always transported towards destination

line-based Dial-a-Ride-Problem (liDARP)

liDARP

- Line of stations
- Directionality: passengers always transported towards destination

line-based Dial-a-Ride-Problem (liDARP)

liDARP

- Line of stations
- Directionality: passengers always transported towards destination

line-based Dial-a-Ride-Problem (liDARP)

liDARP

- Line of stations
- Directionality: passengers always transported towards destination

LINE-BASED DIAL-A-RIDE-PROBLEM (LIDARP)

LIDARP

- Line of stations
- Directionality: passengers always transported towards destination

line-based Dial-a-Ride-Problem (liDARP)

liDARP

- Line of stations
- Directionality: passengers always transported towards destination
  - bus empty when changing direction

line-based Dial-a-Ride-Problem (liDARP)

liDARP

- Line of stations
- Directionality: passengers always transported towards destination
  - bus empty when changing direction

line-based Dial-a-Ride-Problem (liDARP)

liDARP

- Line of stations
- Directionality: passengers always transported towards destination
  - bus empty when changing direction

LINE-BASED DIAL-A-RIDE-PROBLEM (LIDARP)

LIDARP

- Line of stations
- Directionality: passengers always transported towards destination
  - bus empty when changing direction

line-based Dial-a-Ride-Problem (liDARP)

liDARP

- Line of stations
- Directionality: passengers always transported towards destination
  - bus empty when changing direction
  - passengers only dropped-off at their destination

line-based Dial-a-Ride-Problem (liDARP)

liDARP

- Line of stations
- Directionality: passengers always transported towards destination
  - bus empty when changing direction
  - passengers only dropped-off at their destination

line-based Dial-a-Ride-Problem (liDARP)

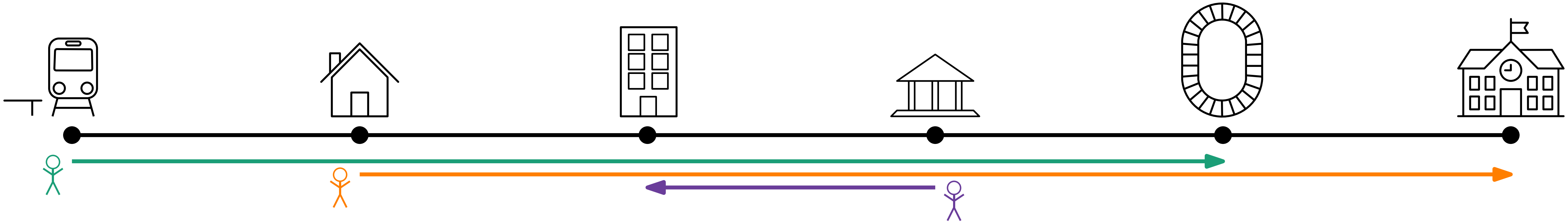
Variants

line-based Dial-a-Ride-Problem (liDARP)

Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p

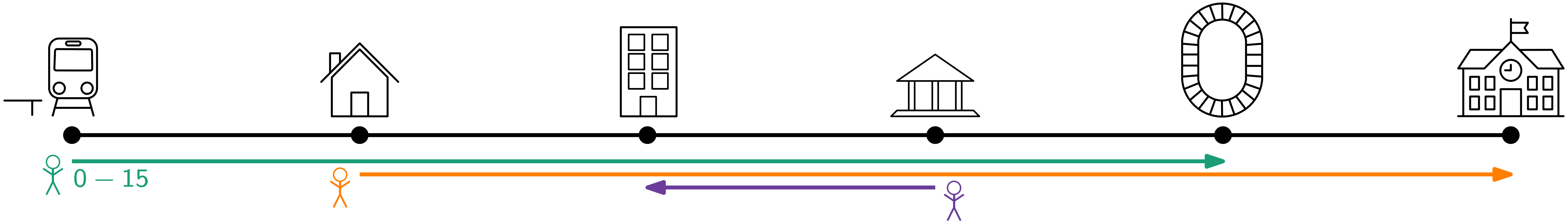
line-based Dial-a-Ride-Problem (liDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p

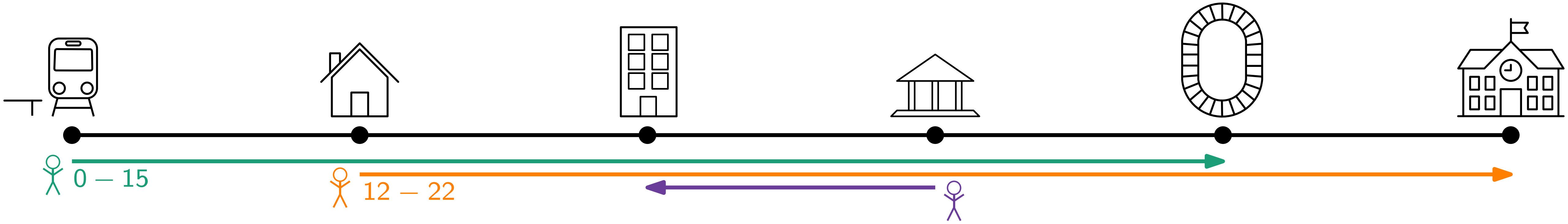
line-based Dial-a-Ride-Problem (liDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p

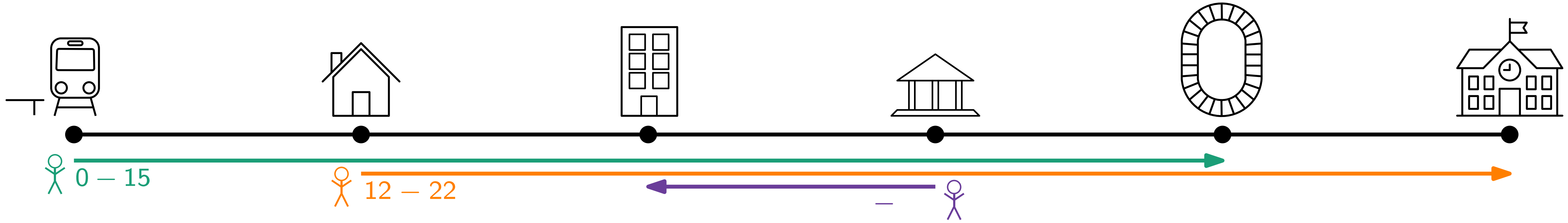
line-based Dial-a-Ride-Problem (liDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p

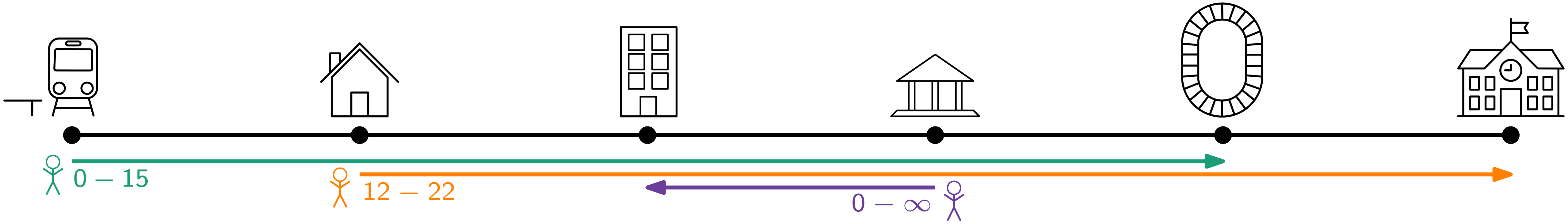
line-based Dial-a-Ride-Problem (liDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p

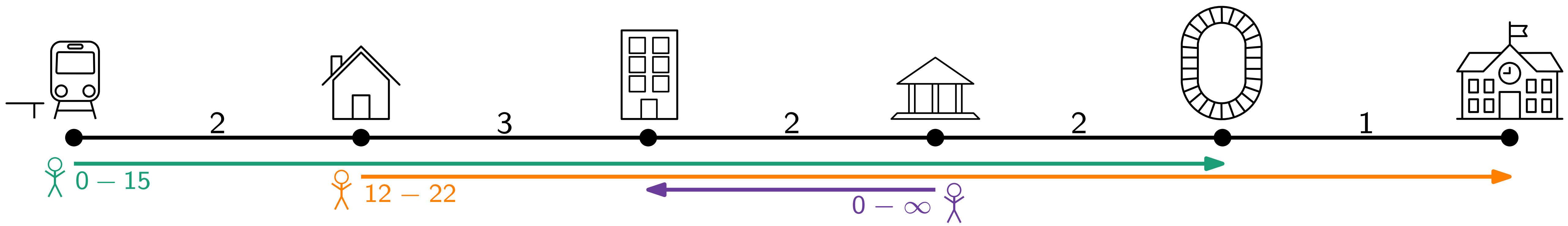
LINE-BASED DIAL-A-RIDE-PROBLEM (LIDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p

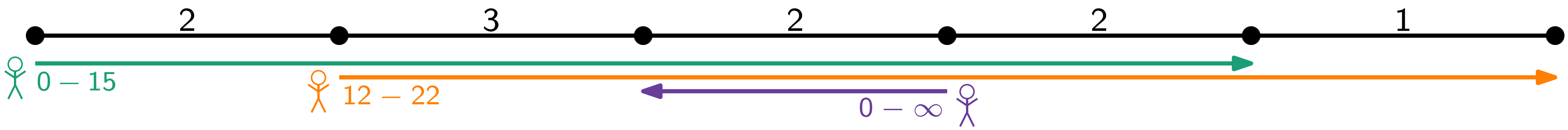
line-based Dial-a-Ride-Problem (liDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p

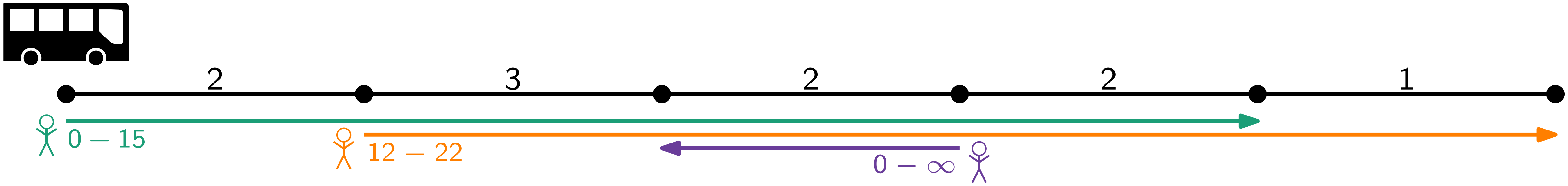
line-based Dial-a-Ride-Problem (liDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p

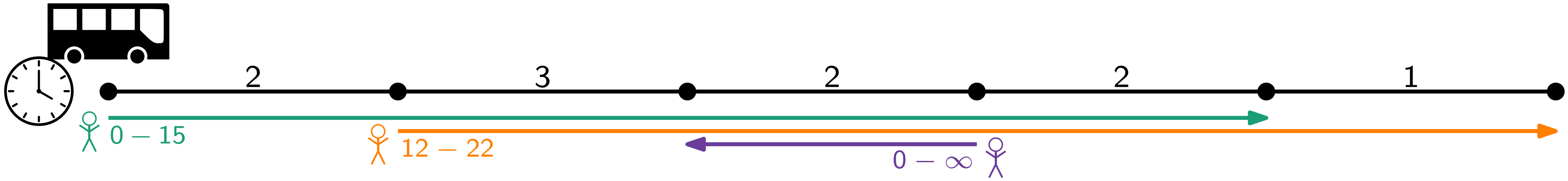
line-based Dial-a-Ride-Problem (liDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p

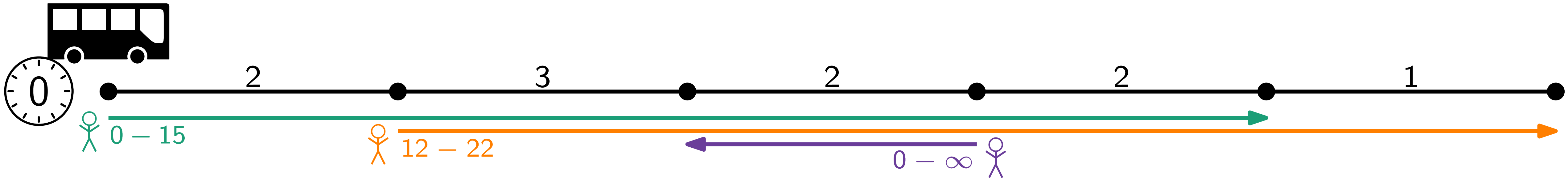
LINE-BASED DIAL-A-RIDE-PROBLEM (LIDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p

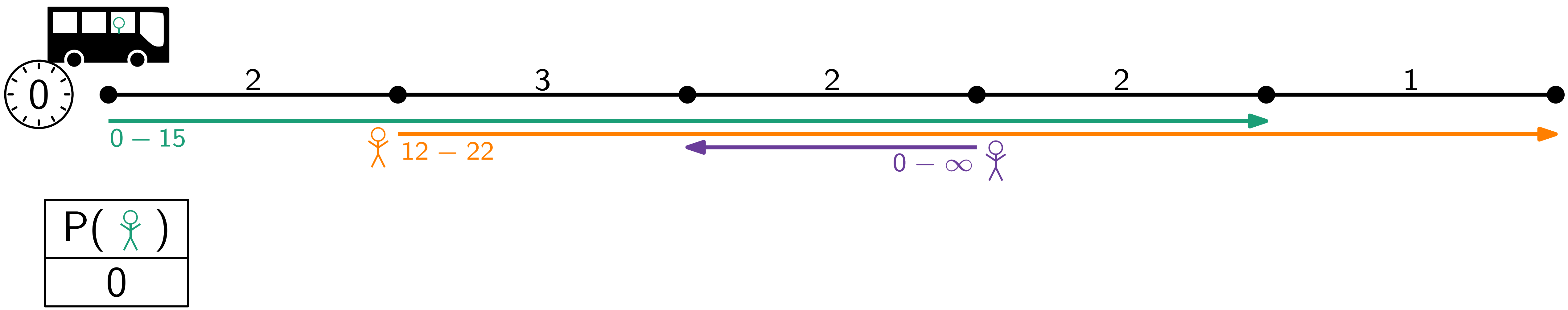
line-based Dial-a-Ride-Problem (liDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p

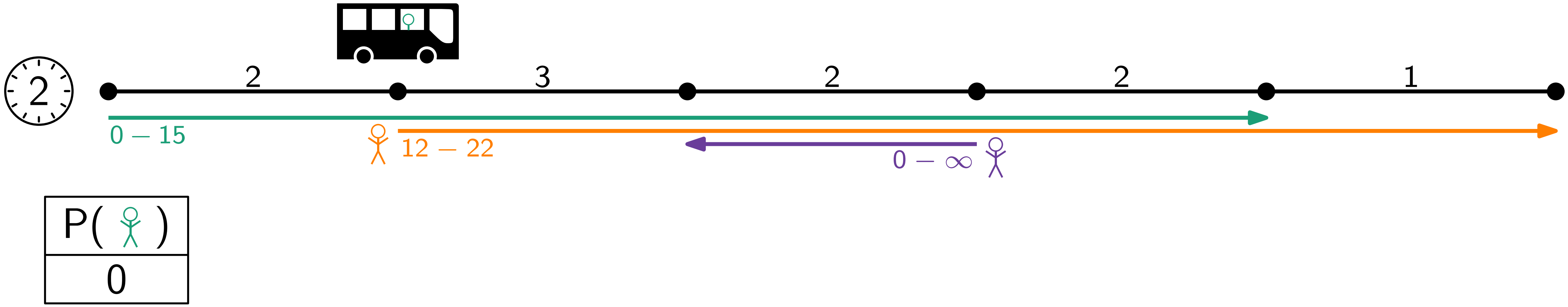
line-based Dial-a-Ride-Problem (liDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p

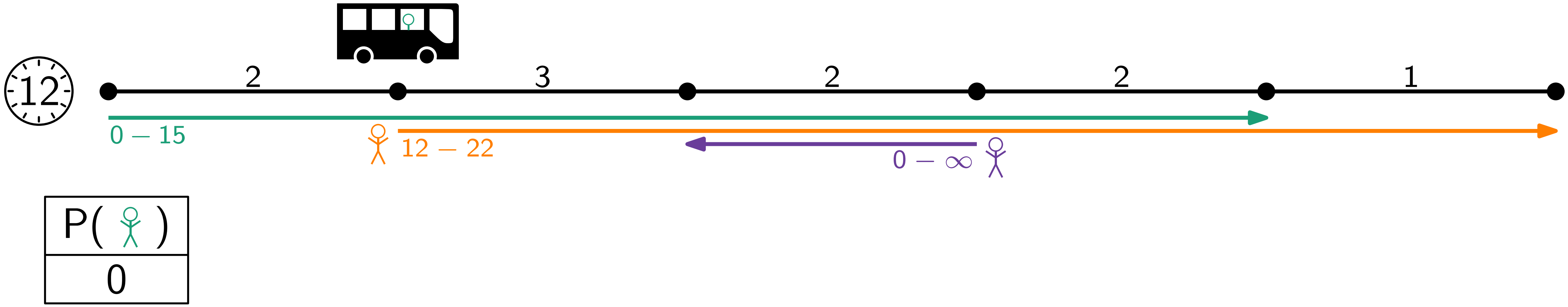
LINE-BASED DIAL-A-RIDE-PROBLEM (LIDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p

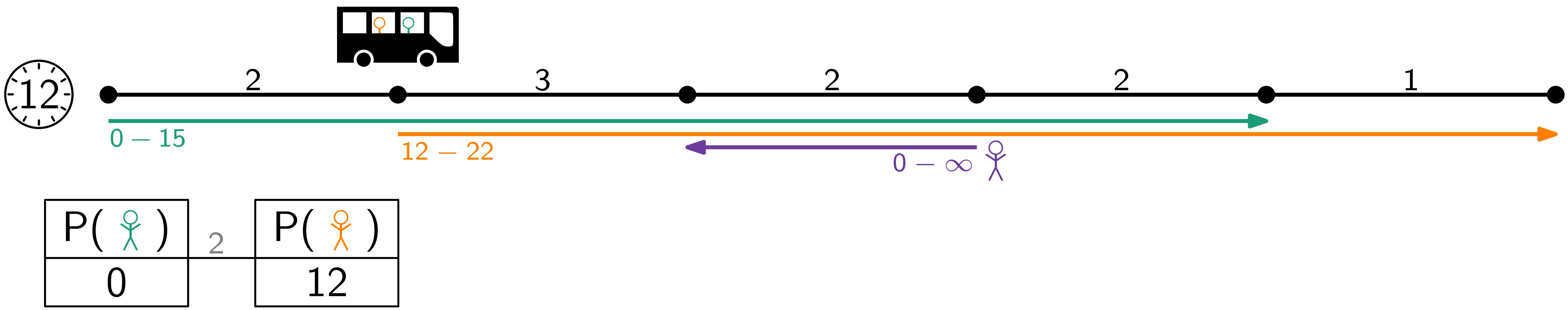
line-based Dial-a-Ride-Problem (liDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p

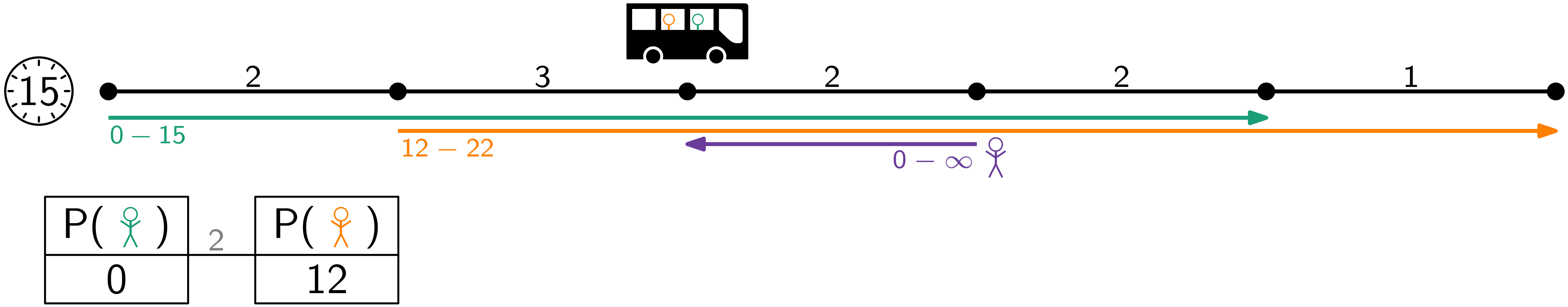
LINE-BASED DIAL-A-RIDE-PROBLEM (LIDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p

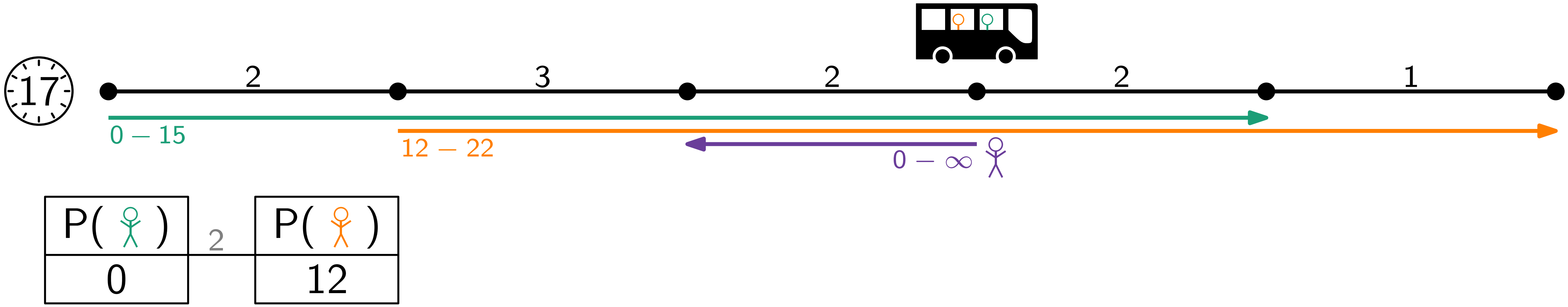
line-based Dial-a-Ride-Problem (liDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p

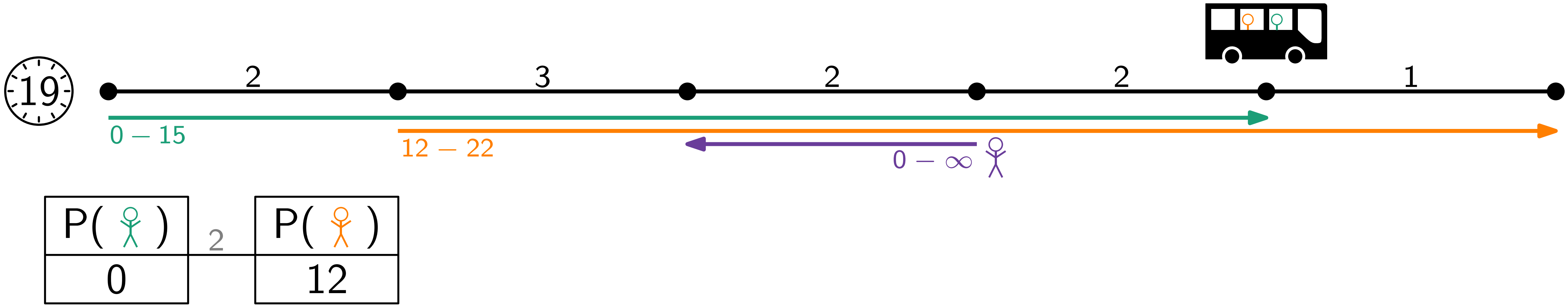
line-based Dial-a-Ride-Problem (liDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p

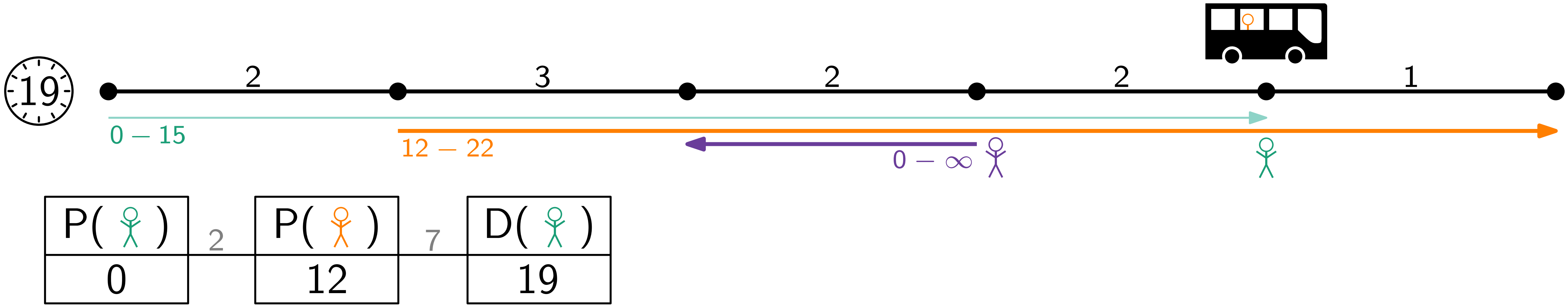
line-based Dial-a-Ride-Problem (liDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p

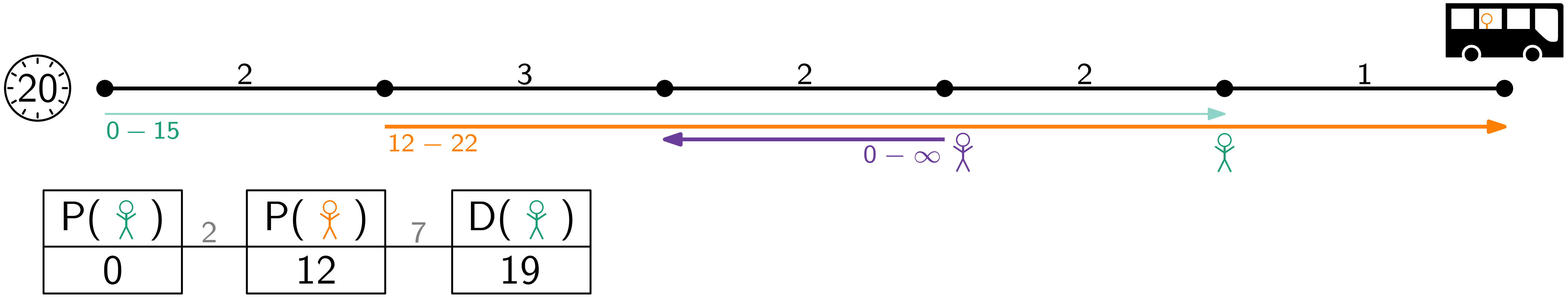
LINE-BASED DIAL-A-RIDE-PROBLEM (LIDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p

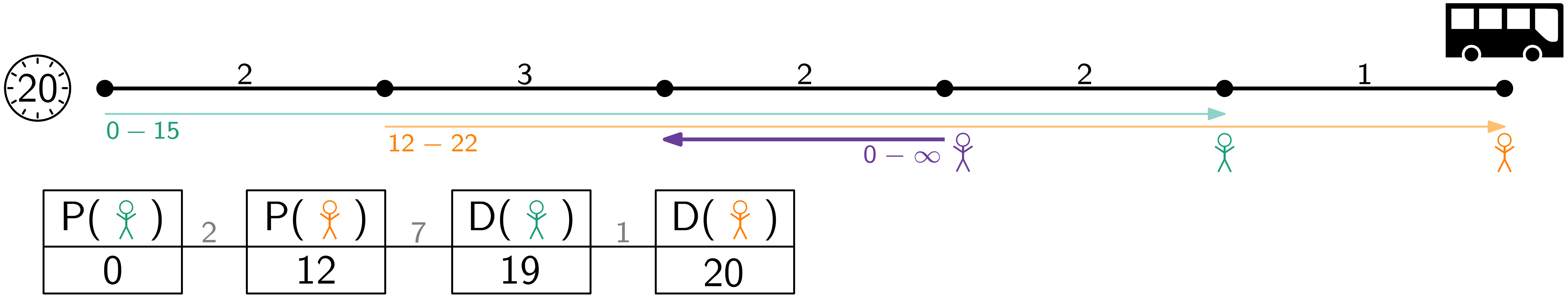
LINE-BASED DIAL-A-RIDE-PROBLEM (LIDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p

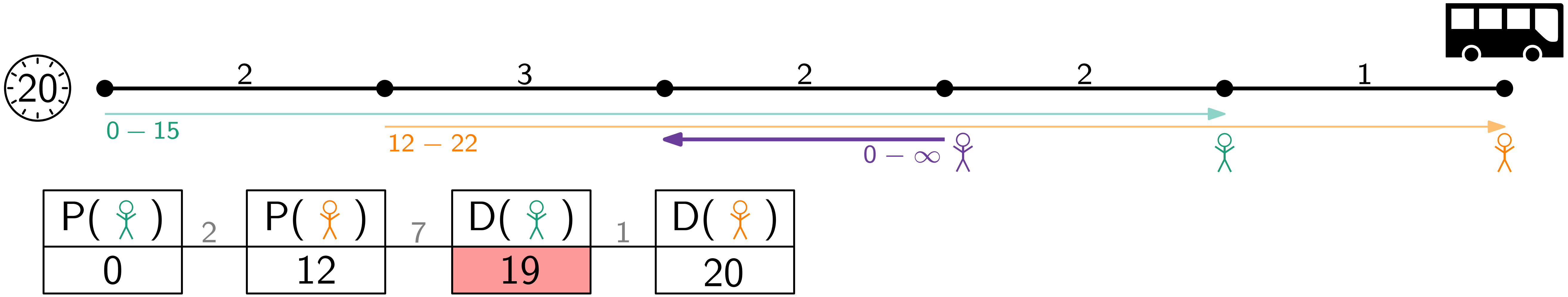
LINE-BASED DIAL-A-RIDE-PROBLEM (liDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p

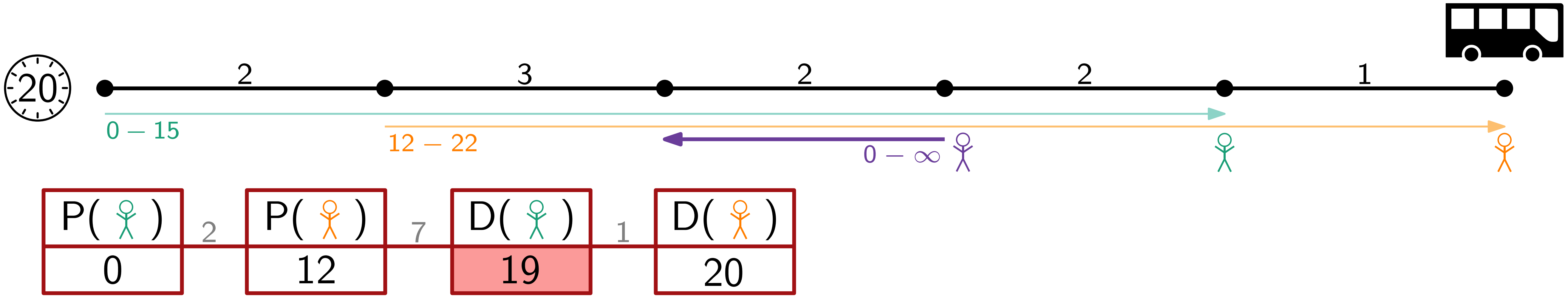
line-based Dial-a-Ride-Problem (liDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p

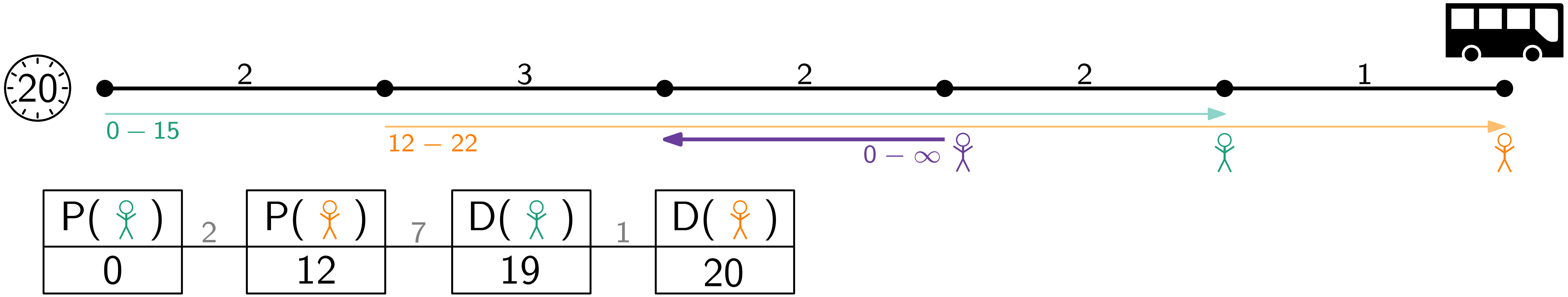
LINE-BASED DIAL-A-RIDE-PROBLEM (LIDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p

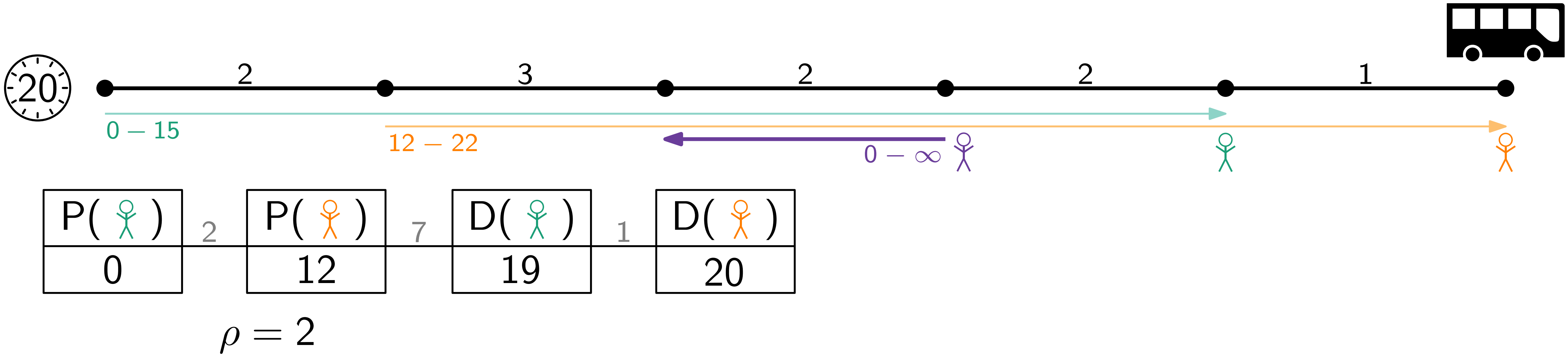
LINE-BASED DIAL-A-RIDE-PROBLEM (LIDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p
- Service Promise (SP): passenger at most ρ times longer in bus than in direct route

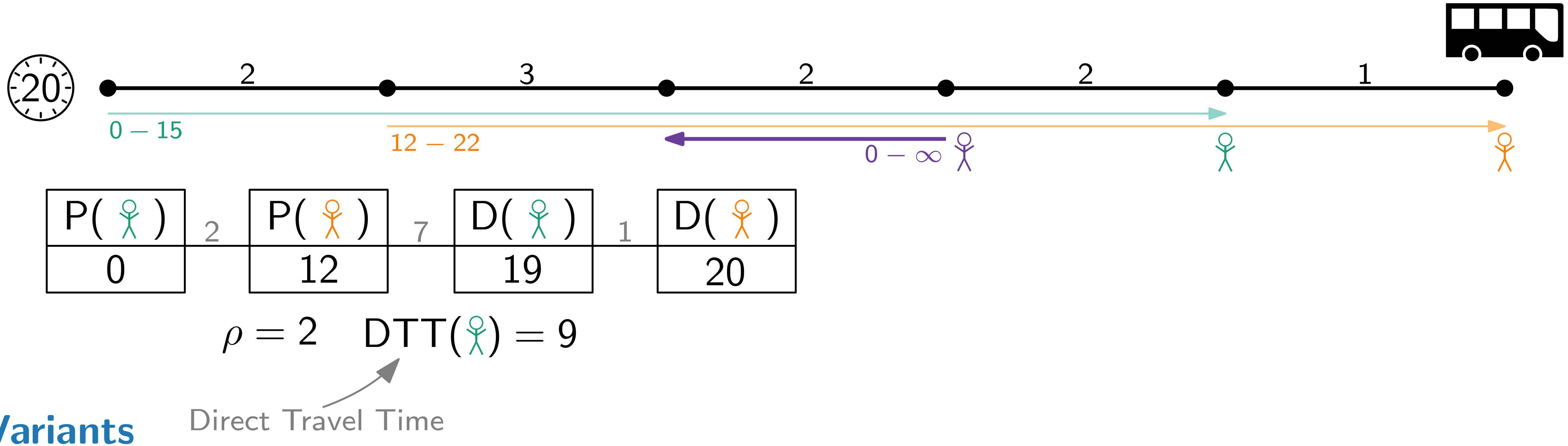
LINE-BASED DIAL-A-RIDE-PROBLEM (LIDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p
- Service Promise (SP): passenger at most ρ times longer in bus than in direct route

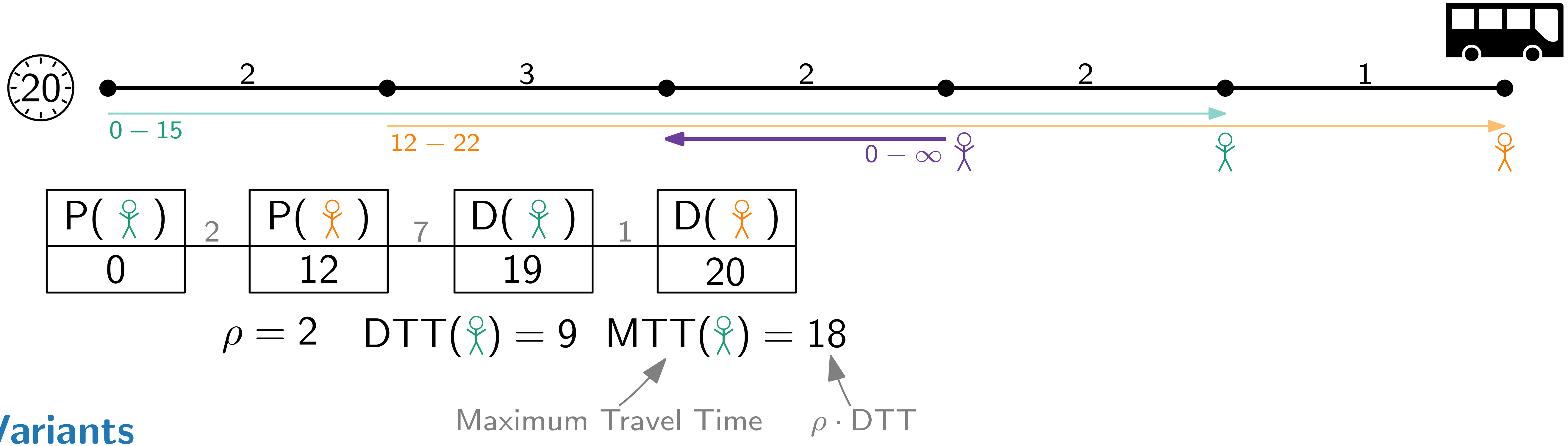
LINE-BASED DIAL-A-RIDE-PROBLEM (LIDARP)



Variants

- **Time Windows (TW):** earliest pick-up e_p and latest drop-off ℓ_p for request p
- **Service Promise (SP):** passenger at most ρ times longer in bus than in direct route

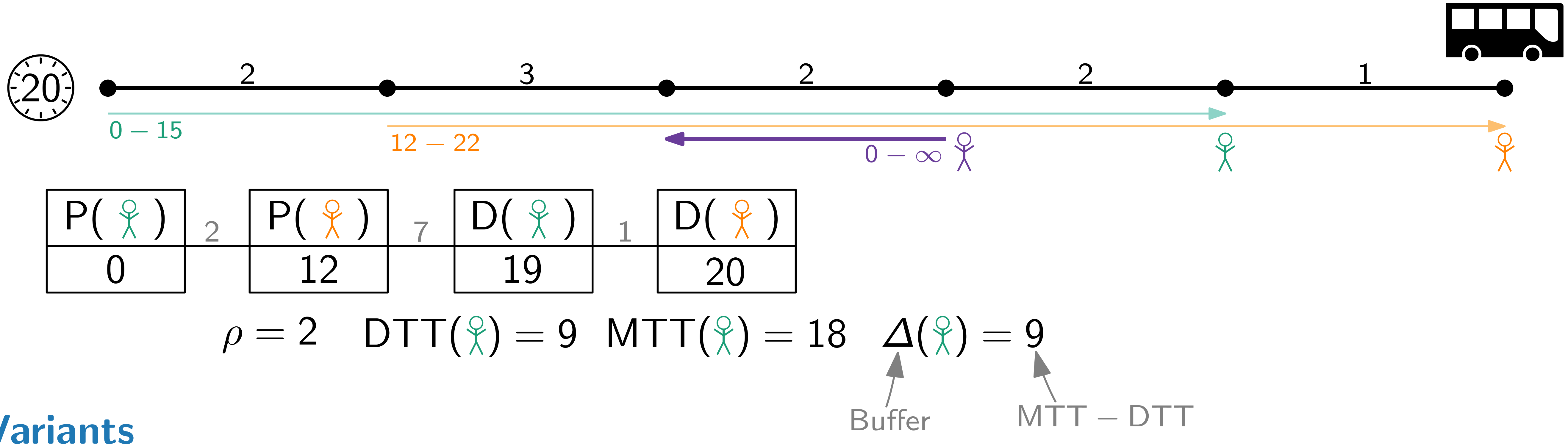
LINE-BASED DIAL-A-RIDE-PROBLEM (LIDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p
- Service Promise (SP): passenger at most ρ times longer in bus than in direct route

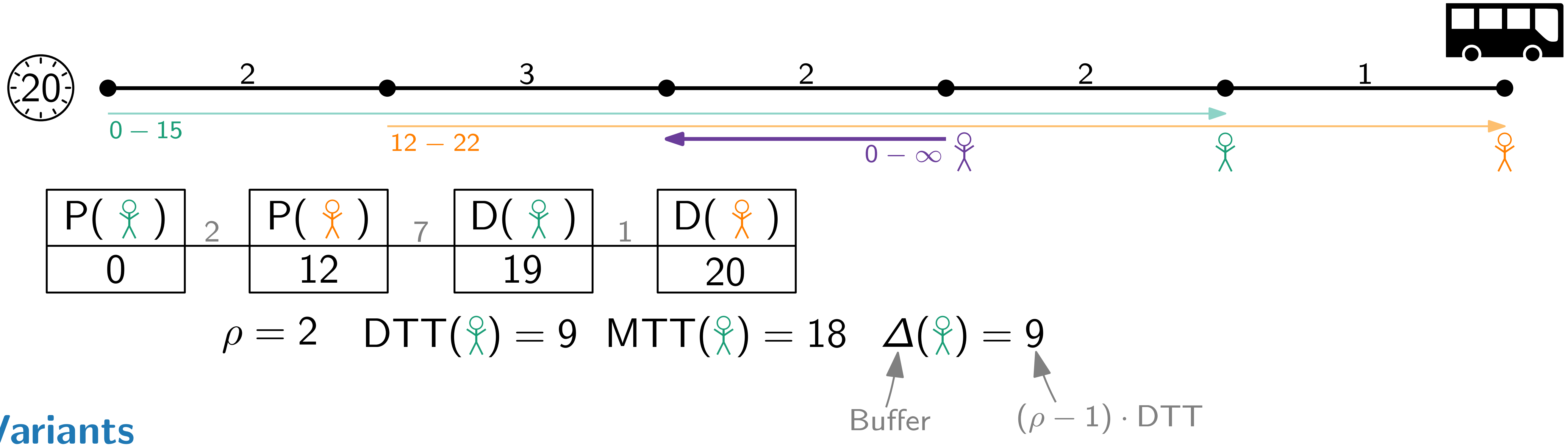
line-based Dial-a-Ride-Problem (liDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p
- Service Promise (SP): passenger at most ρ times longer in bus than in direct route

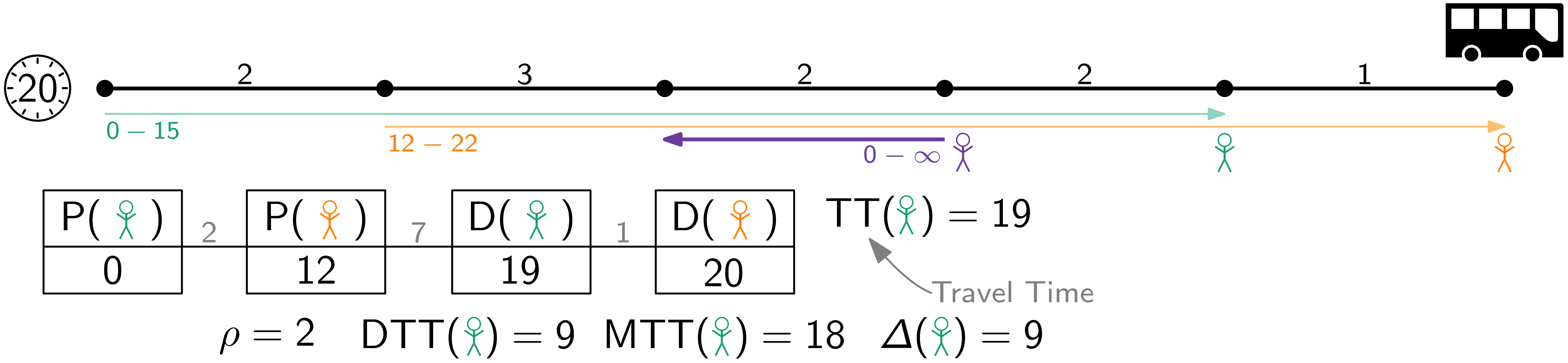
line-based Dial-a-Ride-Problem (liDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p
- Service Promise (SP): passenger at most ρ times longer in bus than in direct route

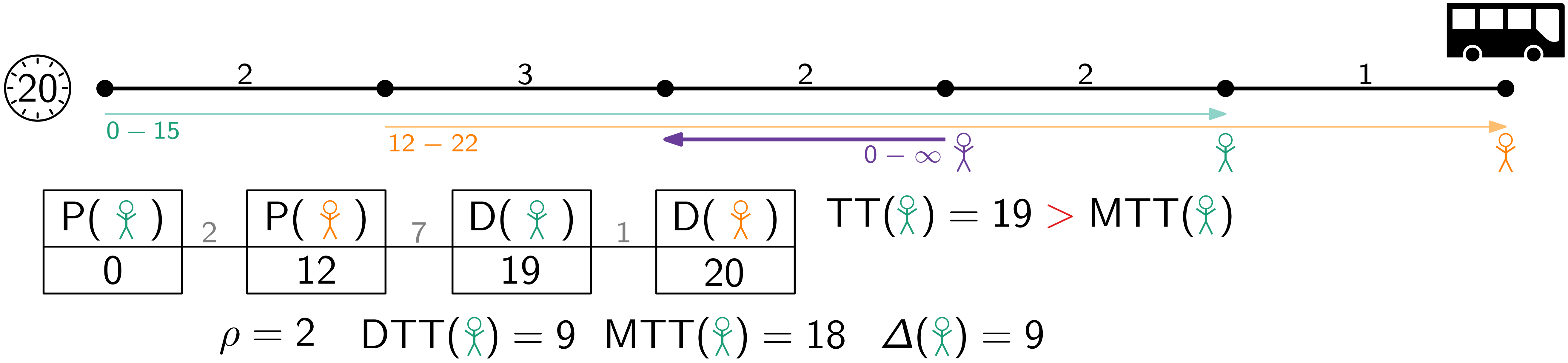
LINE-BASED DIAL-A-RIDE-PROBLEM (LIDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p
- Service Promise (SP): passenger at most ρ times longer in bus than in direct route

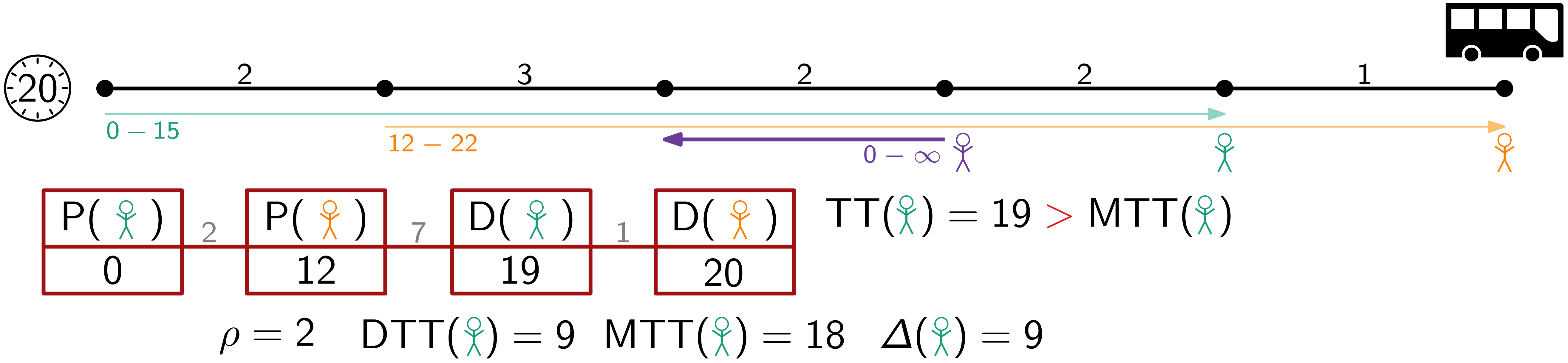
line-based Dial-a-Ride-Problem (liDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p
- Service Promise (SP): passenger at most ρ times longer in bus than in direct route

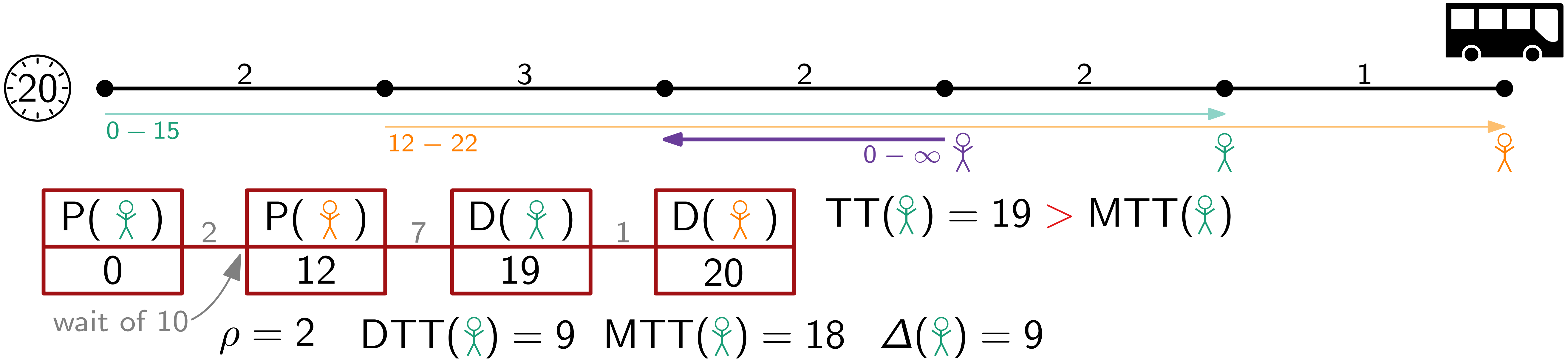
LINE-BASED DIAL-A-RIDE-PROBLEM (LIDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p
- Service Promise (SP): passenger at most ρ times longer in bus than in direct route

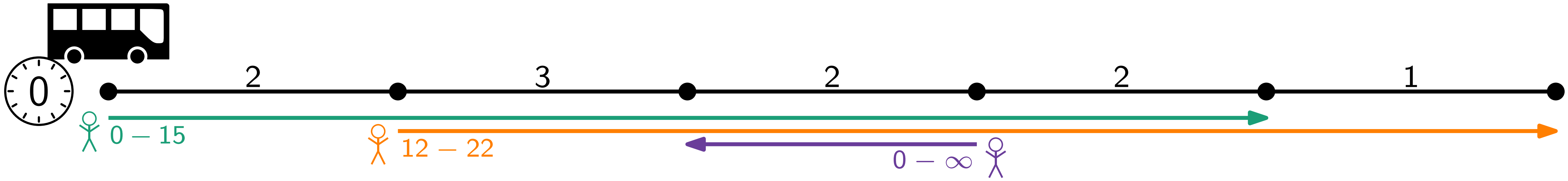
LINE-BASED DIAL-A-RIDE-PROBLEM (LIDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p
- Service Promise (SP): passenger at most ρ times longer in bus than in direct route

LINE-BASED DIAL-A-RIDE-PROBLEM (LIDARP)

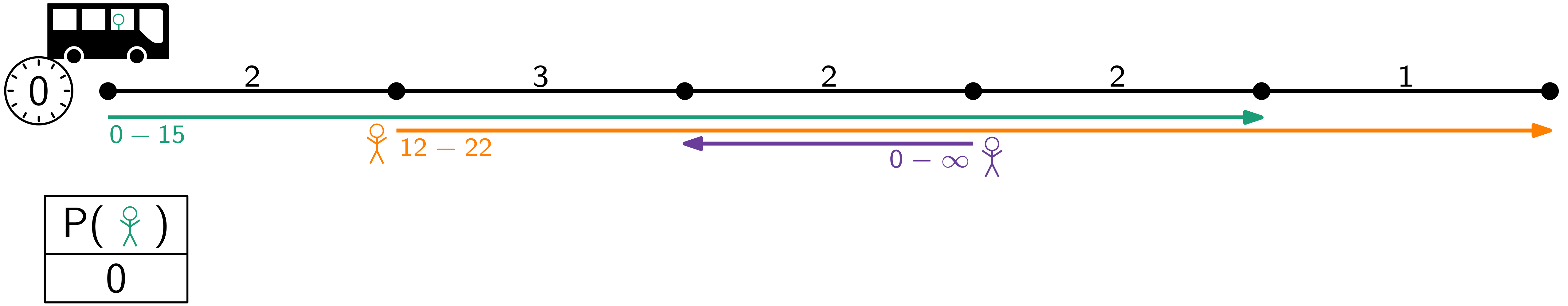


$\rho = 2 \quad \mathrm{DTT}(\;) = 9 \quad \mathrm{MTT}(\;) = 18 \quad \Delta(\;) = 9$

Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p
- Service Promise (SP): passenger at most ρ times longer in bus than in direct route

LINE-BASED DIAL-A-RIDE-PROBLEM (liDARP)

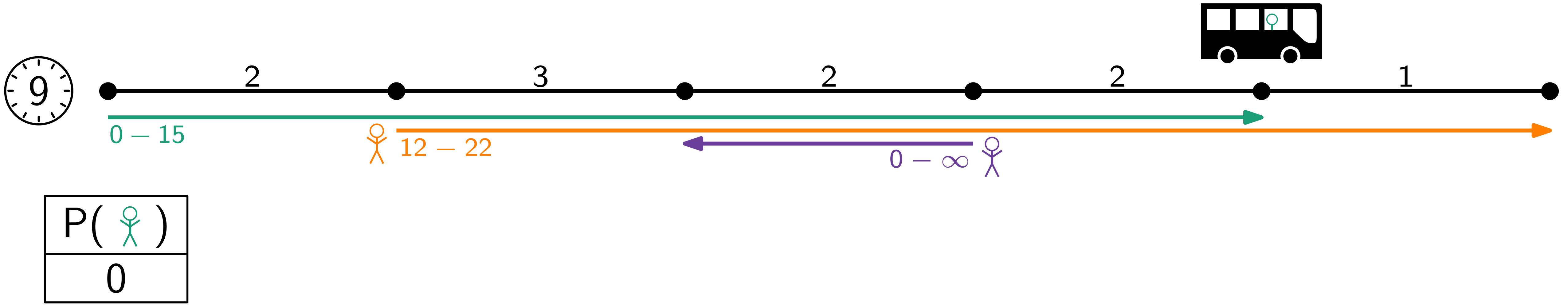


$\rho = 2$ DTT( ) $= 9$ MTT( ) $= 18$ Δ( ) $= 9$

Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p
- Service Promise (SP): passenger at most ρ times longer in bus than in direct route

line-based Dial-a-Ride-Problem (liDARP)

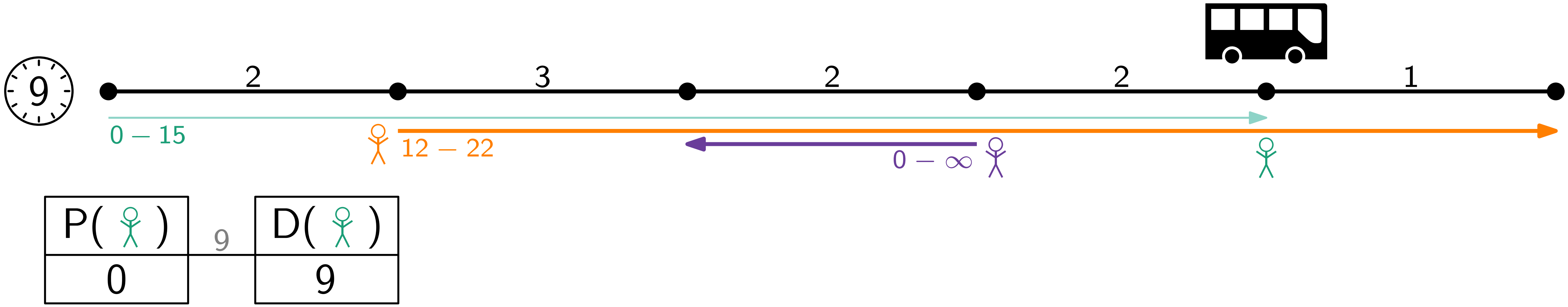


$\rho = 2$ DTT( ) $= 9$ MTT( ) $= 18$ Δ( ) $= 9$

Variants

- **Time Windows (TW):** earliest pick-up e_p and latest drop-off ℓ_p for request p
- **Service Promise (SP):** passenger at most ρ times longer in bus than in direct route

line-based Dial-a-Ride-Problem (liDARP)

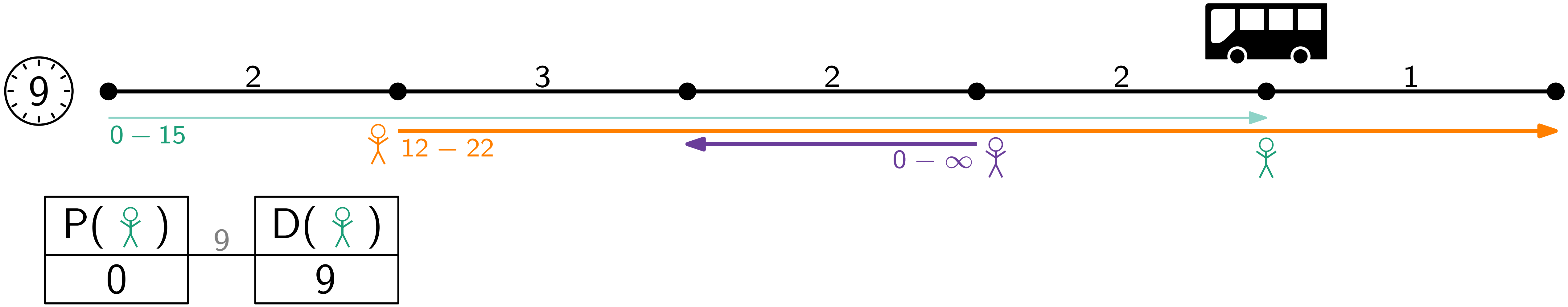


$\rho = 2$ DTT( ) $= 9$ MTT( ) $= 18$ Δ( ) $= 9$

Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p
- Service Promise (SP): passenger at most ρ times longer in bus than in direct route

line-based Dial-a-Ride-Problem (liDARP)

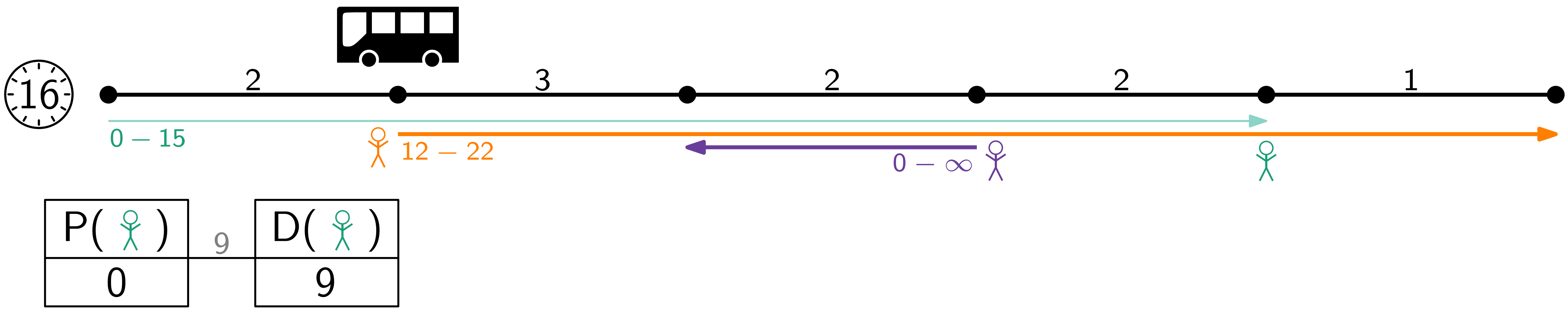


$\rho = 2$ DTT( ) $= 9$ MTT( ) $= 18$ Δ( ) $= 9$

Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p
- Service Promise (SP): passenger at most ρ times longer in bus than in direct route

line-based Dial-a-Ride-Problem (liDARP)

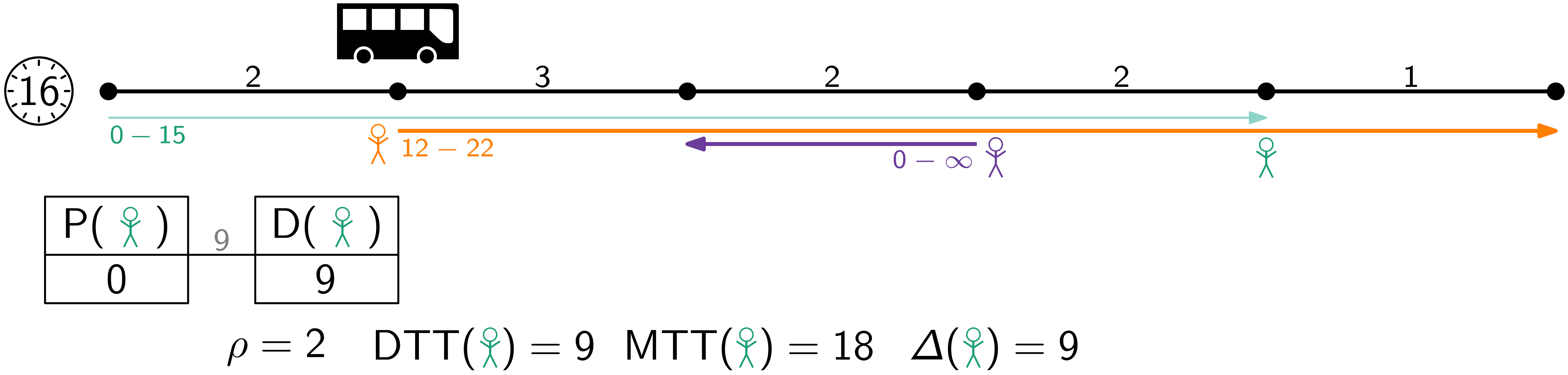


$\rho = 2$ DTT( ) $= 9$ MTT( ) $= 18$ Δ( ) $= 9$

Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p
- Service Promise (SP): passenger at most ρ times longer in bus than in direct route

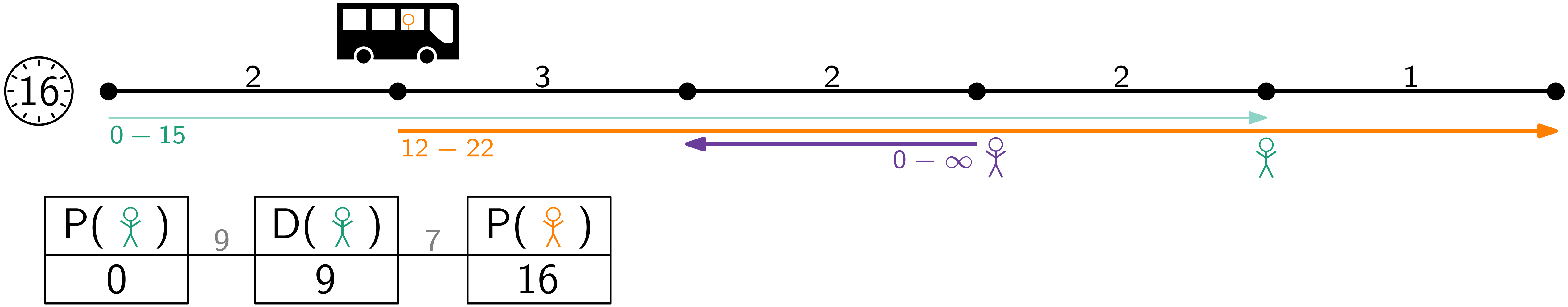
LINE-BASED DIAL-A-RIDE-PROBLEM (LIDARP)



Variants

- **Time Windows (TW):** earliest pick-up e_p and latest drop-off ℓ_p for request p
- **Service Promise (SP):** passenger at most ρ times longer in bus than in direct route

line-based Dial-a-Ride-Problem (liDARP)

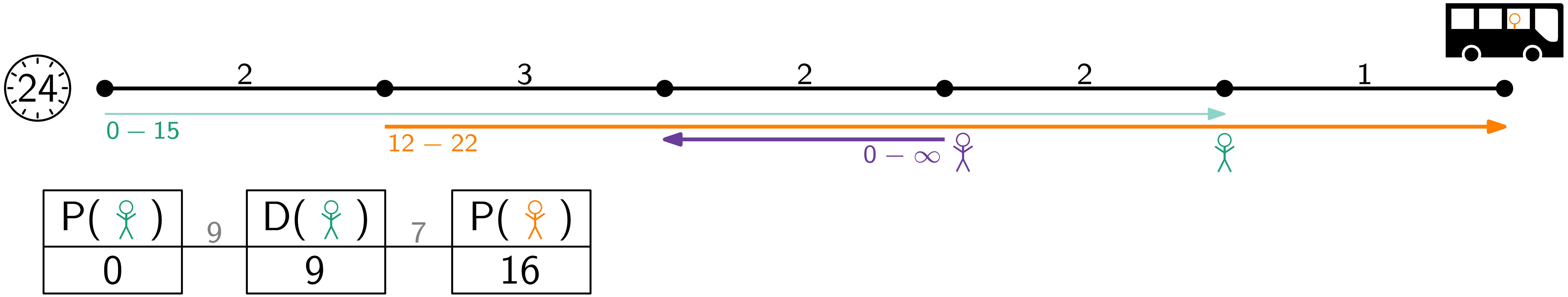


$\rho = 2$ DTT( ) $= 9$ MTT( ) $= 18$ Δ( ) $= 9$

Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p
- Service Promise (SP): passenger at most ρ times longer in bus than in direct route

line-based Dial-a-Ride-Problem (liDARP)

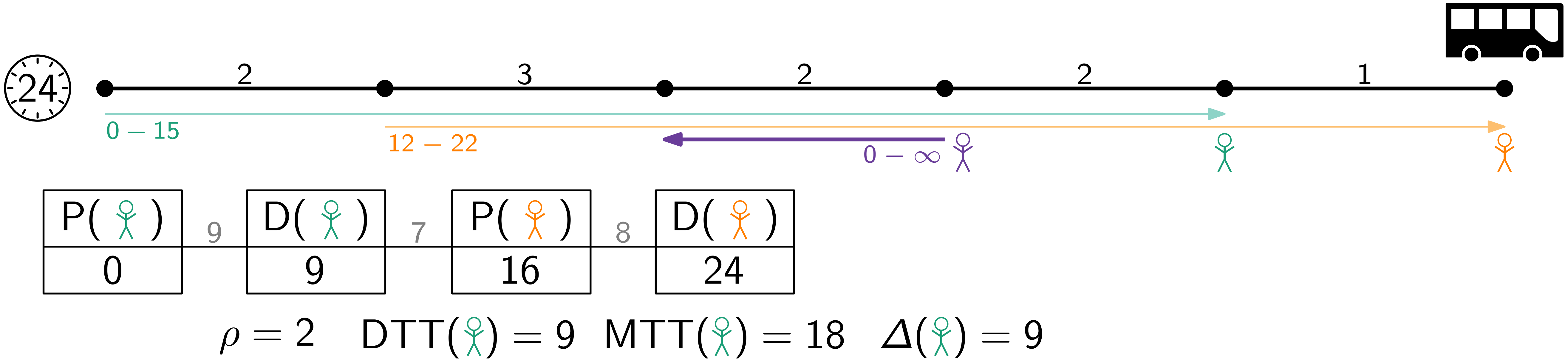


$\rho = 2$ DTT( ) $= 9$ MTT( ) $= 18$ Δ( ) $= 9$

Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p
- Service Promise (SP): passenger at most ρ times longer in bus than in direct route

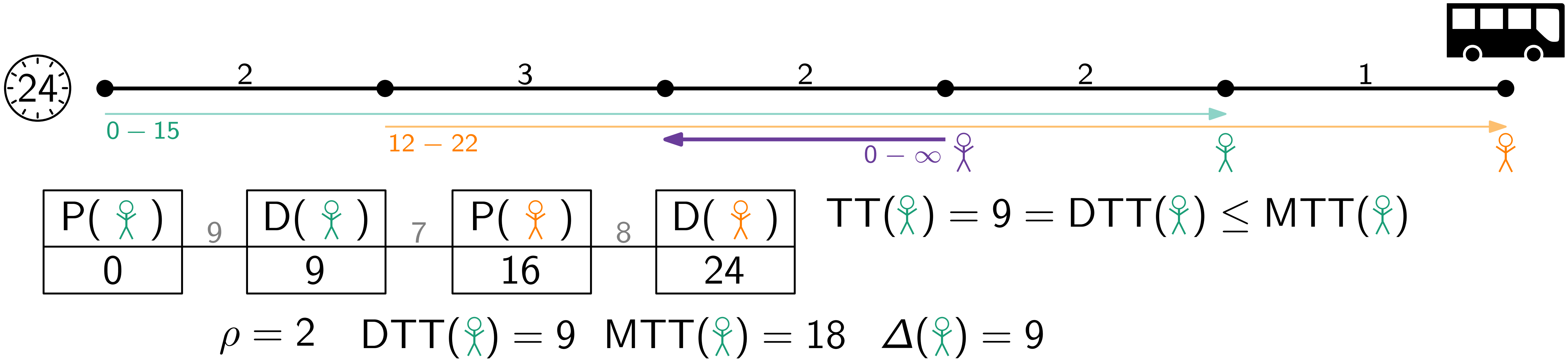
LINE-BASED DIAL-A-RIDE-PROBLEM (liDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p
- Service Promise (SP): passenger at most ρ times longer in bus than in direct route

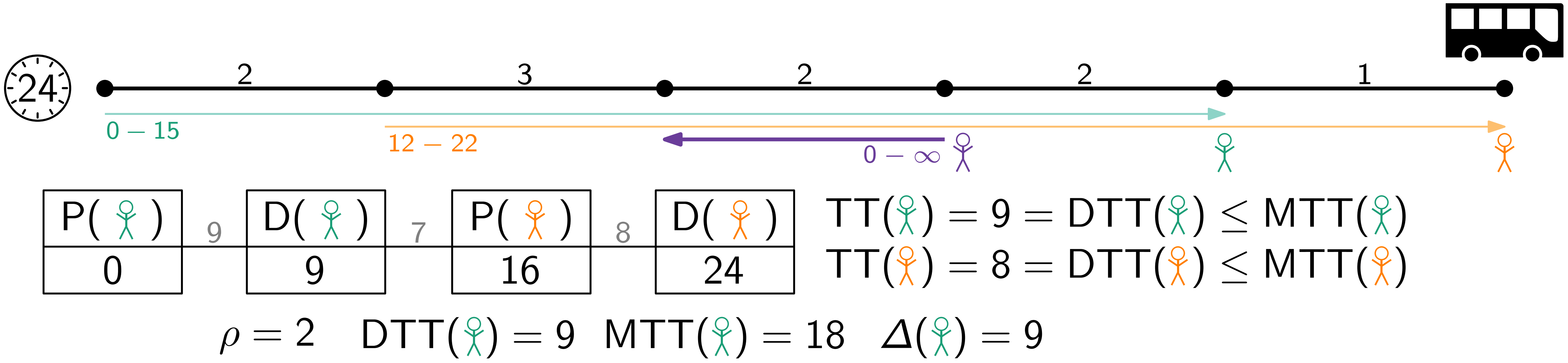
line-based Dial-a-Ride-Problem (liDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p
- Service Promise (SP): passenger at most ρ times longer in bus than in direct route

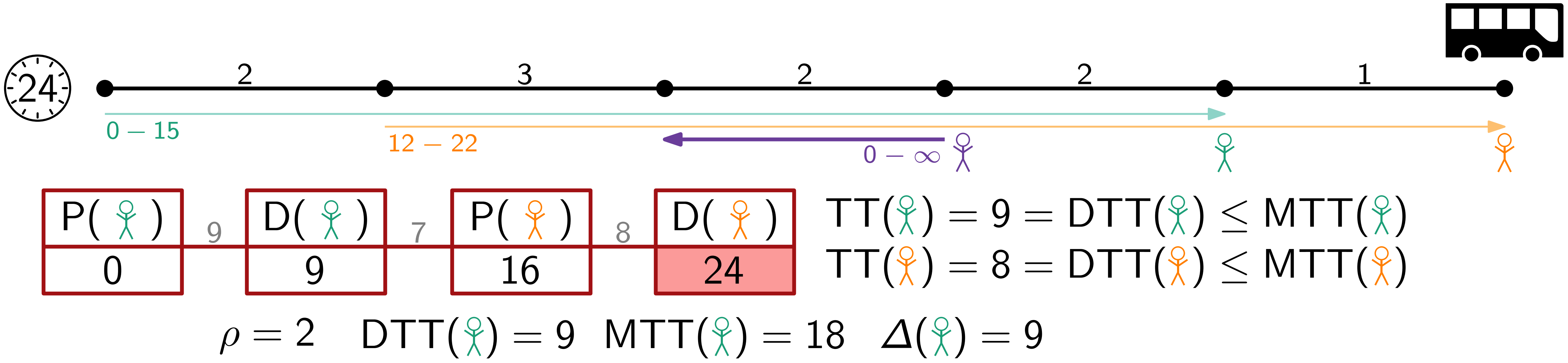
line-based Dial-a-Ride-Problem (liDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p
- Service Promise (SP): passenger at most ρ times longer in bus than in direct route

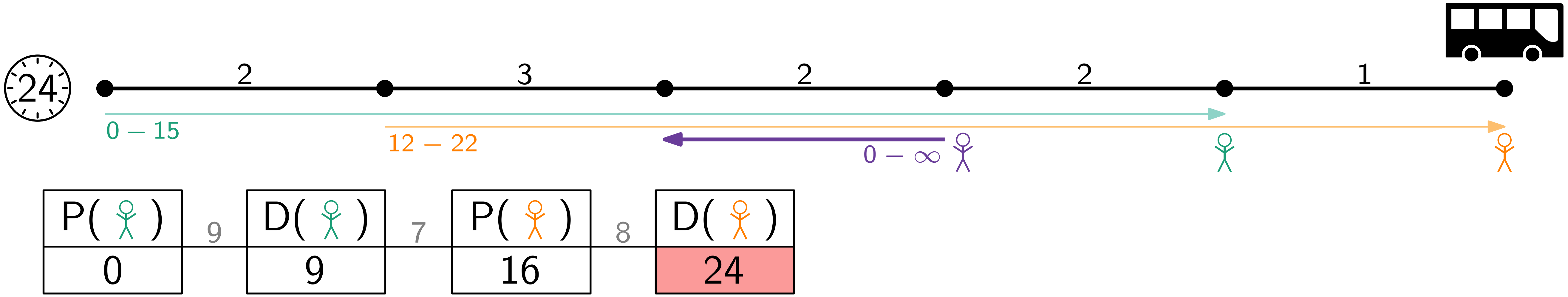
LINE-BASED DIAL-A-RIDE-PROBLEM (LIDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p
- Service Promise (SP): passenger at most ρ times longer in bus than in direct route

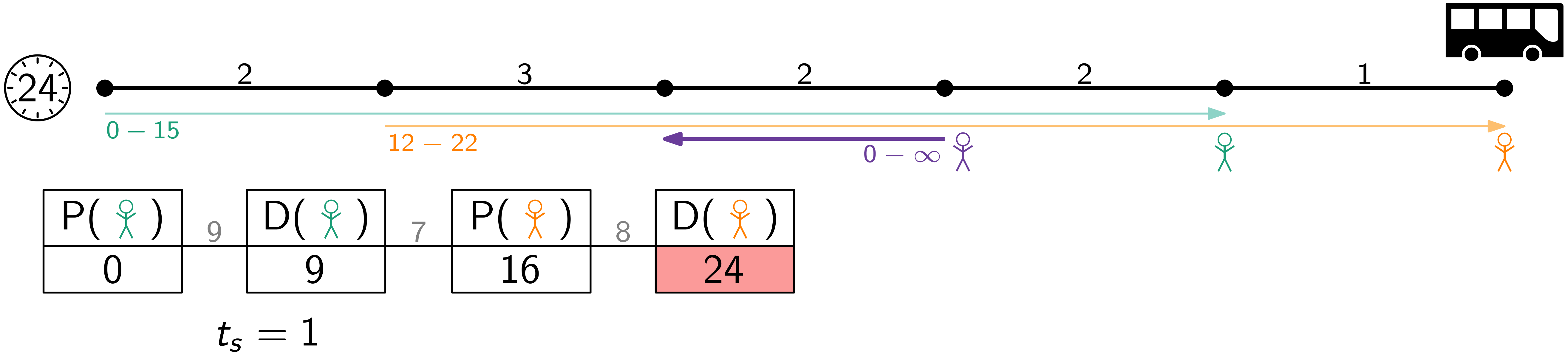
line-based Dial-a-Ride-Problem (liDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p
- Service Promise (SP): passenger at most ρ times longer in bus than in direct route
- Service Time (ST): pick-up / drop-off takes time t_s

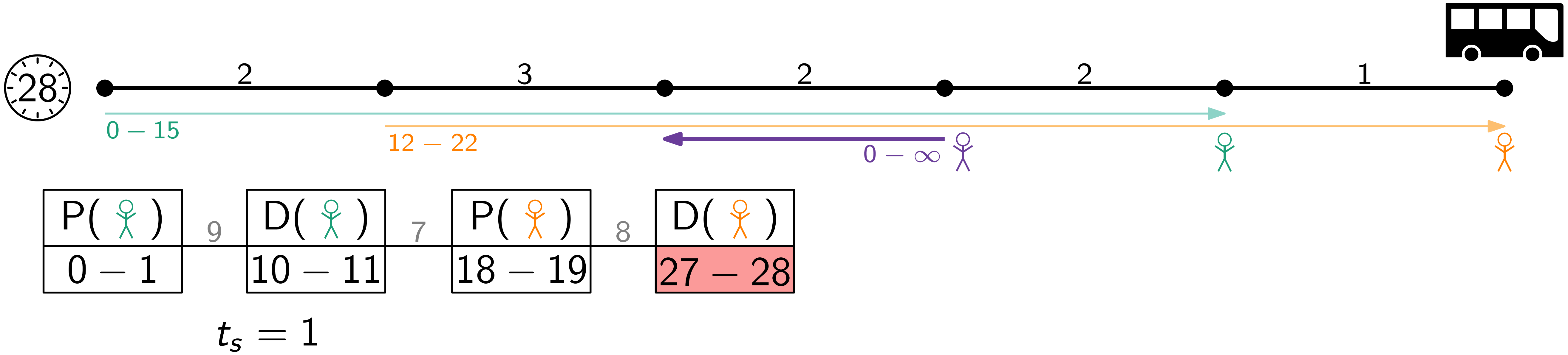
LINE-BASED DIAL-A-RIDE-PROBLEM (LIDARP)



Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p
- Service Promise (SP): passenger at most ρ times longer in bus than in direct route
- Service Time (ST): pick-up / drop-off takes time t_s

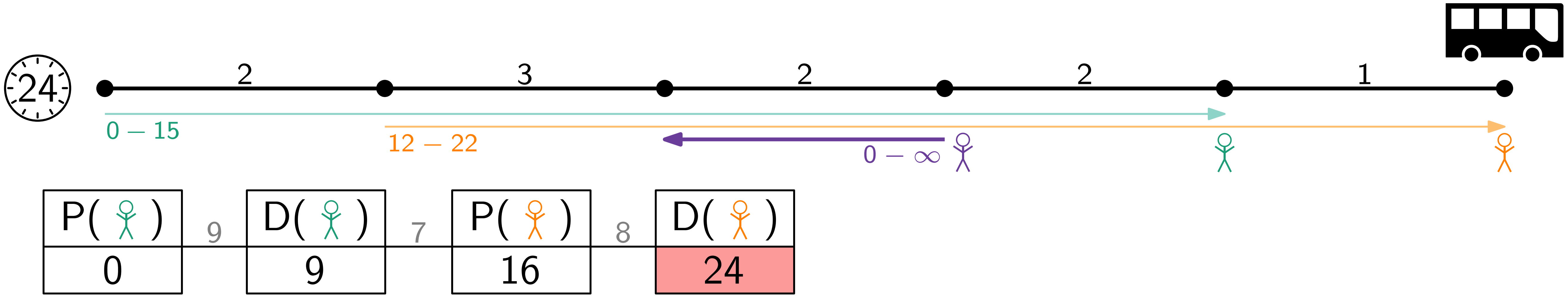
LINE-BASED DIAL-A-RIDE-PROBLEM (LIDARP)



Variants

- **Time Windows (TW):** earliest pick-up e_p and latest drop-off ℓ_p for request p
- **Service Promise (SP):** passenger at most ρ times longer in bus than in direct route
- **Service Time (ST):** pick-up / drop-off takes time t_s

line-based Dial-a-Ride-Problem (liDARP)

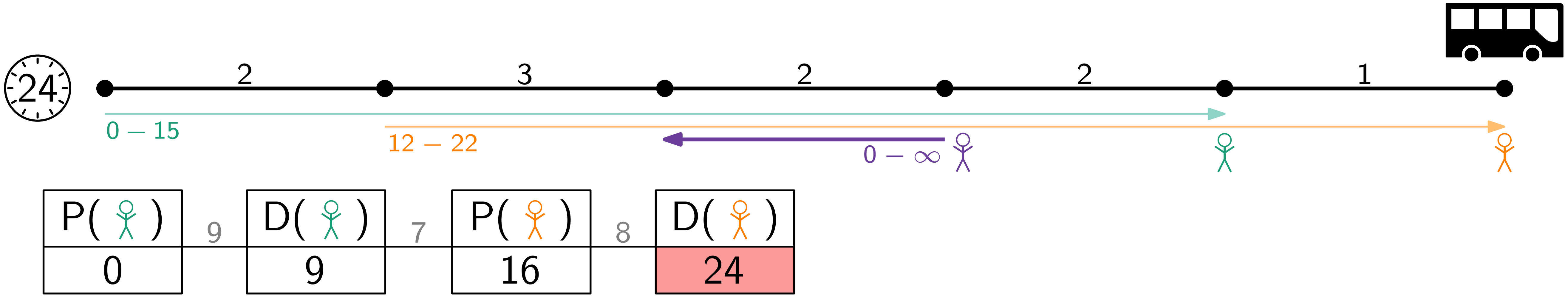


$t_s = 0$

Variants

- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p
- Service Promise (SP): passenger at most ρ times longer in bus than in direct route
- Service Time (ST): pick-up / drop-off takes time t_s

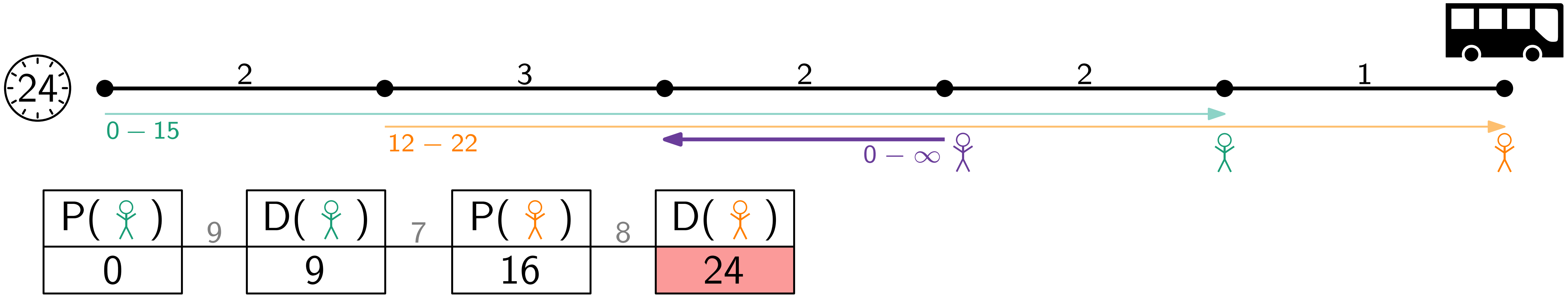
line-based Dial-a-Ride-Problem (liDARP)



Variants

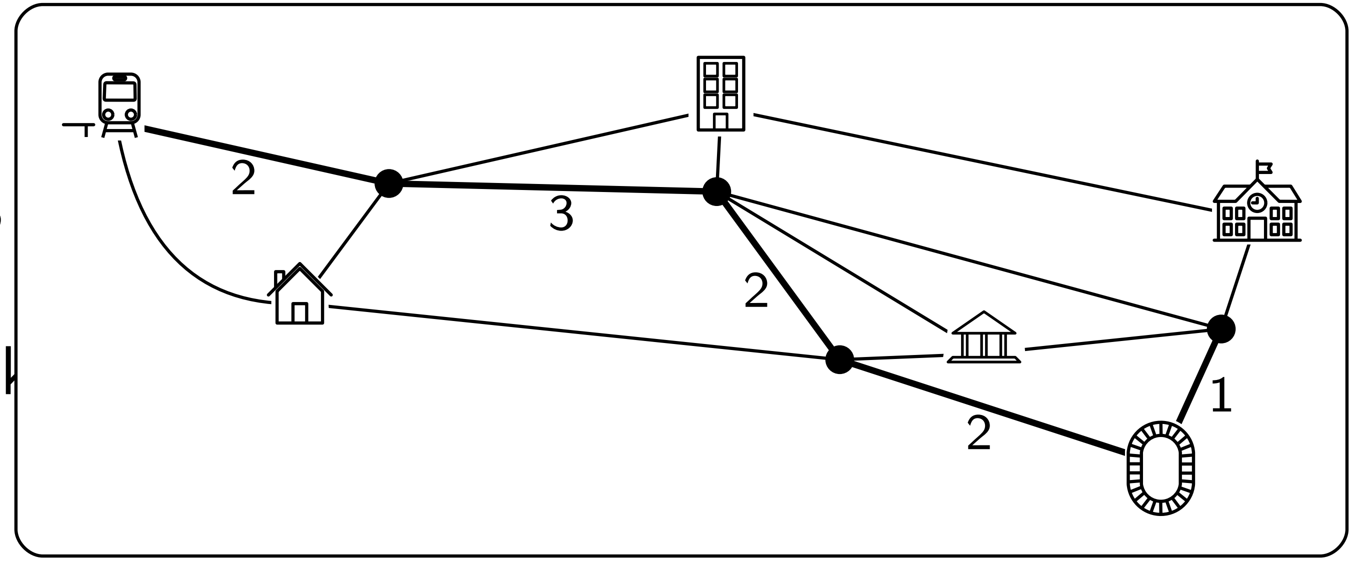
- Time Windows (TW): earliest pick-up e_p and latest drop-off ℓ_p for request p
- Service Promise (SP): passenger at most ρ times longer in bus than in direct route
- Service Time (ST): pick-up / drop-off takes time t_s
- Shortcuts (SC)

line-based Dial-a-Ride-Problem (liDARP)

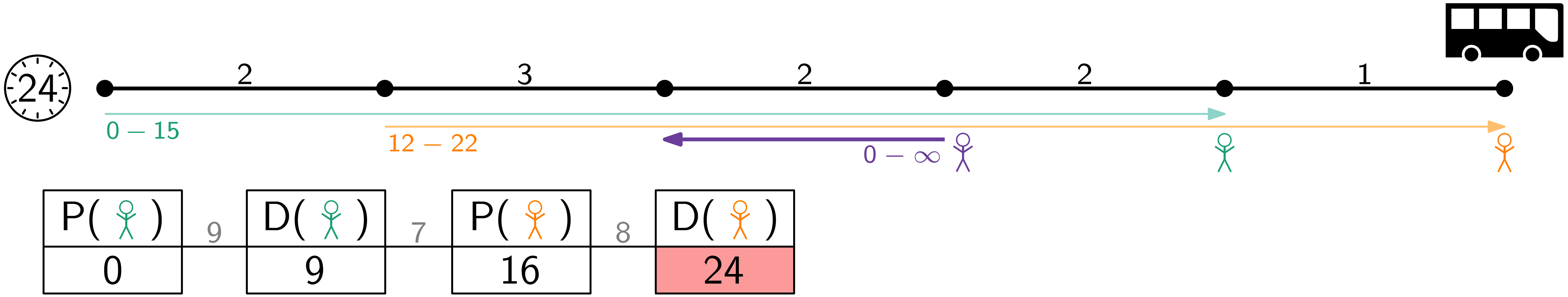


Variants

- Time Windows (TW): earliest pick-up e_p
- Service Promise (SP): passenger at most
- Service Time (ST): pick-up / drop-off tak
- Shortcuts (SC)

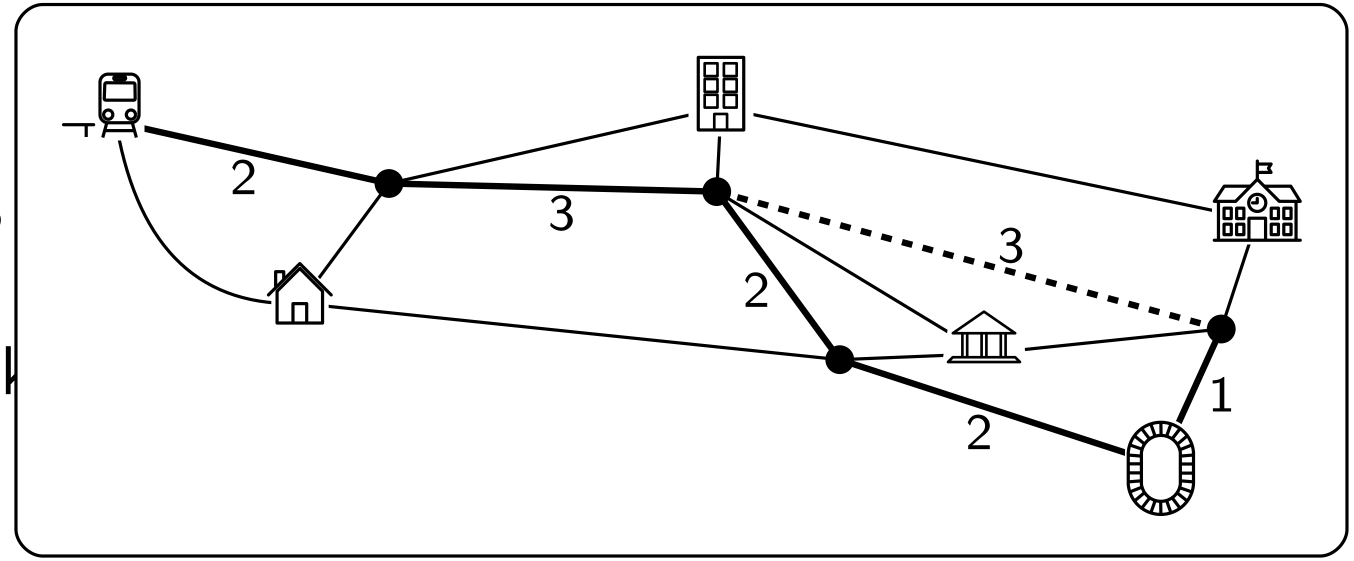


line-based Dial-a-Ride-Problem (liDARP)

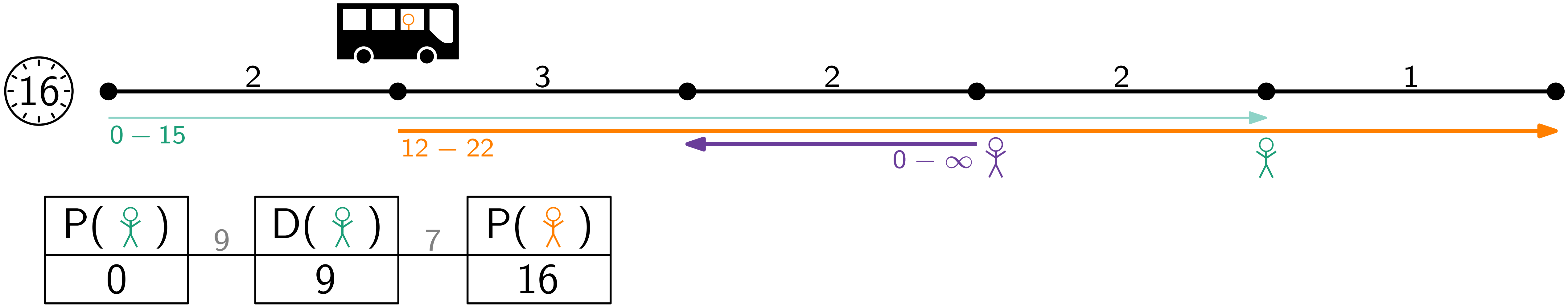


Variants

- Time Windows (TW): earliest pick-up e_p
- Service Promise (SP): passenger at most
- Service Time (ST): pick-up / drop-off tak
- Shortcuts (SC)

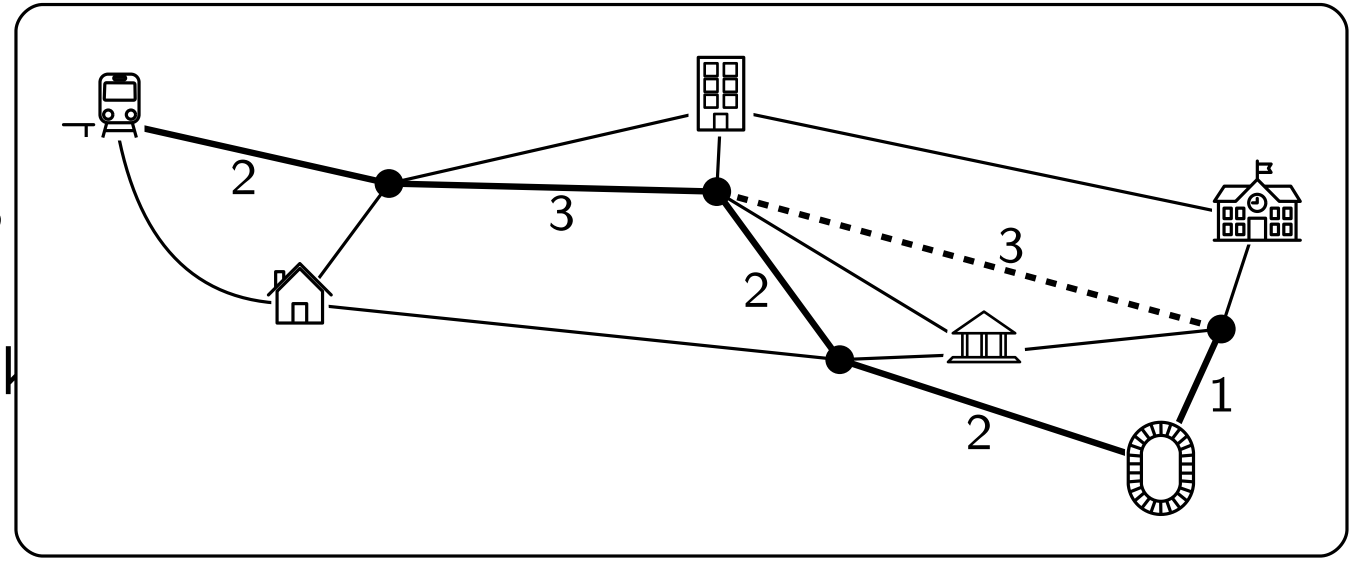


LINE-BASED DIAL-A-RIDE-PROBLEM (LIDARP)

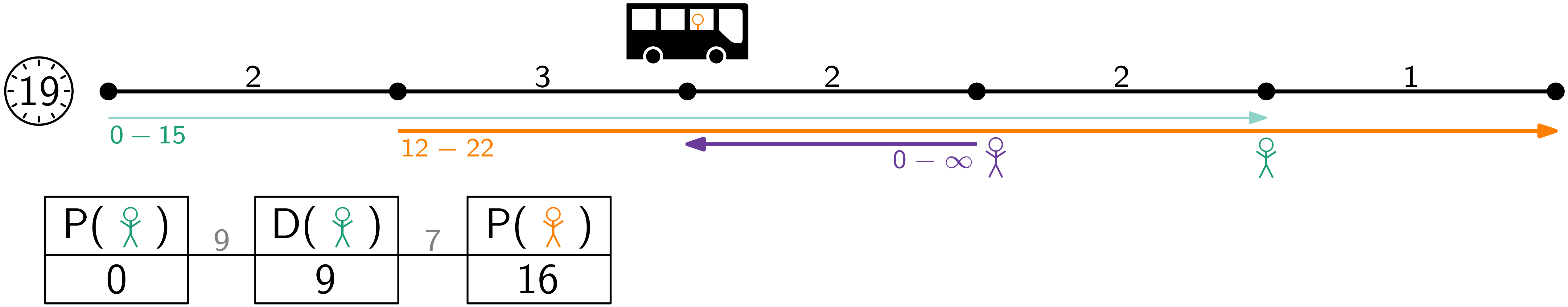


Variants

- Time Windows (TW): earliest pick-up e_p
- Service Promise (SP): passenger at most
- Service Time (ST): pick-up / drop-off tak
- Shortcuts (SC)

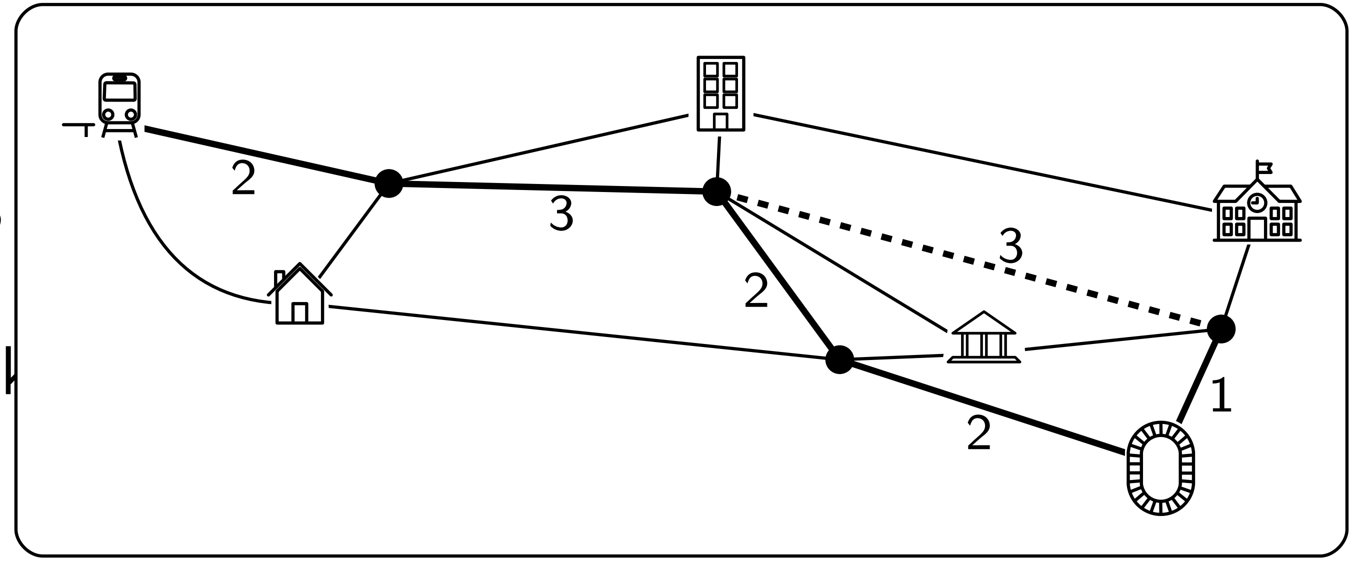


line-based Dial-a-Ride-Problem (liDARP)

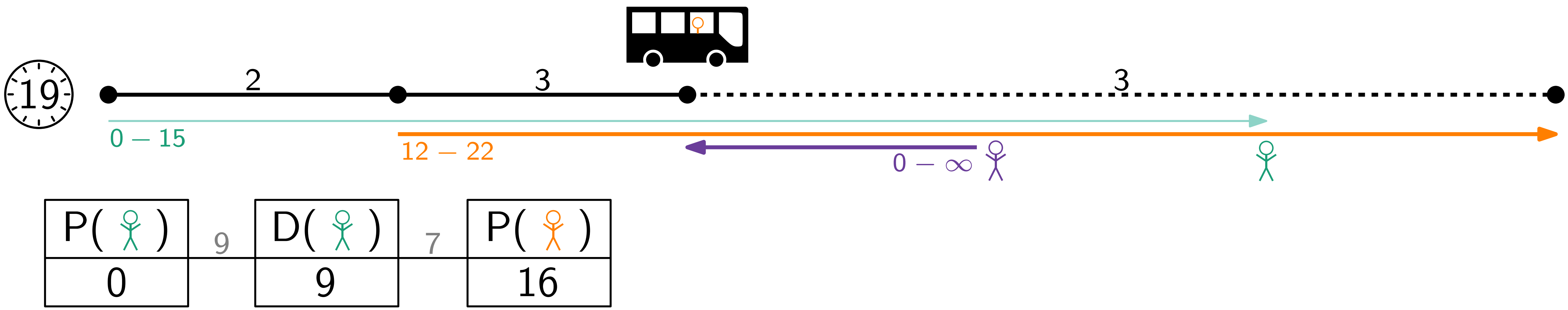


Variants

- Time Windows (TW): earliest pick-up e_p
- Service Promise (SP): passenger at most
- Service Time (ST): pick-up / drop-off tak
- Shortcuts (SC)

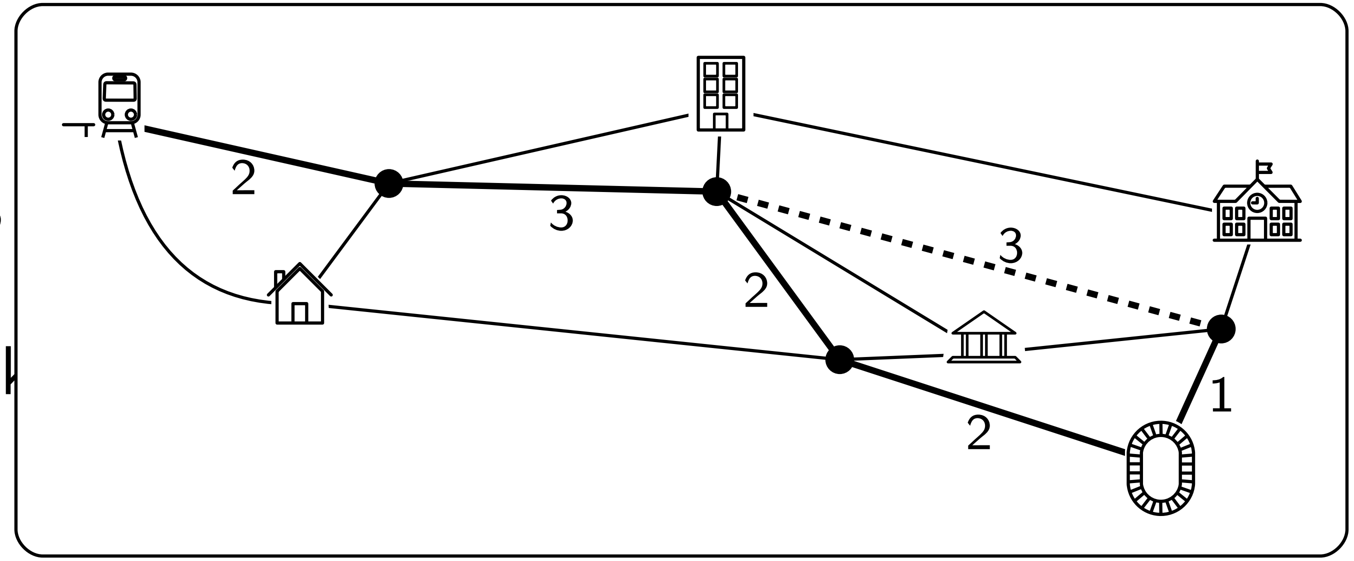


line-based Dial-a-Ride-Problem (liDARP)

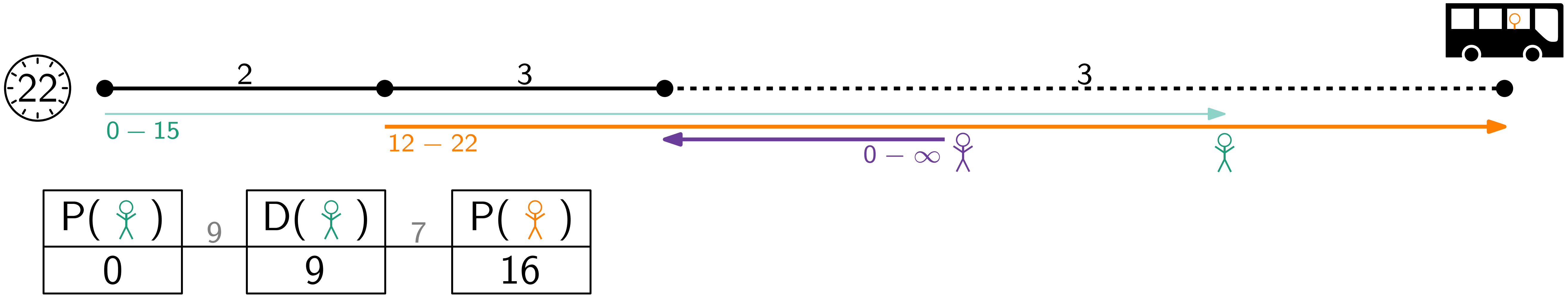


Variants

- Time Windows (TW): earliest pick-up e_p
- Service Promise (SP): passenger at most
- Service Time (ST): pick-up / drop-off tak
- Shortcuts (SC)

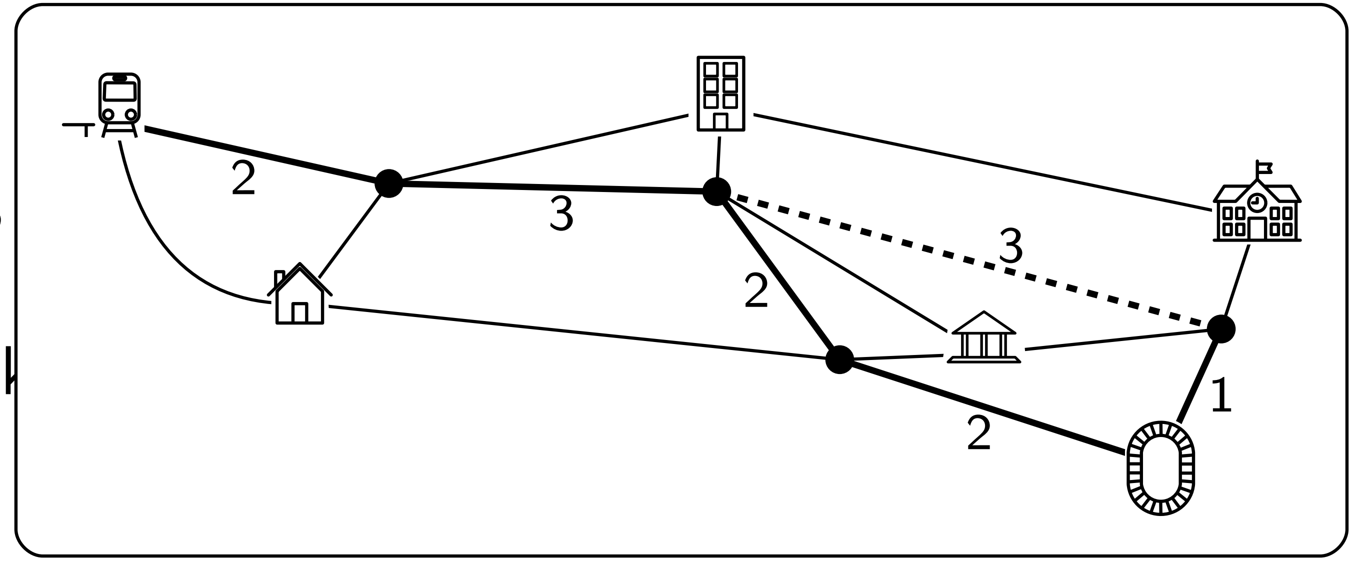


line-based Dial-a-Ride-Problem (liDARP)



Variants

- Time Windows (TW): earliest pick-up e_p
- Service Promise (SP): passenger at most
- Service Time (ST): pick-up / drop-off tak
- Shortcuts (SC)

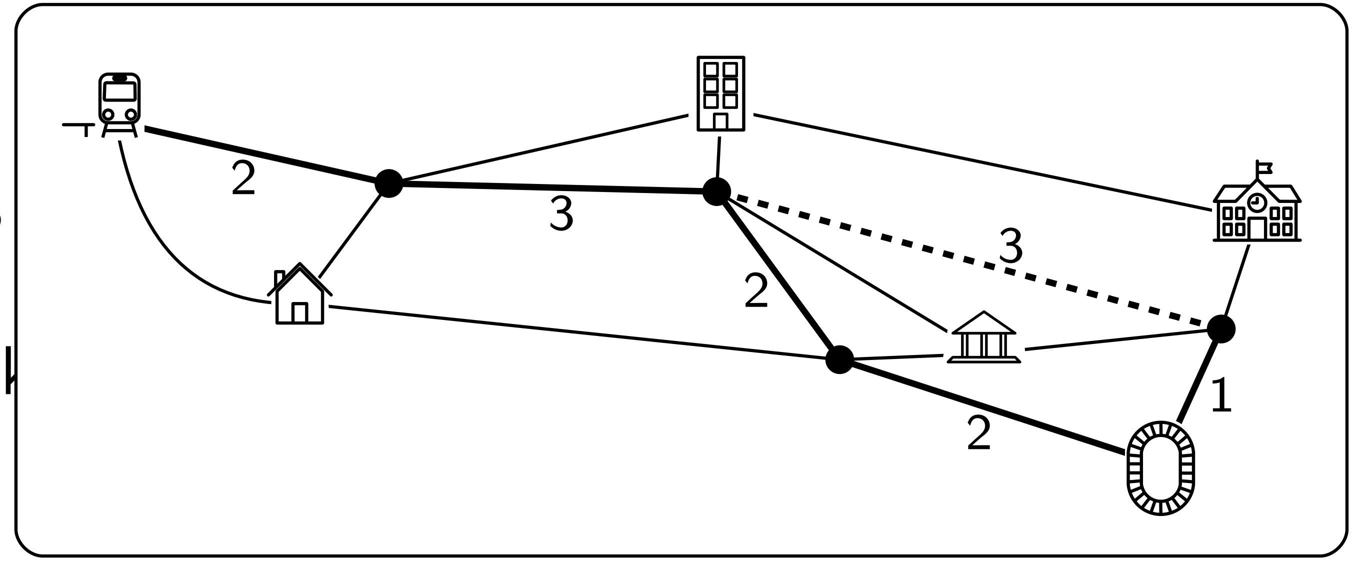


line-based Dial-a-Ride-Problem (liDARP)

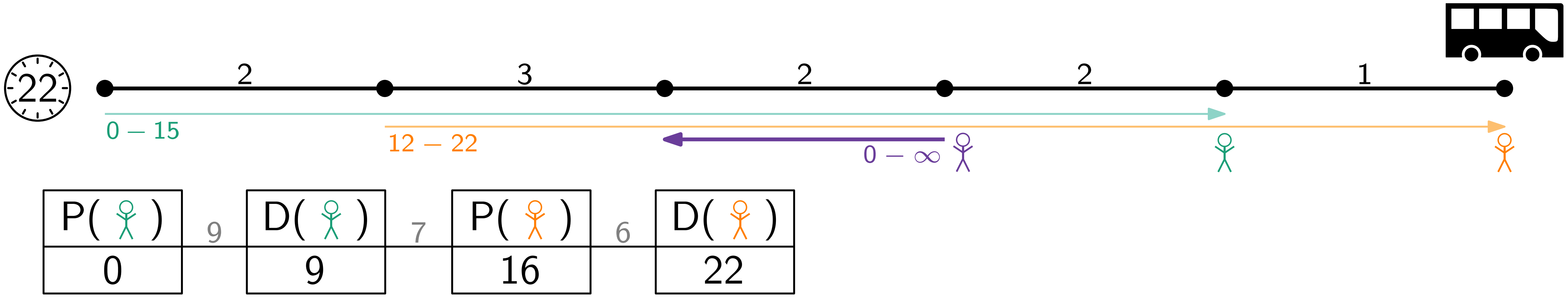
P( )	9	D( )	7	P( )	6	D( )
0		9		16		22

Variants

- Time Windows (TW): earliest pick-up e_p
- Service Promise (SP): passenger at most
- Service Time (ST): pick-up / drop-off tak
- Shortcuts (SC)



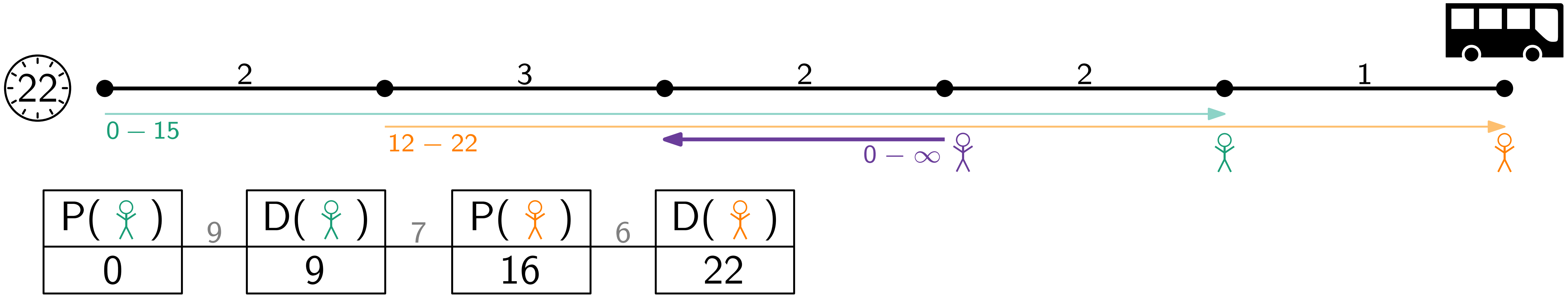
line-based Dial-a-Ride-Problem (liDARP)



Tour

- respects directionality and capacity
- timestamps adhere to distances and service time

line-based Dial-a-Ride-Problem (liDARP)



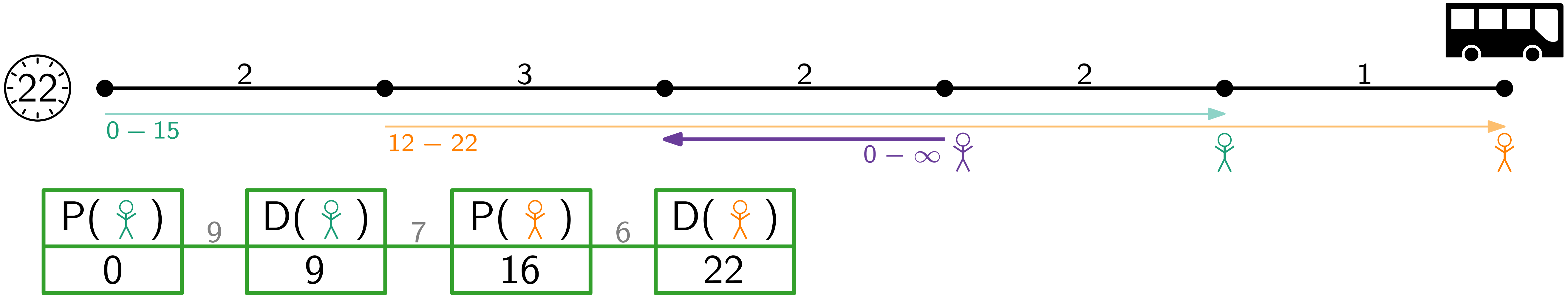
Tour

- respects directionality and capacity
- timestamps adhere to distances and service time

Feasible

- respects time windows and service promise

LINE-BASED DIAL-A-RIDE-PROBLEM (LIDARP)



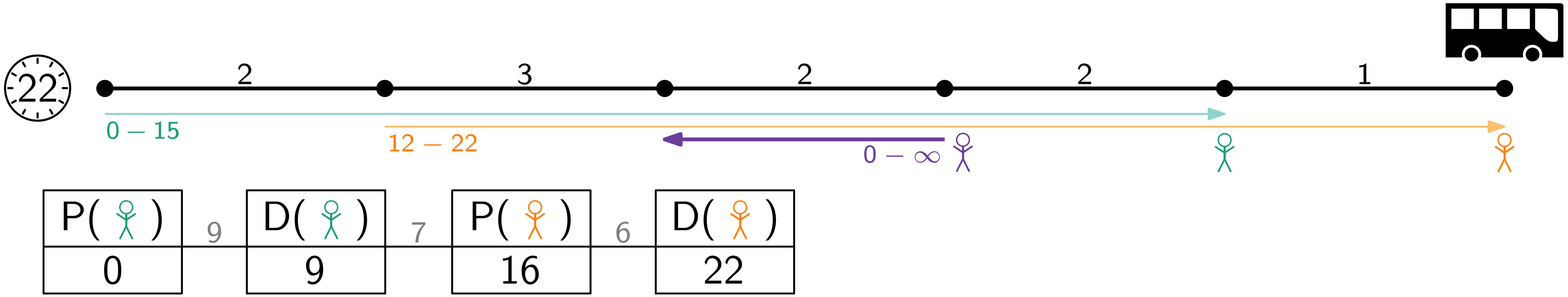
Tour

- respects directionality and capacity
- timestamps adhere to distances and service time

Feasible

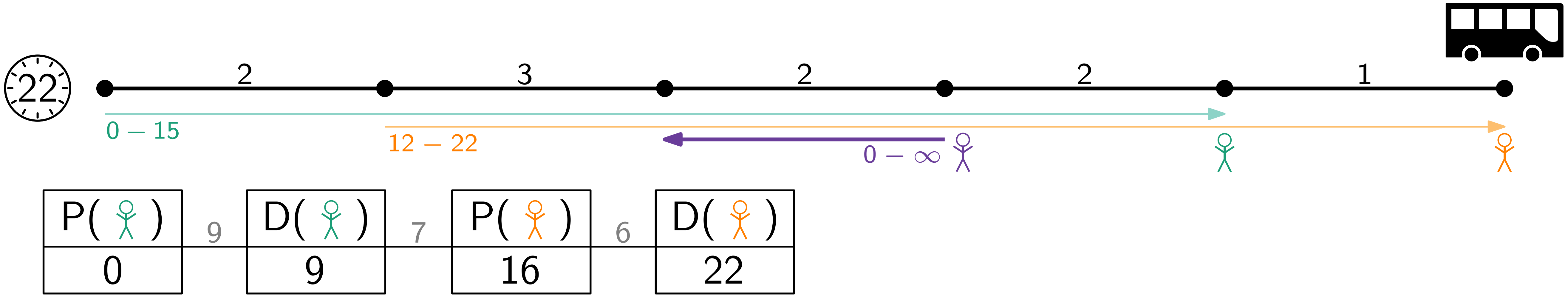
- respects time windows and service promise

line-based Dial-a-Ride-Problem (liDARP)



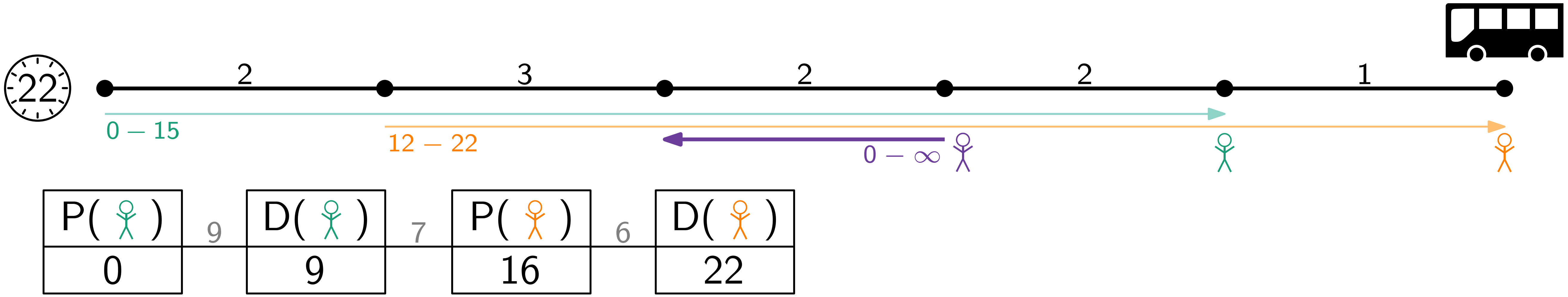
Objective:

line-based Dial-a-Ride-Problem (liDARP)



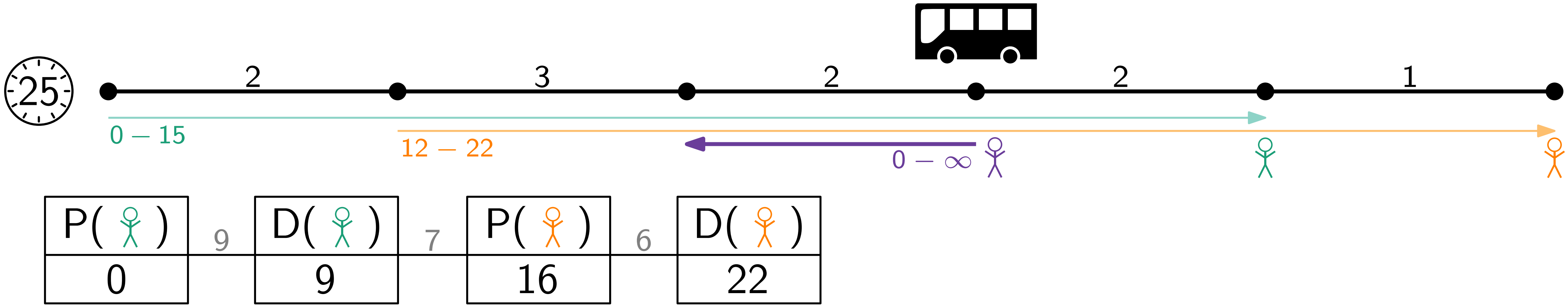
Objective: serve as many requests as possible

line-based Dial-a-Ride-Problem (liDARP)



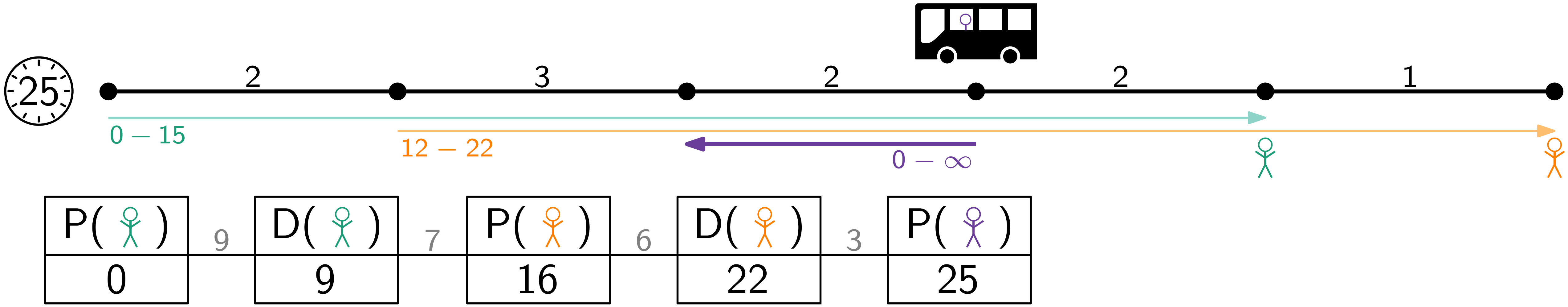
Objective: serve as many requests as possible

line-based Dial-a-Ride-Problem (liDARP)



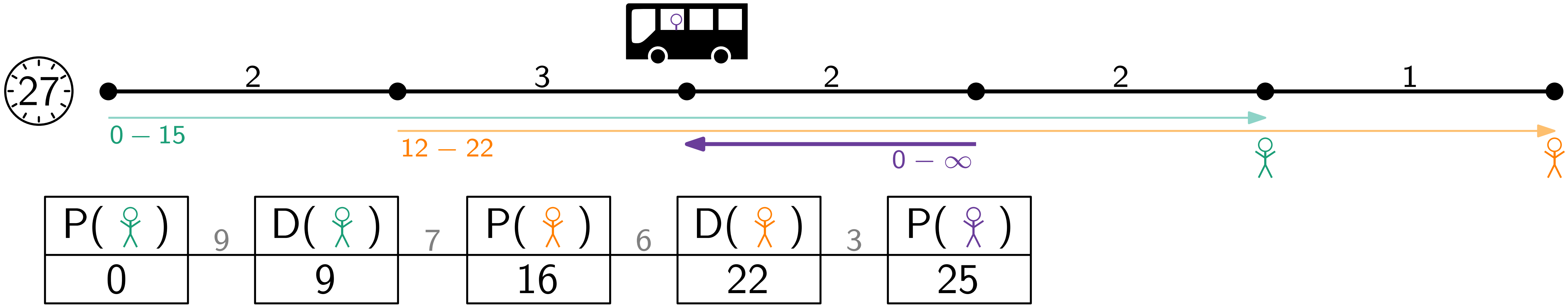
Objective: serve as many requests as possible

line-based Dial-a-Ride-Problem (liDARP)



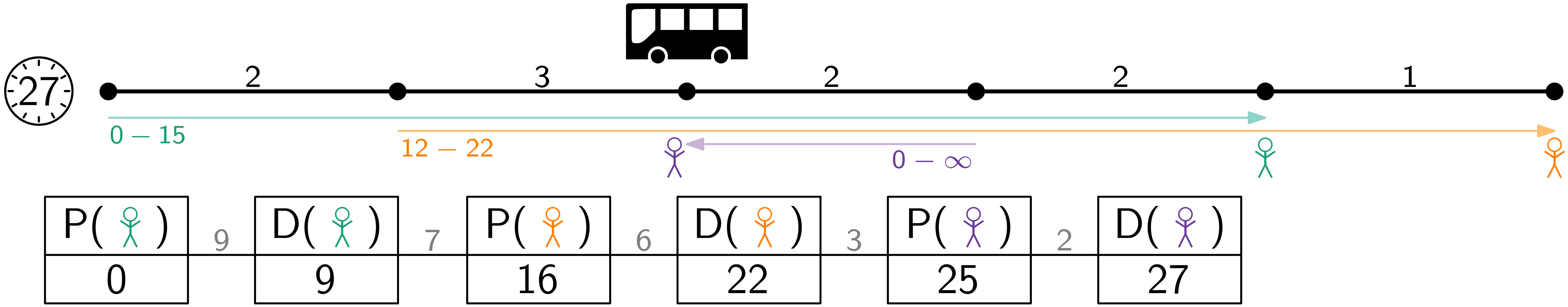
Objective: serve as many requests as possible

line-based Dial-a-Ride-Problem (liDARP)



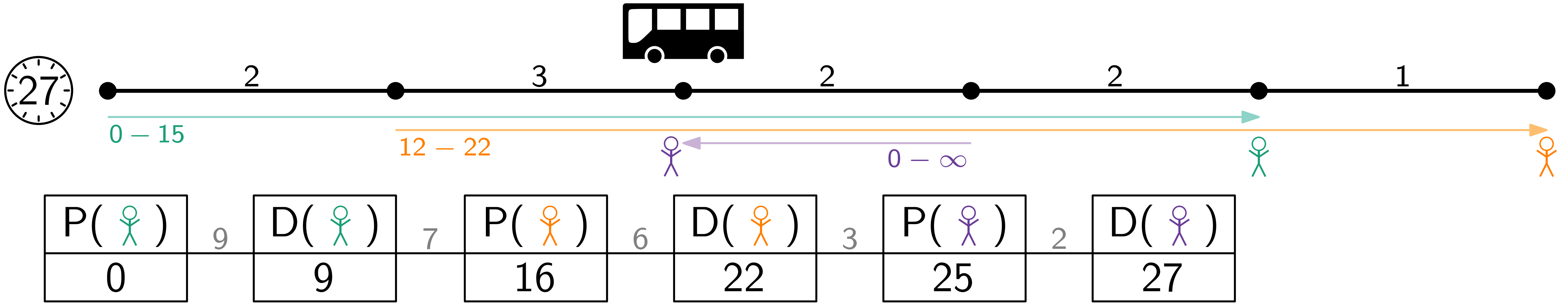
Objective: serve as many requests as possible

line-based Dial-a-Ride-Problem (liDARP)



Objective: serve as many requests as possible

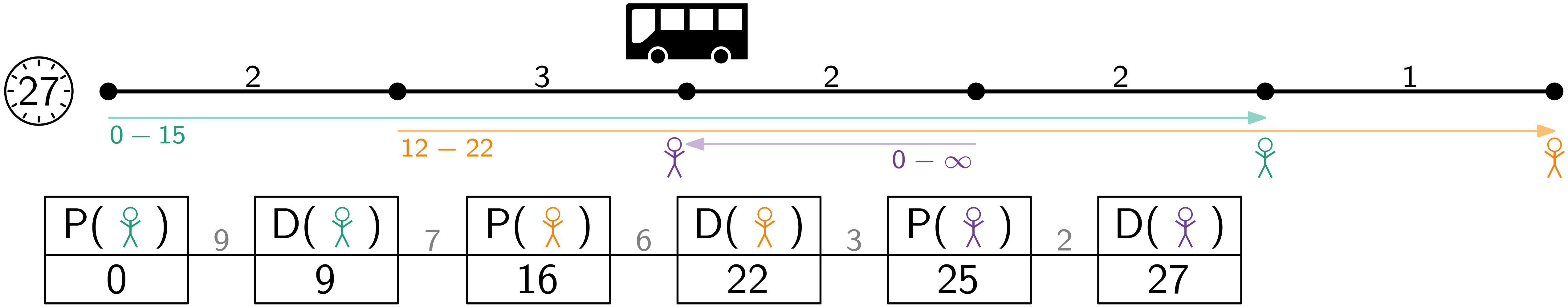
line-based Dial-a-Ride-Problem (liDARP)



Objective: serve as many requests as possible

TW	✓
SP	✓
ST	✗
SC	✓

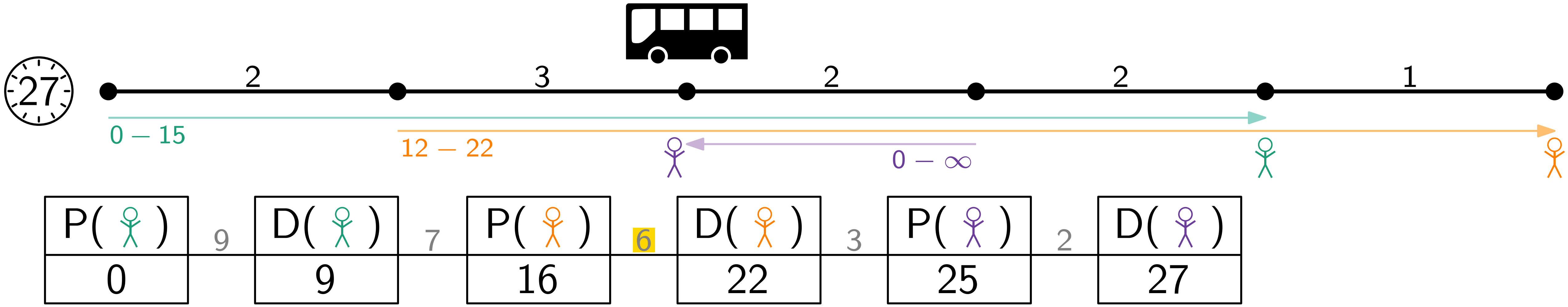
line-based Dial-a-Ride-Problem (liDARP)



Objective: serve as many requests as possible

TW	✓
SP	✓
ST	✗
SC	✗

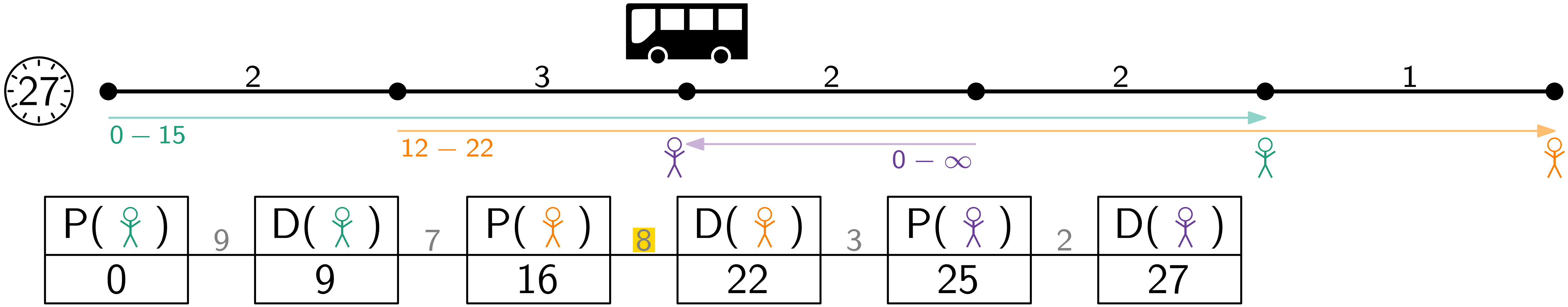
line-based Dial-a-Ride-Problem (liDARP)



Objective: serve as many requests as possible

TW	✓
SP	✓
ST	✗
SC	✗

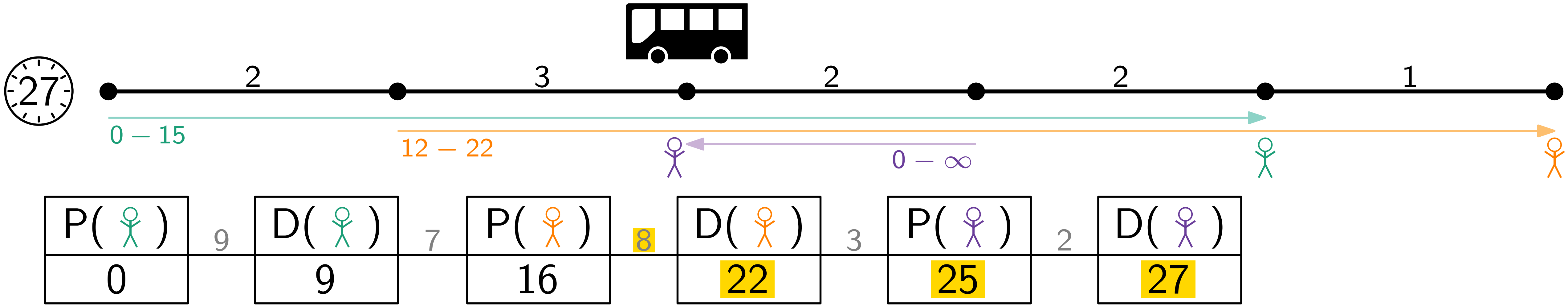
line-based Dial-a-Ride-Problem (liDARP)



Objective: serve as many requests as possible

TW	✓
SP	✓
ST	✗
SC	✗

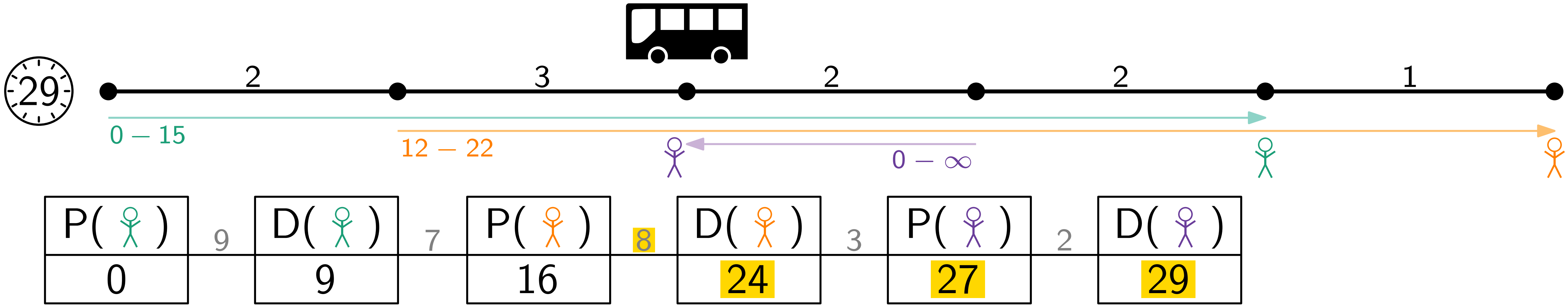
line-based Dial-a-Ride-Problem (liDARP)



Objective: serve as many requests as possible

TW	✓
SP	✓
ST	✗
SC	✗

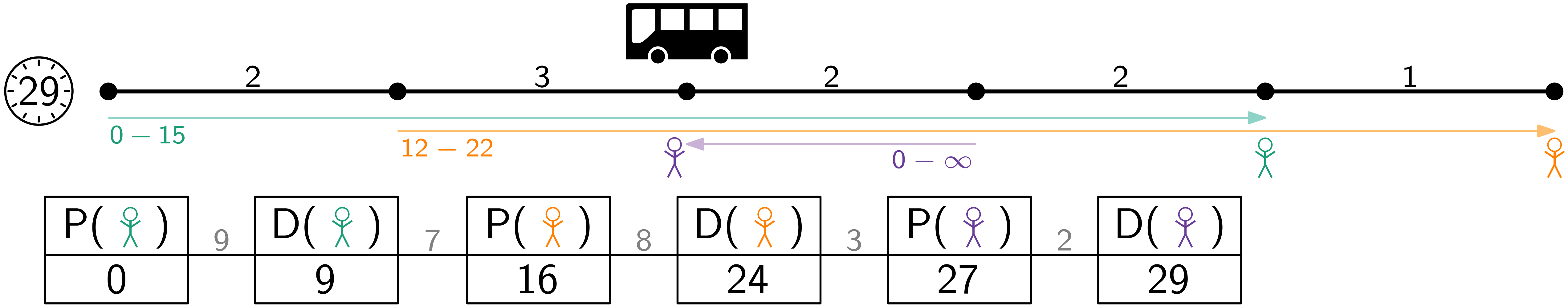
line-based Dial-a-Ride-Problem (liDARP)



Objective: serve as many requests as possible

TW	✓
SP	✓
ST	✗
SC	✗

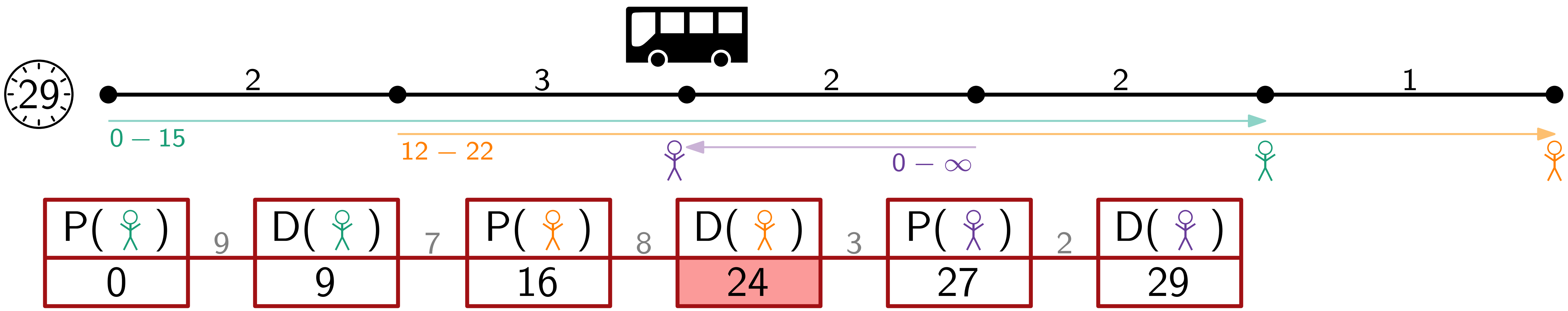
line-based Dial-a-Ride-Problem (liDARP)



Objective: serve as many requests as possible

TW	✓
SP	✓
ST	✗
SC	✗

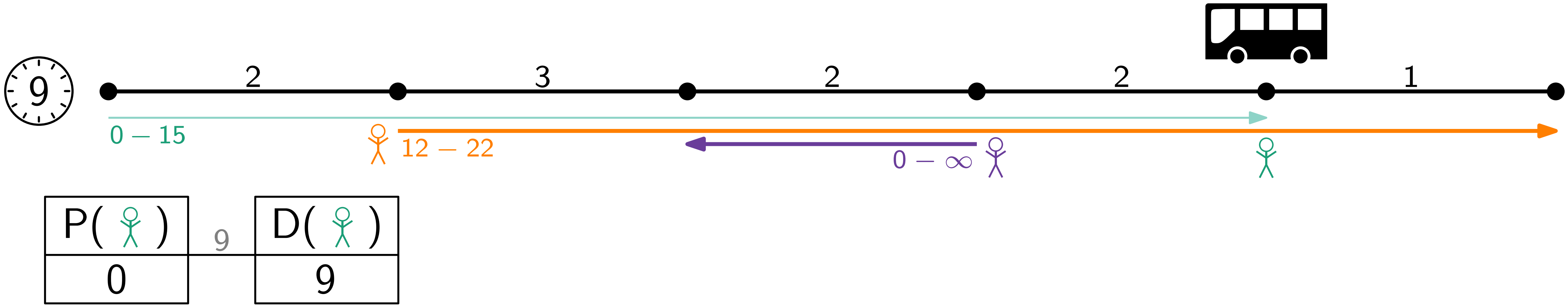
line-based Dial-a-Ride-Problem (liDARP)



Objective: serve as many requests as possible

TW	✓
SP	✓
ST	✗
SC	✗

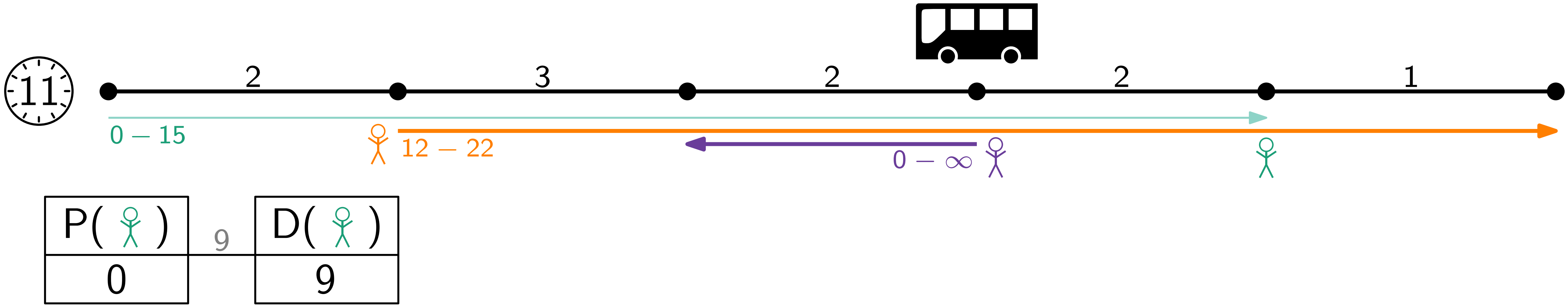
LINE-BASED DIAL-A-RIDE-PROBLEM (liDARP)



Objective: serve as many requests as possible

TW	✓
SP	✓
ST	✗
SC	✗

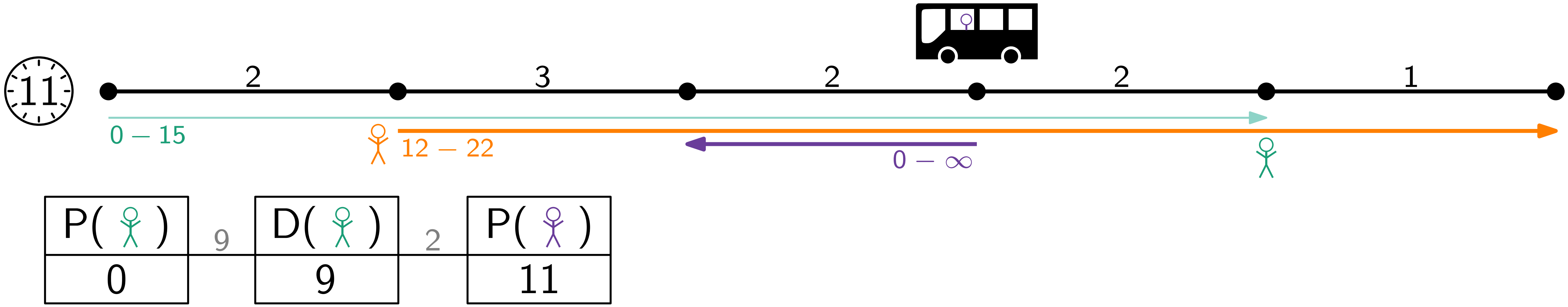
line-based Dial-a-Ride-Problem (liDARP)



Objective: serve as many requests as possible

TW	✓
SP	✓
ST	✗
SC	✗

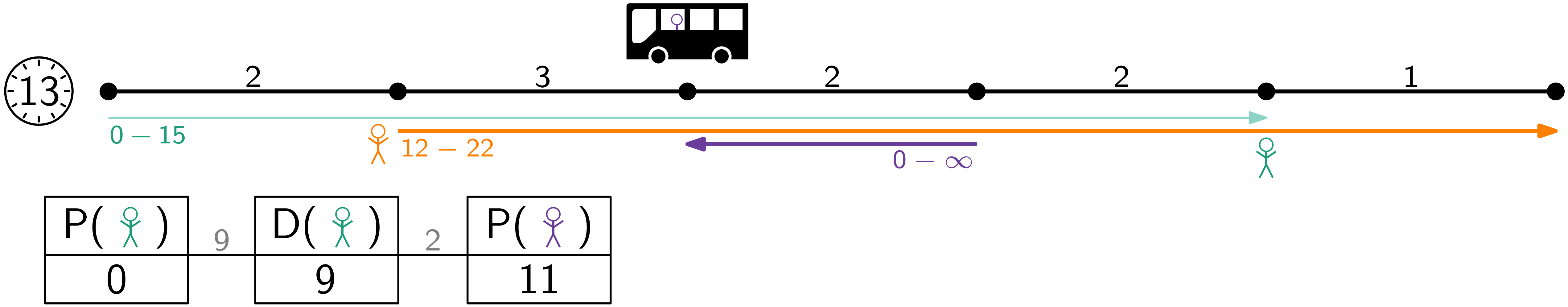
line-based Dial-a-Ride-Problem (liDARP)



Objective: serve as many requests as possible

TW	✓
SP	✓
ST	✗
SC	✗

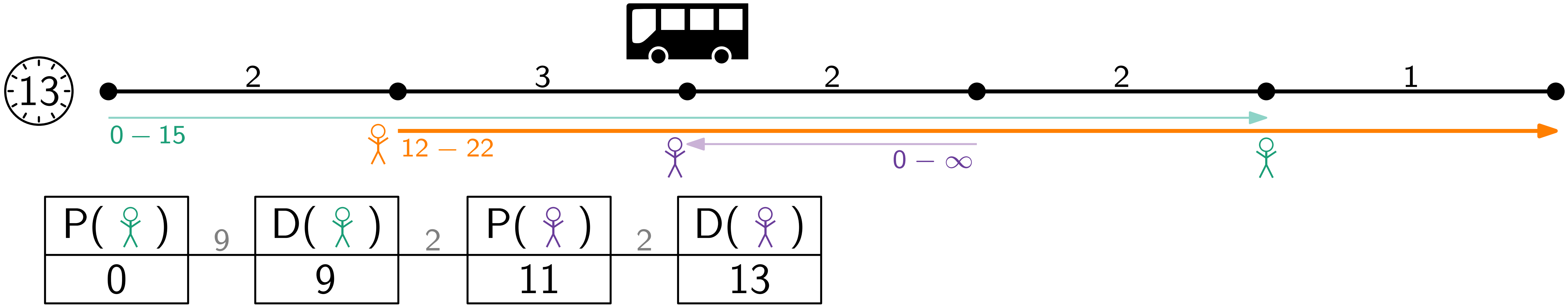
line-based Dial-a-Ride-Problem (liDARP)



Objective: serve as many requests as possible

TW	✓
SP	✓
ST	✗
SC	✗

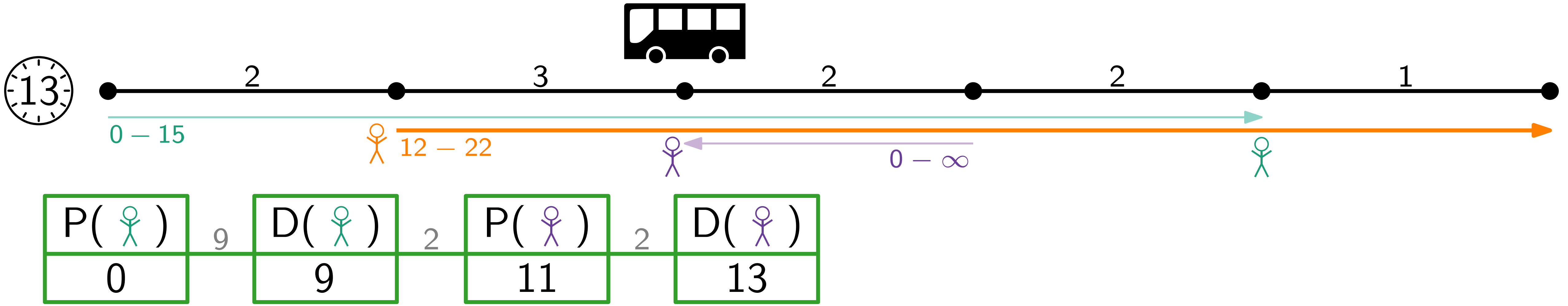
LINE-BASED DIAL-A-RIDE-PROBLEM (LIDARP)



Objective: serve as many requests as possible

TW	✓
SP	✓
ST	✗
SC	✗

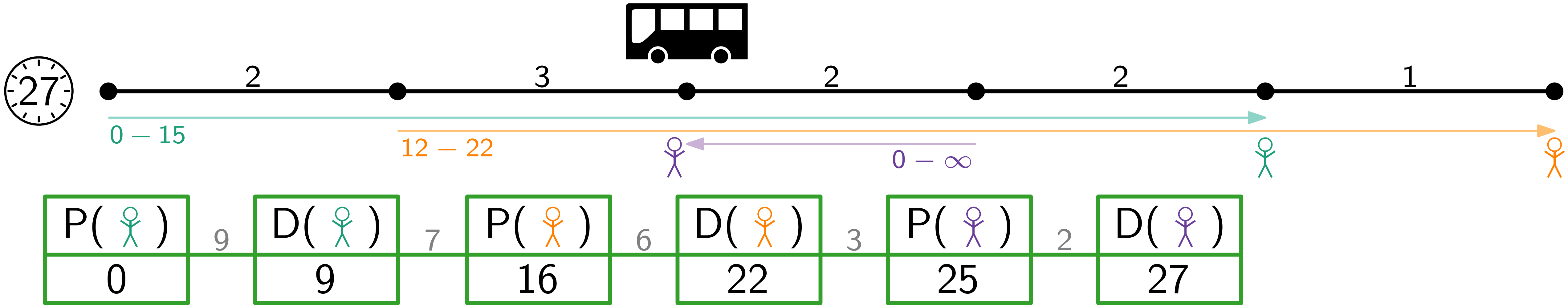
line-based Dial-a-Ride-Problem (liDARP)



Objective: serve as many requests as possible

TW	✓
SP	✓
ST	✗
SC	✗

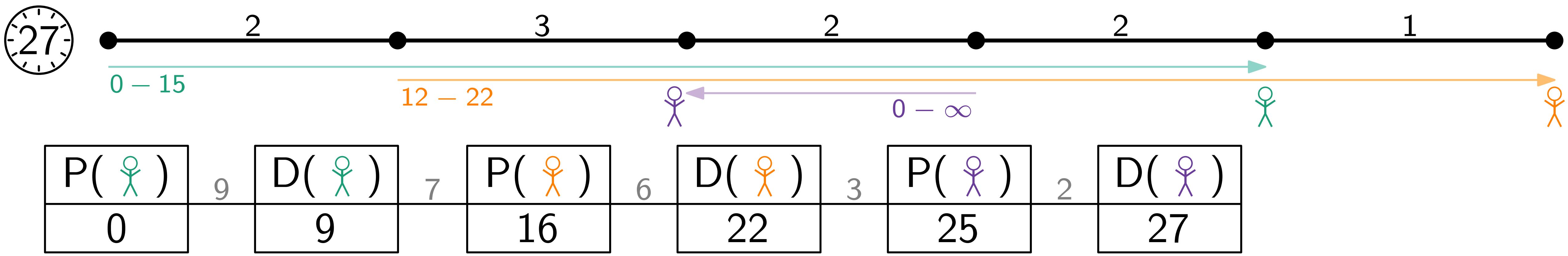
line-based Dial-a-Ride-Problem (liDARP)



Objective: serve as many requests as possible

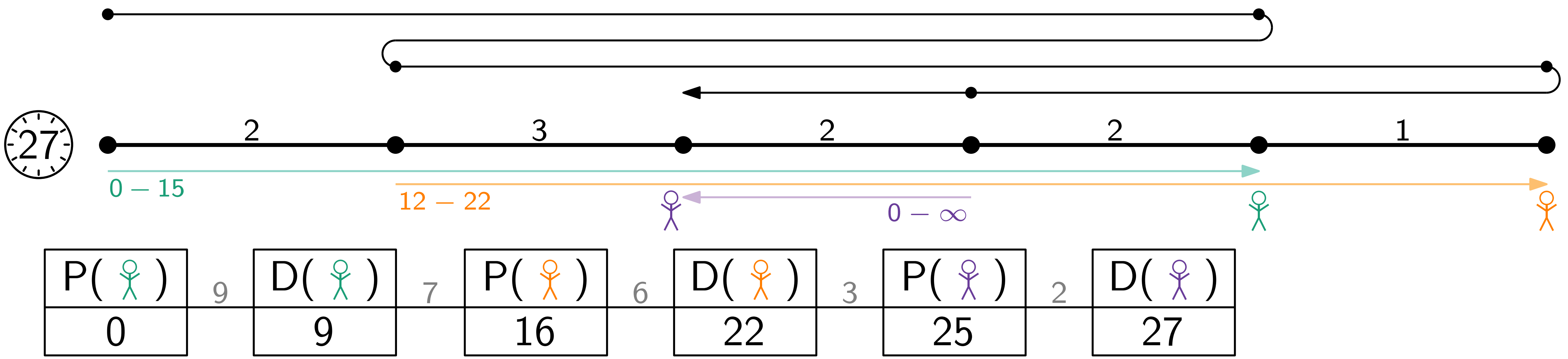
TW	✓
SP	✓
ST	✗
SC	✓

MinTurn



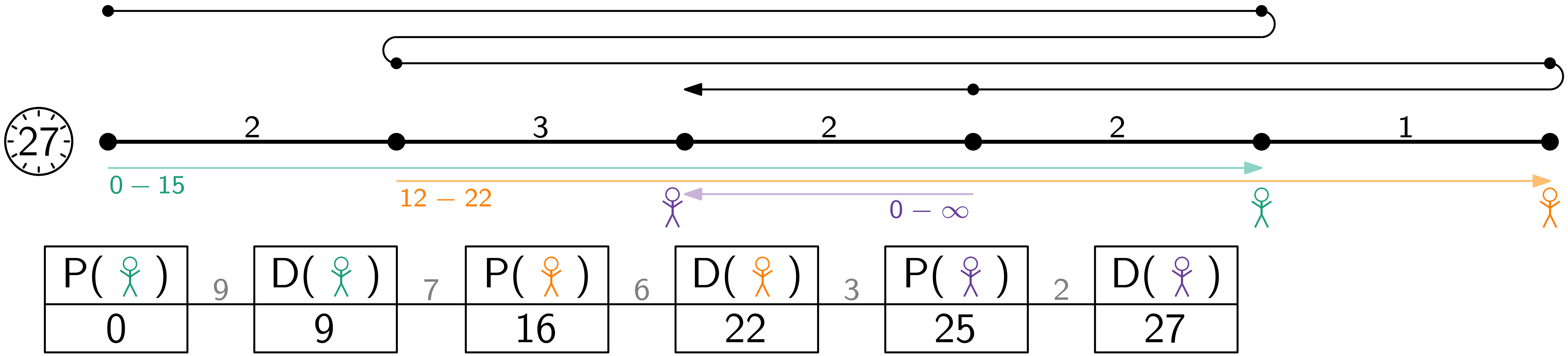
TW	✓
SP	✓
ST	✗
SC	✓

MinTurn



TW	✓
SP	✓
ST	✗
SC	✓

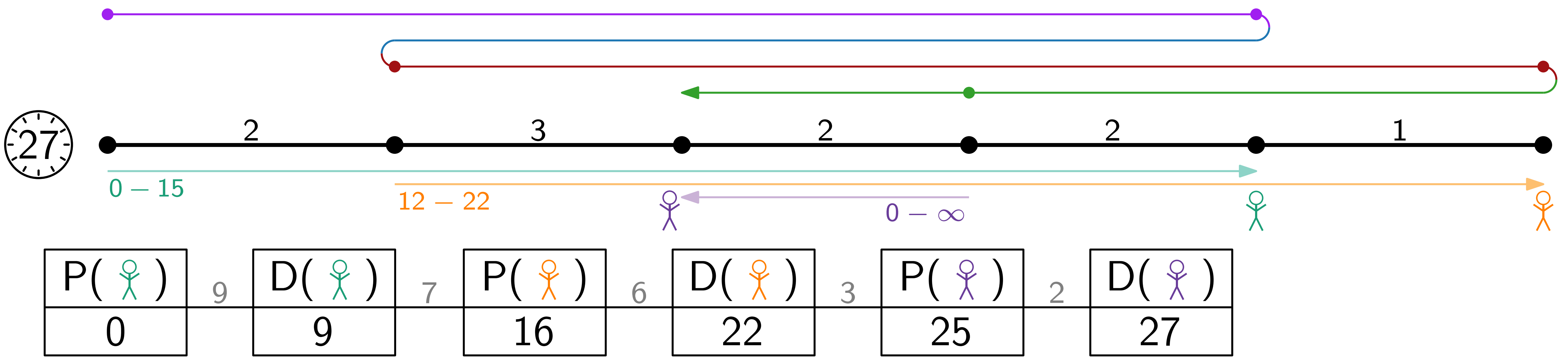
MinTurn



Subtour: tour segment where bus does not change direction

TW	✓
SP	✓
ST	✗
SC	✓

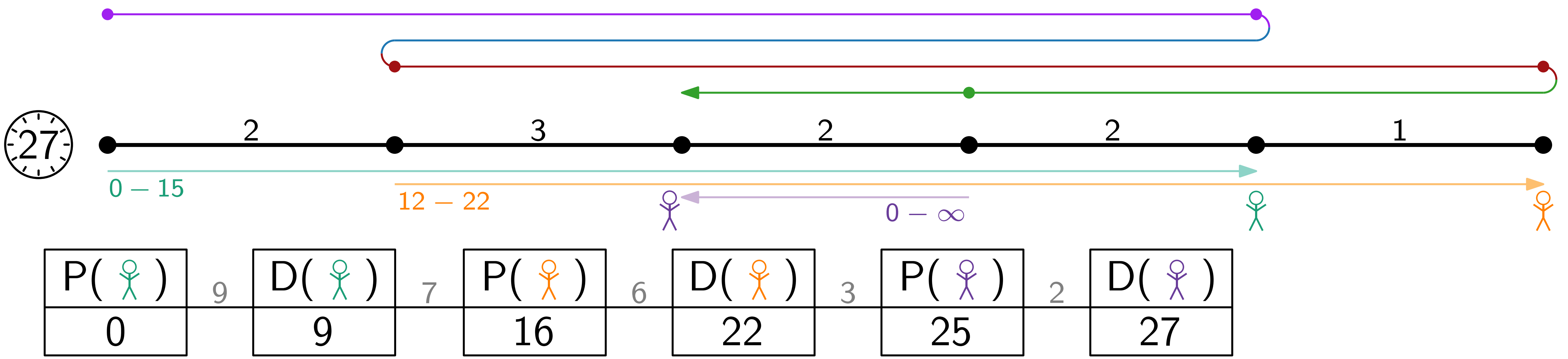
MinTurn



Subtour: tour segment where bus does not change direction

TW	✓
SP	✓
ST	✗
SC	✓

MinTurn

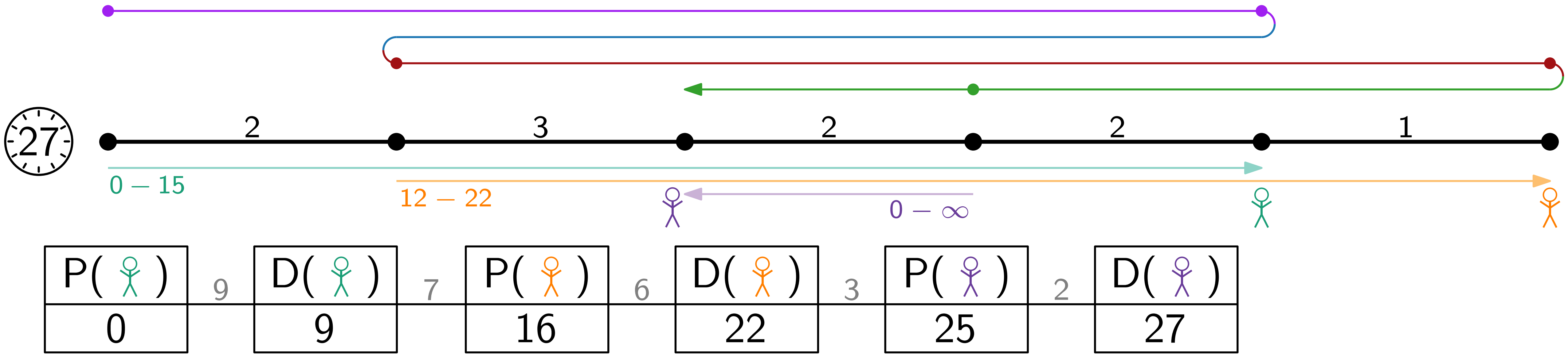


Subtour: tour segment where bus does not change direction

MinTurn

TW	✓
SP	✓
ST	✗
SC	✓

MinTurn



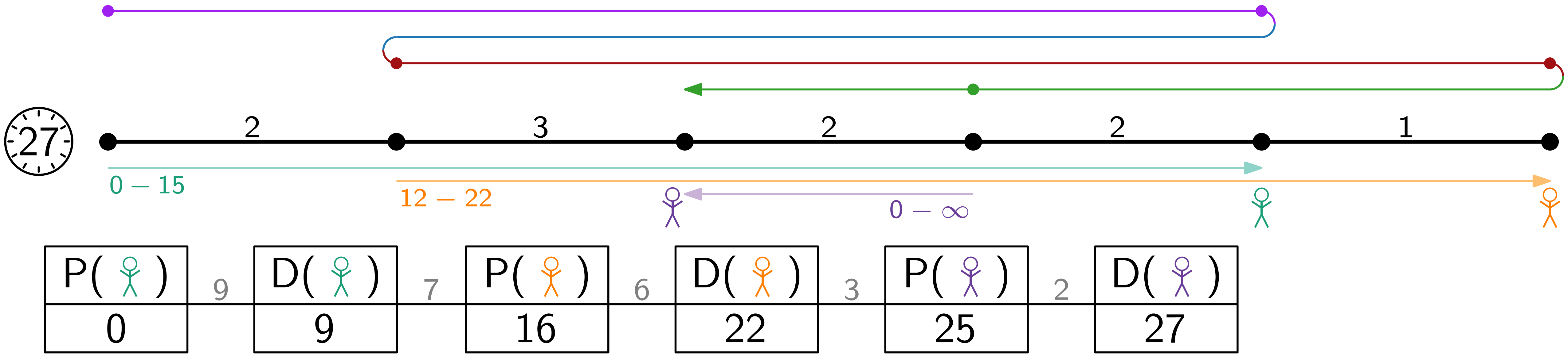
Subtour: tour segment where bus does not change direction

MinTurn

Input: liDARP-Instance

TW	✓
SP	✓
ST	✗
SC	✓

MinTurn



Subtour: tour segment where bus does not change direction

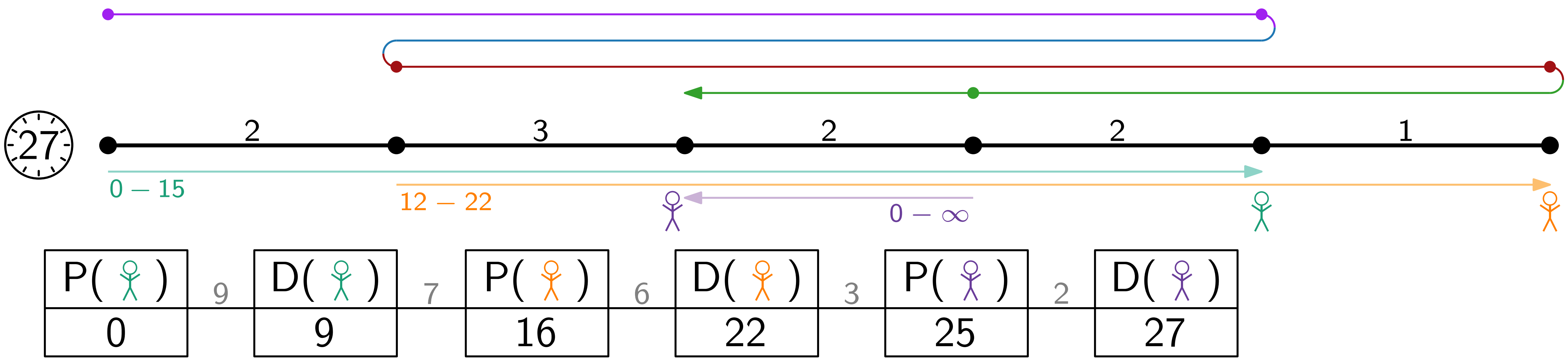
MinTurn

Input: liDARP-Instance

Output: minimal number of subtours of a tour in an optimal liDARP solution

TW	✓
SP	✓
ST	✗
SC	✓

MinTurn



Subtour: tour segment where bus does not change direction

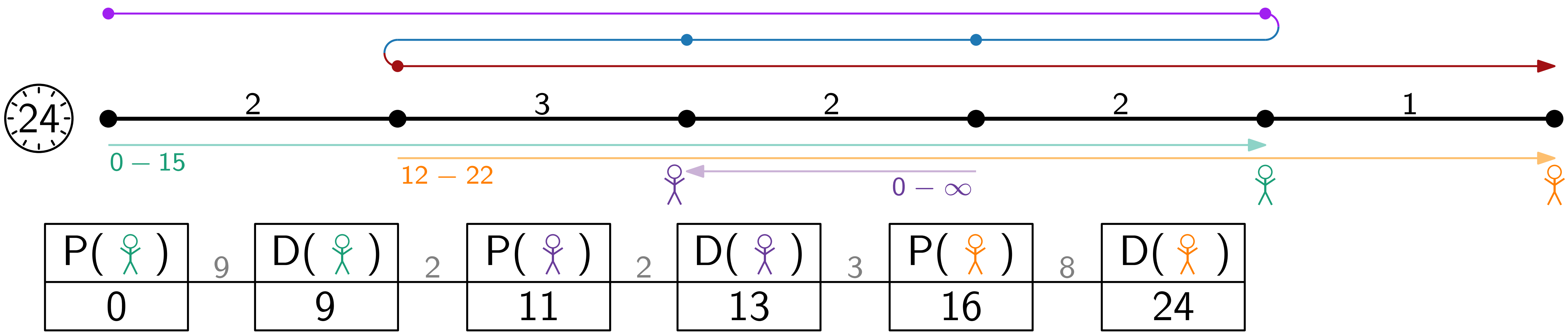
MinTurn

Input: liDARP-Instance

Output: minimal number of subtours of a tour in an optimal liDARP solution $=: \tau$

TW	✓
SP	✓
ST	✗
SC	✓

MinTurn



Subtour: tour segment where bus does not change direction

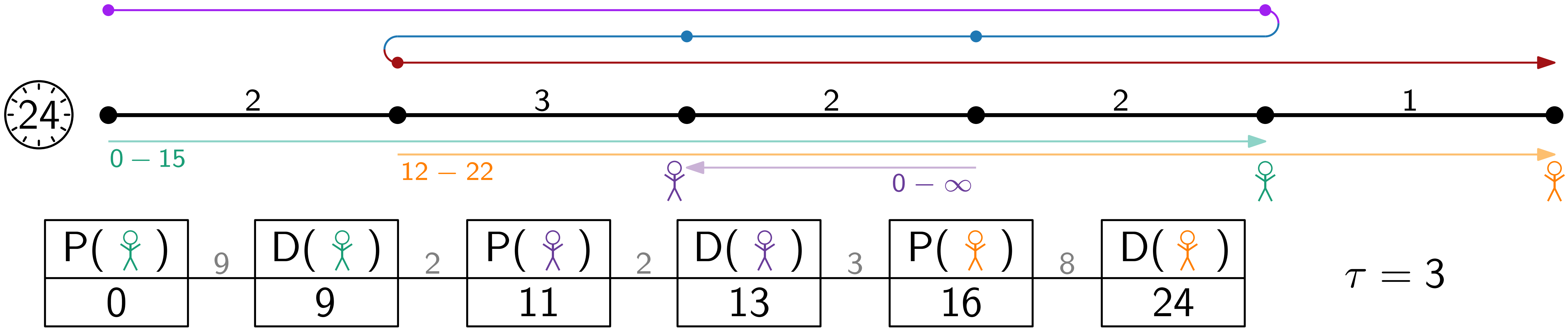
MinTurn

Input: LIDARP-Instance

Output: minimal number of subtours of a tour in an optimal LIDARP solution $=: \tau$

TW	✓
SP	✓
ST	✗
SC	✓

MinTurn



$\tau = 3$

Subtour: tour segment where bus does not change direction

MinTurn

Input: liDARP-Instance

Output: minimal number of subtours of a tour in an optimal liDARP solution $=: \tau$

TW	✓
SP	✓
ST	✗
SC	✓

Contribution

Contribution

Related Work

- study LIDARP with other objectives
- assume all requests can be served

Contribution

Related Work

- study LIDARP with other objectives
- assume all requests can be served

Complexity

Contribution

Related Work

- study LIDARP with other objectives
- assume all requests can be served

#Vehicles	k
Capacity	c
Time Windows	TW
Service Promise	SP
Service Time	ST
Shortcuts	SC

Complexity

LIDARP

k	c	TW	SP	ST	SC

Contribution

Related Work

- study LIDARP with other objectives
- assume all requests can be served

$\mathcal{P}$

#Vehicles	k
Capacity	c
Time Windows	TW
Service Promise	SP
Service Time	ST
Shortcuts	SC

Complexity

LIDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓

Contribution

Related Work

- study LIDARP with other objectives
- assume all requests can be served

$\mathcal{P}$	$\mathcal{NP}$-hard

#Vehicles	k
Capacity	c
Time Windows	TW
Service Promise	SP
Service Time	ST
Shortcuts	SC

Complexity

LIDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓
≥ 1	≥ 1	✓	✗	✗	✗

Contribution

Related Work

- study LIDARP with other objectives
- assume all requests can be served

$\mathcal{P}$	$\mathcal{NP}$-hard

#Vehicles	k
Capacity	c
Time Windows	TW
Service Promise	SP
Service Time	ST
Shortcuts	SC

Complexity

LIDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓
≥ 1	≥ 1	✓	✗	✗	✗

MINTURN

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✗	✗
≥ 1	≥ 1	✗	✗	✓	✓
≥ 1	1	✗	✓	✓	✓
≥ 1	≥ 2	✗	✓	✗	✓
≥ 1	≥ 2	✗	✓	✓	✗
≥ 1	≥ 1	✓	✗	✗	✗

Contribution

Related Work

- study LIDARP with other objectives
- assume all requests can be served

Parameterized Algorithms

Complexity

LIDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓
≥ 1	≥ 1	✓	✗	✗	✗

MinTurn

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✗	✗
≥ 1	≥ 1	✗	✗	✓	✓
≥ 1	1	✗	✓	✓	✓
≥ 1	≥ 2	✗	✓	✗	✓
≥ 1	≥ 2	✗	✓	✓	✗
≥ 1	≥ 1	✓	✗	✗	✗

Contribution

Related Work

- study LIDARP with other objectives
- assume all requests can be served

Parameterized Algorithms

Parameters

- $k :=$ #vehicles
- $c :=$ capacity

Complexity

LIDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓
≥ 1	≥ 1	✓	✗	✗	✗

MINTURN

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✗	✗
≥ 1	≥ 1	✗	✗	✓	✓
≥ 1	1	✗	✓	✓	✓
≥ 1	≥ 2	✗	✓	✗	✓
≥ 1	≥ 2	✗	✓	✓	✗
≥ 1	≥ 1	✓	✗	✗	✗

Contribution

Related Work

- study LIDARP with other objectives
- assume all requests can be served

Parameterized Algorithms

Parameters

- k := #vehicles
- c := capacity
- h := #stations
- t := max. time

determined by time windows

Complexity

LIDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓
≥ 1	≥ 1	✓	✗	✗	✗

MINTURN

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✗	✗
≥ 1	≥ 1	✗	✗	✓	✓
≥ 1	1	✗	✓	✓	✓
≥ 1	≥ 2	✗	✓	✗	✓
≥ 1	≥ 2	✗	✓	✓	✗
≥ 1	≥ 1	✓	✗	✗	✗

Contribution

Related Work

- study LIDARP with other objectives
- assume all requests can be served

Parameterized Algorithms

- FPT-algorithm for LIDARP and MINTURN

$$O^*((h^2 \cdot t^3 \cdot c \cdot k)^{2 \cdot t \cdot c \cdot k})$$

Parameters

- $k :=$ #vehicles
- $c :=$ capacity
- $h :=$ #stations
- $t :=$ max. time

Complexity

LIDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓
≥ 1	≥ 1	✓	✗	✗	✗

MINTURN

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✗	✗
≥ 1	≥ 1	✗	✗	✓	✓
≥ 1	1	✗	✓	✓	✓
≥ 1	≥ 2	✗	✓	✗	✓
≥ 1	≥ 2	✗	✓	✓	✗
≥ 1	≥ 1	✓	✗	✗	✗

Contribution

Related Work

- study LIDARP with other objectives
- assume all requests can be served

Parameterized Algorithms

- FPT-algorithm for LIDARP and MINTURN

$$O^*((h^2 \cdot t^3 \cdot c \cdot k)^{2 \cdot t \cdot c \cdot k})$$

Problem: without time windows $t = \infty$

Parameters

- $k :=$ #vehicles
- $c :=$ capacity
- $h :=$ #stations
- $t :=$ max. time

Complexity

LIDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓
≥ 1	≥ 1	✓	✗	✗	✗

MINTURN

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✗	✗
≥ 1	≥ 1	✗	✗	✓	✓
≥ 1	1	✗	✓	✓	✓
≥ 1	≥ 2	✗	✓	✗	✓
≥ 1	≥ 2	✗	✓	✓	✗
≥ 1	≥ 1	✓	✗	✗	✗

Contribution

Related Work

- study LIDARP with other objectives
- assume all requests can be served

Parameterized Algorithms

- FPT-algorithm for LIDARP and MINTURN

$$O^*((h^2 \cdot t^3 \cdot c \cdot k)^{2 \cdot t \cdot c \cdot k})$$

- XP-algorithm for MINTURN without time windows

$$O^*(n^{h^2} \cdot h^{4 \cdot c \cdot h})$$

Parameters

- $k :=$ #vehicles
- $c :=$ capacity
- $h :=$ #stations
- $t :=$ max. time

Complexity

LIDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓
≥ 1	≥ 1	✓	✗	✗	✗

MINTURN

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✗	✗
≥ 1	≥ 1	✗	✗	✓	✓
≥ 1	1	✗	✓	✓	✓
≥ 1	≥ 2	✗	✓	✗	✓
≥ 1	≥ 2	✗	✓	✓	✗
≥ 1	≥ 1	✓	✗	✗	✗

Contribution

Related Work

- study LIDARP with other objectives
- assume all requests can be served

Parameterized Algorithms

- FPT-algorithm for LIDARP and MINTURN

$$O^*((h^2 \cdot t^3 \cdot c \cdot k)^{2 \cdot t \cdot c \cdot k})$$

- XP-algorithm for MINTURN without time windows

$$O^*(n^{h^2} \cdot h^{4 \cdot c \cdot h})$$

$n :=$ #requests

Parameters

- $k :=$ #vehicles
- $c :=$ capacity
- $h :=$ #stations
- $t :=$ max. time

Complexity

LIDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓
≥ 1	≥ 1	✓	✗	✗	✗

MINTURN

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✗	✗
≥ 1	≥ 1	✗	✗	✓	✓
≥ 1	1	✗	✓	✓	✓
≥ 1	≥ 2	✗	✓	✗	✓
≥ 1	≥ 2	✗	✓	✓	✗
≥ 1	≥ 1	✓	✗	✗	✗

Contribution

Related Work

- study LIDARP with other objectives
- assume all requests can be served

Parameterized Algorithms

- FPT-algorithm for LIDARP and MINTURN

$$O^*((h^2 \cdot t^3 \cdot c \cdot k)^{2 \cdot t \cdot c \cdot k})$$

- XP-algorithm for MINTURN without time windows

$$O^*(n^{h^2} \cdot h^{4 \cdot c \cdot h})$$

Parameters

- $k :=$ #vehicles
- $c :=$ capacity
- $h :=$ #stations
- $t :=$ max. time

Complexity

LIDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓
≥ 1	≥ 1	✓	✗	✗	✗

MINTURN

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✗	✗
≥ 1	≥ 1	✗	✗	✓	✓
≥ 1	1	✗	✓	✓	✓
≥ 1	≥ 2	✗	✓	✗	✓
≥ 1	≥ 2	✗	✓	✓	✗
≥ 1	≥ 1	✓	✗	✗	✗

Contribution

Related Work

- study LIDARP with other objectives
- assume all requests can be served

Parameterized Algorithms

- FPT-algorithm for LIDARP and MINTURN

$$O^*((h^2 \cdot t^3 \cdot c \cdot k)^{2 \cdot t \cdot c \cdot k})$$

- XP-algorithm for MINTURN without time windows

$$O^*(n^{h^2} \cdot h^{4 \cdot c \cdot h})$$

Parameters

- $k :=$ #vehicles
- $c :=$ capacity
- $h :=$ #stations
- $t :=$ max. time

Complexity

LIDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓
≥ 1	≥ 1	✓	✗	✗	✗

MINTURN

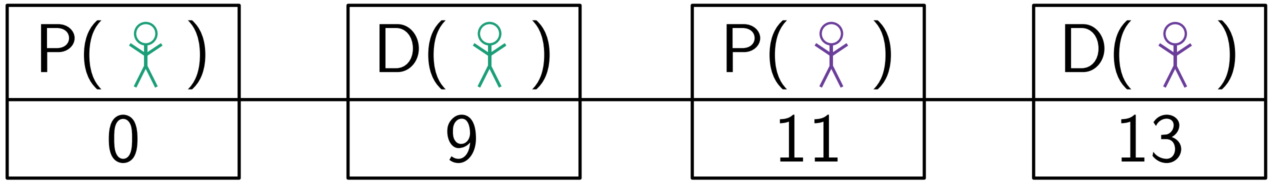
k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✗	✗
≥ 1	≥ 1	✗	✗	✓	✓
≥ 1	1	✗	✓	✓	✓
≥ 1	≥ 2	✗	✓	✗	✓
≥ 1	≥ 2	✗	✓	✓	✗
≥ 1	≥ 1	✓	✗	✗	✗

Easy liDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓

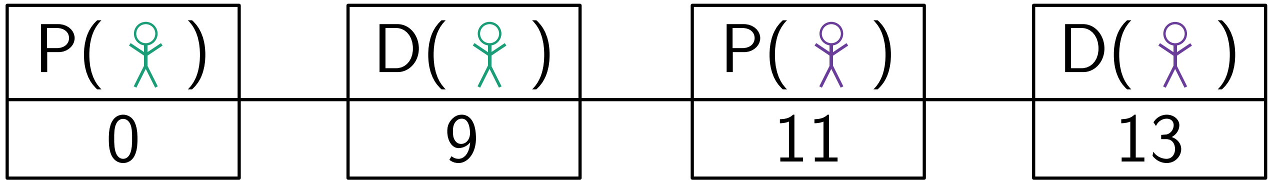
Easy LIDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓



Easy liDARP

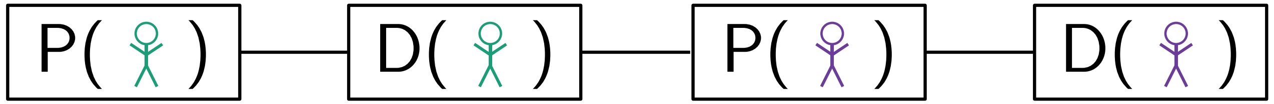
k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓



Route: tour without timestamps

Easy liDARP

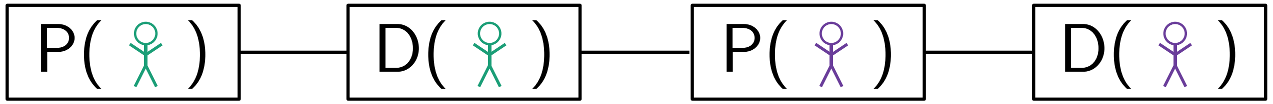
k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓



Route: tour without timestamps

Easy liDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓



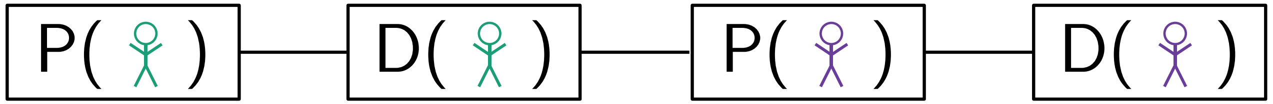
Route: tour without timestamps

- route feasible if there is feasible corresponding tour

Easy liDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓

Lemma [Haugland & Ho 2010]
Testing feasibility of routes (and constructing feasible tours) is possible in polynomial time



Route: tour without timestamps

- route feasible if there is feasible corresponding tour

Easy liDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓

Lemma [Haugland & Ho 2010]
Testing feasibility of routes (and constructing feasible tours) is possible in polynomial time

Lemma
Joining two feasible routes yields feasible route if there are no time windows

Easy liDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓

Lemma [Haugland & Ho 2010]

Testing feasibility of routes (and constructing feasible tours) is possible in polynomial time

Lemma

Joining two feasible routes yields feasible route if there are no time windows

Easy liDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓

Lemma [Haugland & Ho 2010]
Testing feasibility of routes (and constructing feasible tours) is possible in polynomial time

Lemma
Joining two feasible routes yields feasible route if there are no time windows

Easy LIDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓

Lemma [Haugland & Ho 2010]
Testing feasibility of routes (and constructing feasible tours) is possible in polynomial time

Lemma
Joining two feasible routes yields feasible route if there are no time windows

Theorem
Without time windows a tour serving all request can be computed in polynomial time

Easy liDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓

Lemma [Haugland & Ho 2010]
Testing feasibility of routes (and constructing feasible tours) is possible in polynomial time

Lemma
Joining two feasible routes yields feasible route if there are no time windows

Theorem
Without time windows a tour serving all request can be computed in polynomial time

Proof:

Easy LIDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓

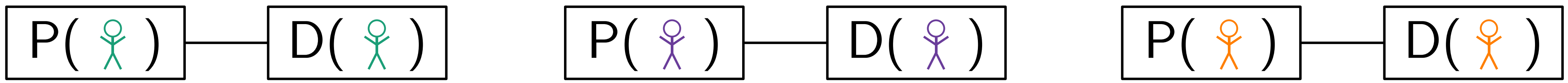
Lemma [Haugland & Ho 2010]
Testing feasibility of routes (and constructing feasible tours) is possible in polynomial time

Lemma
Joining two feasible routes yields feasible route if there are no time windows

Theorem
Without time windows a tour serving all request can be computed in polynomial time

Proof:

- serve each request in separate route (all feasible)



Easy LIDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓

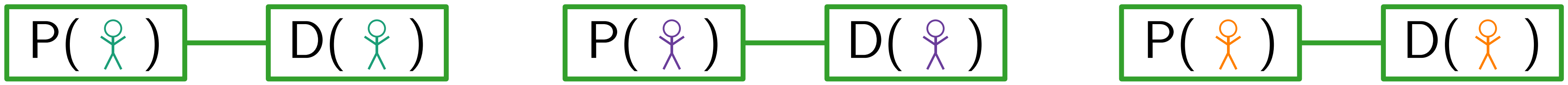
Lemma [Haugland & Ho 2010]
Testing feasibility of routes (and constructing feasible tours) is possible in polynomial time

Lemma
Joining two feasible routes yields feasible route if there are no time windows

Theorem
Without time windows a tour serving all request can be computed in polynomial time

Proof:

- serve each request in separate route (all feasible)



Easy liDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓

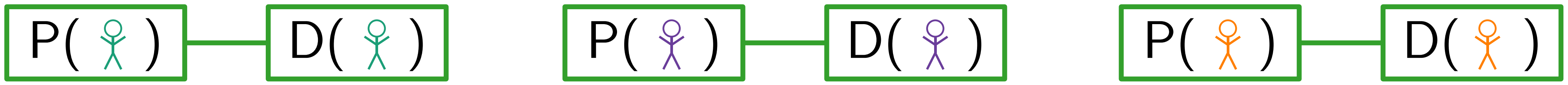
Lemma [Haugland & Ho 2010]
Testing feasibility of routes (and constructing feasible tours) is possible in polynomial time

Lemma
Joining two feasible routes yields feasible route if there are no time windows

Theorem
Without time windows a tour serving all request can be computed in polynomial time

Proof:

- serve each request in separate route (all feasible)
- join all routes into feasible route serving all requests



Easy LIDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓

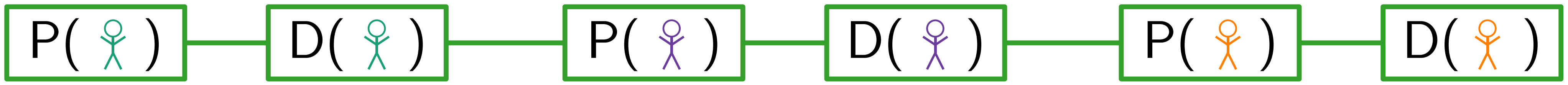
Lemma [Haugland & Ho 2010]
Testing feasibility of routes (and constructing feasible tours) is possible in polynomial time

Lemma
Joining two feasible routes yields feasible route if there are no time windows

Theorem
Without time windows a tour serving all request can be computed in polynomial time

Proof:

- serve each request in separate route (all feasible)
- join all routes into feasible route serving all requests



Easy liDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓

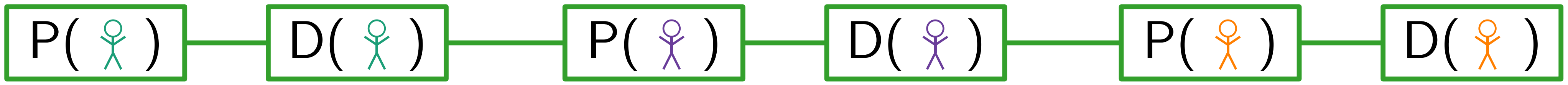
Lemma [Haugland & Ho 2010]
Testing feasibility of routes (and constructing feasible tours) is possible in polynomial time

Lemma
Joining two feasible routes yields feasible route if there are no time windows

Theorem
Without time windows a tour serving all request can be computed in polynomial time

Proof:

- serve each request in separate route (all feasible)
- join all routes into feasible route serving all requests
- compute corresponding feasible tour



Easy liDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓

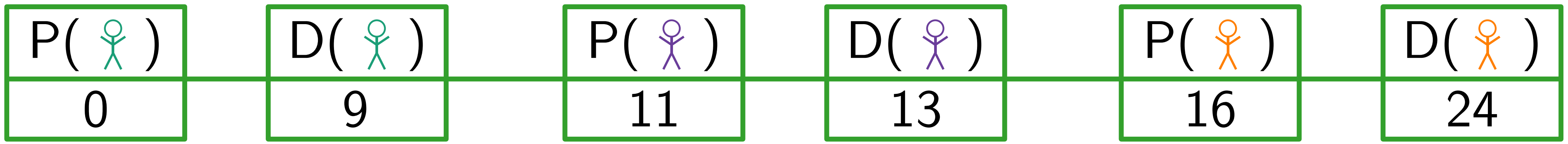
Lemma [Haugland & Ho 2010]
Testing feasibility of routes (and constructing feasible tours) is possible in polynomial time

Lemma
Joining two feasible routes yields feasible route if there are no time windows

Theorem
Without time windows a tour serving all request can be computed in polynomial time

Proof:

- serve each request in separate route (all feasible)
- join all routes into feasible route serving all requests
- compute corresponding feasible tour



Hard MinTurn

liDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓
≥ 1	≥ 1	✓	✗	✗	✗

MinTurn

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✗	✗
≥ 1	≥ 1	✗	✗	✓	✓
≥ 1	1	✗	✓	✓	✓
≥ 1	≥ 2	✗	✓	✗	✓
≥ 1	≥ 2	✗	✓	✓	✗
≥ 1	≥ 1	✓	✗	✗	✗

Hard MinTurn

liDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓
≥ 1	≥ 1	✓	✗	✗	✗

MinTurn

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✗	✗
≥ 1	≥ 1	✗	✗	✓	✓
≥ 1	1	✗	✓	✓	✓
≥ 1	≥ 2	✗	✓	✗	✓
≥ 1	≥ 2	✗	✓	✓	✗
≥ 1	≥ 1	✓	✗	✗	✗

Hard MinTurn

liDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓
≥ 1	≥ 1	✓	✗	✗	✗

MinTurn

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✗	✗
≥ 1	≥ 1	✗	✗	✓	✓
≥ 1	1	✗	✓	✓	✓
≥ 1	≥ 2	✗	✓	✗	✓
$= 1$	$= 2$	✗	✓	✓	✗
≥ 1	≥ 1	✓	✗	✗	✗

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗

Hard MinTurn

k	c	TW	SP	ST	SC
$=1$	$=2$	✗	✓	✓	✗

General Idea

Reduction from 3-Partition

Hard MinTurn

k	c	TW	SP	ST	SC
$=1$	$=2$	✗	✓	✓	✗

General Idea

Reduction from 3-Partition

Definition (3-Partition):

Given: Multiset S of $3m$ positive integers with $\sum_{s\in S} = mT$

Question: Is there a partition of S into m disjoint subsets $S_1, \ldots, S_m$ s.t. each sums up to T?

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗

General Idea
Reduction from 3-Partition

$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

Definition (3-Partition):
Given: Multiset S of $3m$ positive integers with $\sum_{s \in S} = mT$
Question: Is there a partition of S into m disjoint subsets $S_1, \dots, S_m$ s.t. each sums up to T?

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗

General Idea

Reduction from 3-Partition

Definition (3-Partition):

Given: Multiset S of $3m$ positive integers with $\sum_{s \in S} = mT$

Question: Is there a partition of S into m disjoint subsets $S_1, \ldots, S_m$ s.t. each sums up to T?

$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

$\{1, 6, 8\}, \{3, 7, 5\}$

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗

General Idea

Reduction from 3-Partition

Definition (3-Partition):

Given: Multiset S of $3m$ positive integers with $\sum_{s \in S} = mT$

Question: Is there a partition of S into m disjoint subsets $S_1, \dots, S_m$ s.t. each sums up to T?

Theorem [Garey & Johnson 1979]

3-Partition is **strongly** $\mathcal{NP}$-hard

$S = \{1, 6, 8, 3, 7, 5\}$ $m = 2$, $T = 15$
$\{1, 6, 8\}, \{3, 7, 5\}$

Hard MinTurn

k	c	TW	SP	ST	SC
$=1$	$=2$	✗	✓	✓	✗

General Idea

Given 3-Partition-Instance S:

$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2,\ T = 15$

Hard MinTurn

k	c	TW	SP	ST	SC
$=1$	$=2$	✗	✓	✓	✗

General Idea

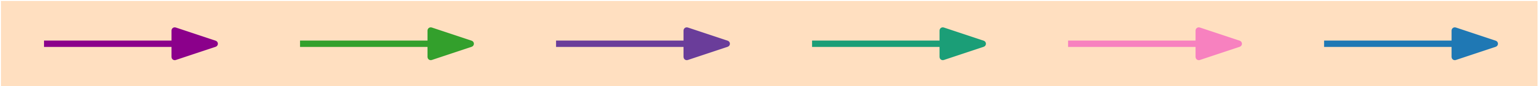
Given 3-Partition-Instance S:

- for each value s_i there is a *value request* v_i

$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗



General Idea

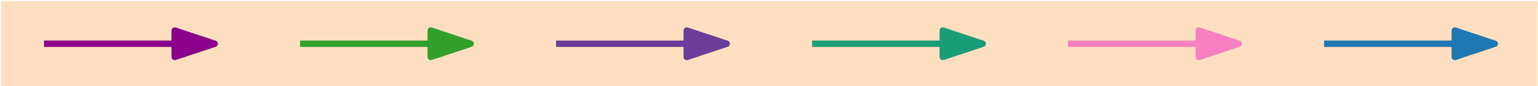
Given 3-Partition-Instance S:

- for each value s_i there is a *value request* v_i

$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2, T = 15$

Hard MinTurn

k	c	TW	SP	ST	SC
$=1$	$=2$	✗	✓	✓	✗



General Idea

Given 3-Partition-Instance S:

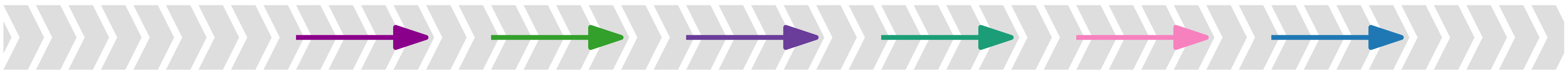
- for each value s_i there is a *value request* v_i

$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

Goal: partition of values corresponds to grouping of value requests into m *ascending* subroutes

Hard $\textsc{MinTurn}$

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗



General Idea

Given 3-$\textsc{Partition}$-Instance S:

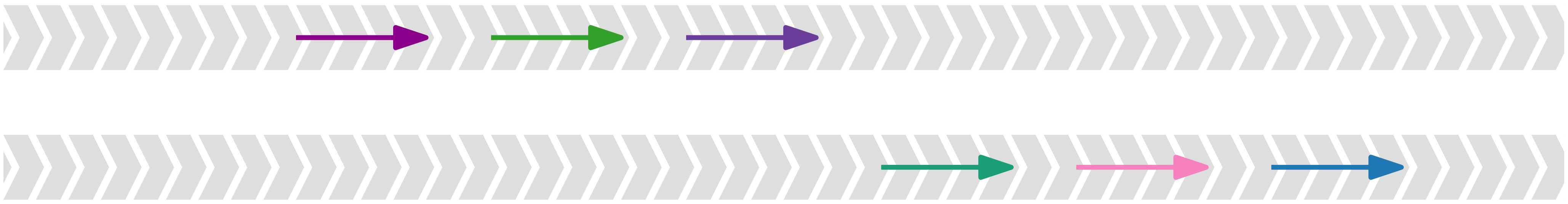
- for each value s_i there is a *value request* v_i

$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

Goal: partition of values corresponds to grouping of value requests into m *ascending* subroutes

Hard MinTurn

k	c	TW	SP	ST	SC
$=1$	$=2$	✗	✓	✓	✗



General Idea

Given 3-Partition-Instance S:

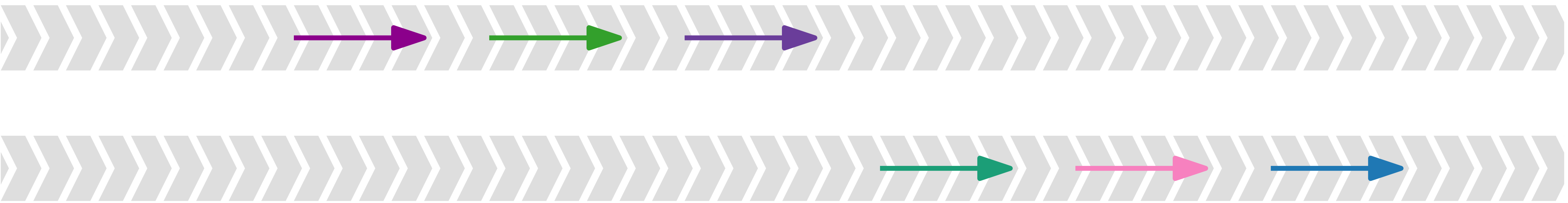
- for each value s_i there is a *value request* v_i

$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

Goal: partition of values corresponds to grouping of value requests into m *ascending* subroutes

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗



$\{1, 6, 8\}$

$\{3, 7, 5\}$

General Idea

Given 3-Partition-Instance S:

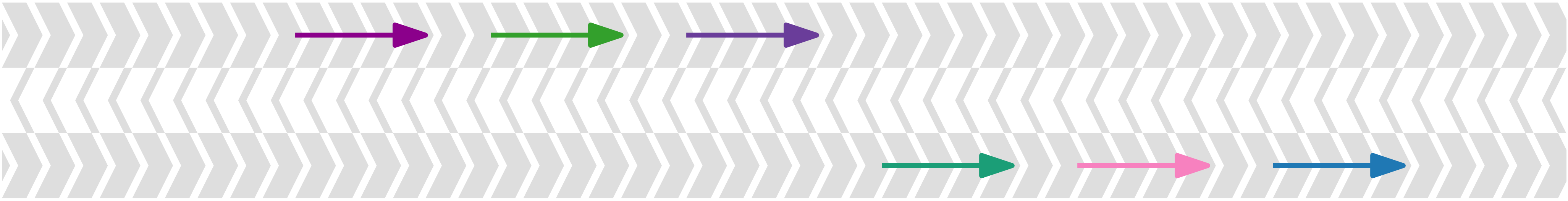
- for each value s_i there is a *value request* v_i

$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

Goal: partition of values corresponds to grouping of value requests into *m ascending* subroutes

Hard MinTurn

k	c	TW	SP	ST	SC
$=1$	$=2$	✗	✓	✓	✗



$\{1, 6, 8\}$

$\{3, 7, 5\}$

$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

General Idea

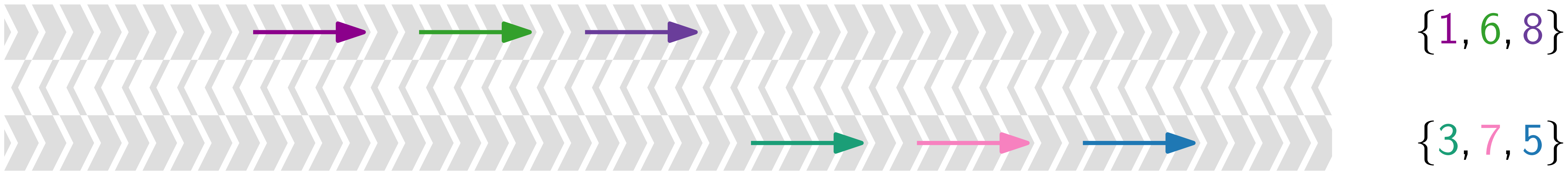
Given 3-Partition-Instance S:

- for each value s_i there is a *value request* v_i

Goal: partition of values corresponds to grouping of value requests into m *ascending* subroutes

Hard MinTurn

k	c	TW	SP	ST	SC
$=1$	$=2$	✗	✓	✓	✗



General Idea

Given 3-Partition-Instance S:

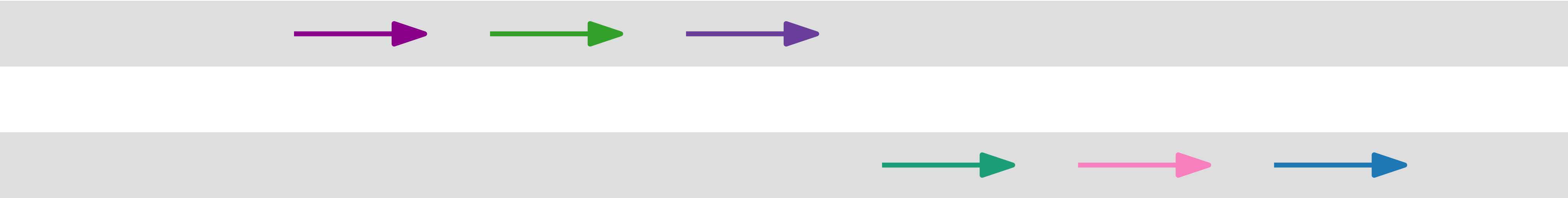
- for each value s_i there is a *value request* v_i

$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

Goal: partition of values corresponds to grouping of value requests into m *ascending* subroutes
$\Leftrightarrow \tau = 2m - 1$

Hard MinTurn

k	c	TW	SP	ST	SC
$=1$	$=2$	✗	✓	✓	✗



$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

General Idea

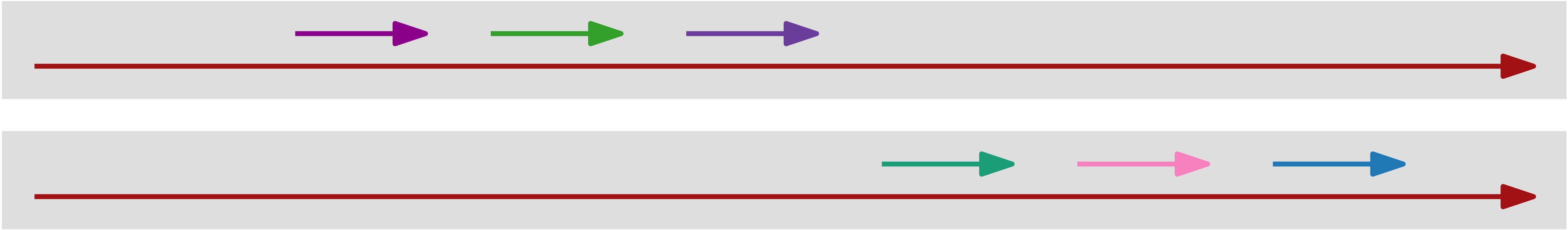
Given 3-Partition-Instance S:

- for each value s_i there is a *value request* v_i

Goal: partition of values corresponds to grouping of value requests into m *ascending* subroutes $\Leftrightarrow \tau = 2m - 1$

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗



$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

General Idea

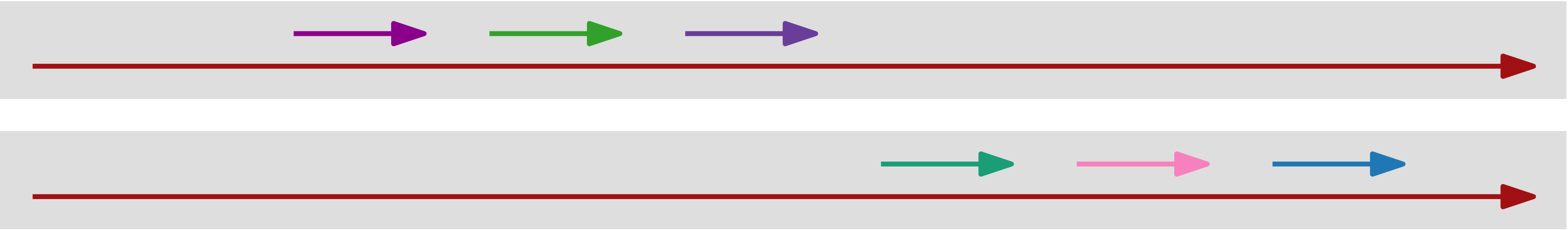
Given 3-Partition-Instance S:

- for each value s_i there is a *value request* v_i
- sum of values to T in each set is enforced by a *promise request* per set

Goal: partition of values corresponds to grouping of value requests into m *ascending* subroutes
$\Leftrightarrow \tau = 2m - 1$

Hard MinTurn

k	c	TW	SP	ST	SC
$=1$	$=2$	✗	✓	✓	✗



$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

General Idea

Given 3-Partition-Instance S:

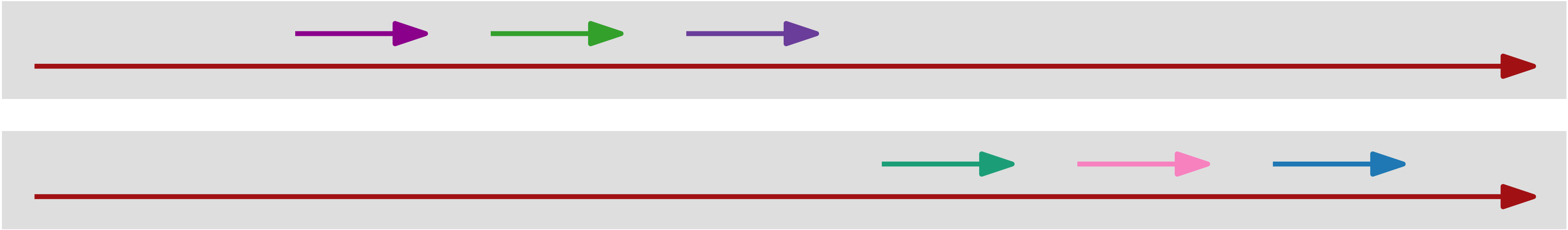
- for each value s_i there is a *value request* v_i
- sum of values to T in each set is enforced by a *promise request* per set

$\Delta = 2T$

Goal: partition of values corresponds to grouping of value requests into m *ascending* subroutes
$\Leftrightarrow \tau = 2m - 1$

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗



General Idea

Given 3-Partition-Instance S:

delay of $2s_i$

- for each value s_i there is a *value request* v_i
- sum of values to T in each set is enforced by a *promise request* per set

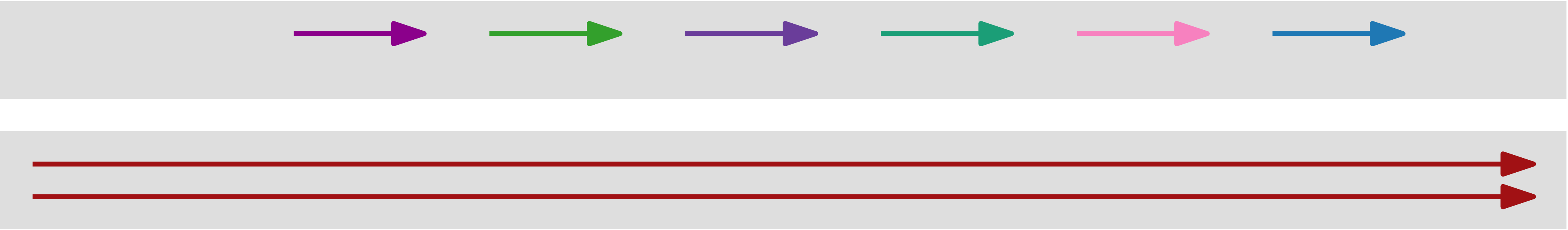
$\Delta = 2T$

$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2,\ T = 15$

Goal: partition of values corresponds to grouping of value requests into m *ascending* subroutes $\Leftrightarrow \tau = 2m - 1$

Hard MinTurn

k	c	TW	SP	ST	SC
$=1$	$=2$	✗	✓	✓	✗



$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

General Idea

Given 3-Partition-Instance S:

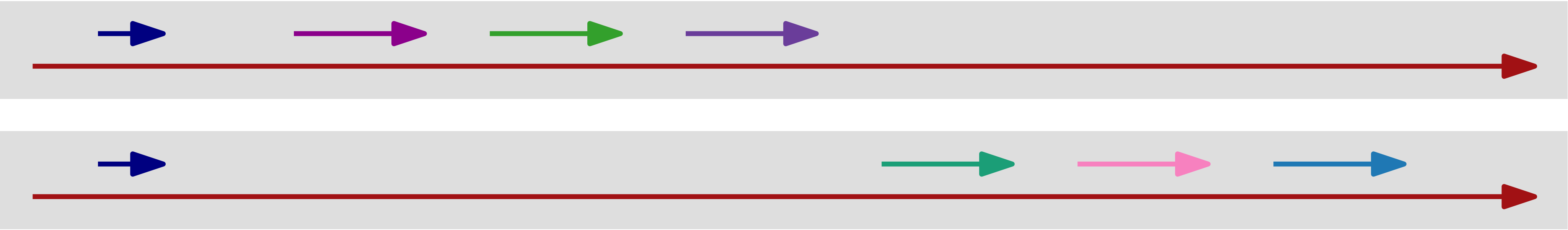
- for each value s_i there is a *value request* v_i
- sum of values to T in each set is enforced by a *promise request* per set

Problem: not always one promise request in each subroute

Goal: partition of values corresponds to
grouping of value requests into m *ascending* subroutes
$\Leftrightarrow \tau = 2m - 1$

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗



$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2, T = 15$

General Idea

Given 3-Partition-Instance S:

- for each value s_i there is a *value request* v_i
- sum of values to T in each set is enforced by a *promise request* per set
- one promise requests per subroute is enforced by m *filter requests*

Goal: partition of values corresponds to
grouping of value requests into m *ascending* subroutes
$\Leftrightarrow \tau = 2m - 1$

Hard MinTurn

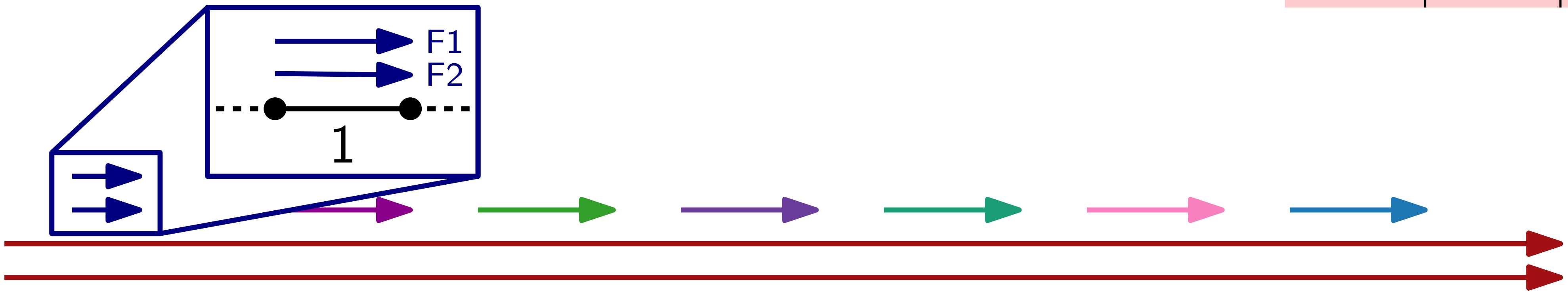
k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗

Set service time $t_s = 1$ and service promise $\rho < 2$

$$S = \{1, 6, 8, 3, 7, 5\}$$
$$m = 2,\ T = 15$$

Hard $\textsc{MinTurn}$

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗



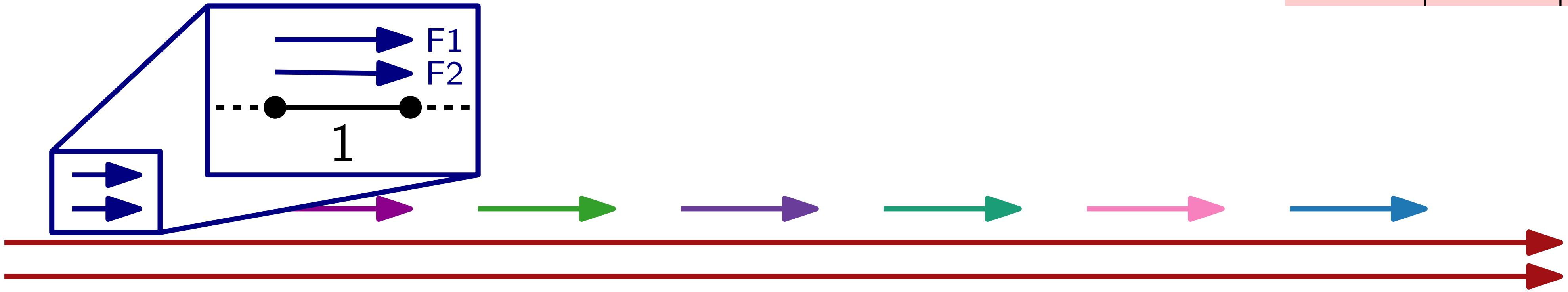
Set service time $t_s = 1$ and service promise $\rho < 2$

Filter Requests

$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗



$$S = \{1, 6, 8, 3, 7, 5\}$$
$$m = 2,\ T = 15$$

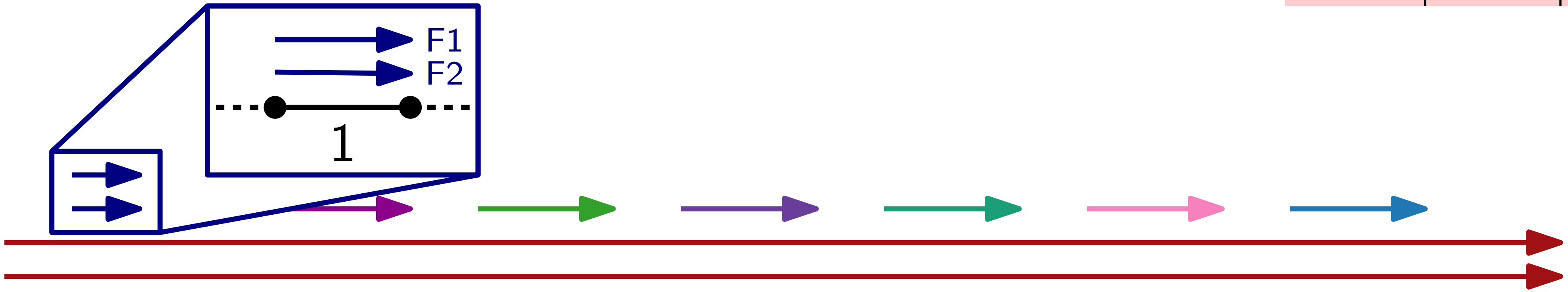
Set service time $t_s = 1$ and service promise $\rho < 2$

Filter Requests

- DTT $= 1 \Rightarrow$ MTT < 2

Hard MinTurn

k	c	TW	SP	ST	SC
$=1$	$=2$	✗	✓	✓	✗



$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

Set service time $t_s = 1$ and service promise $\rho < 2$

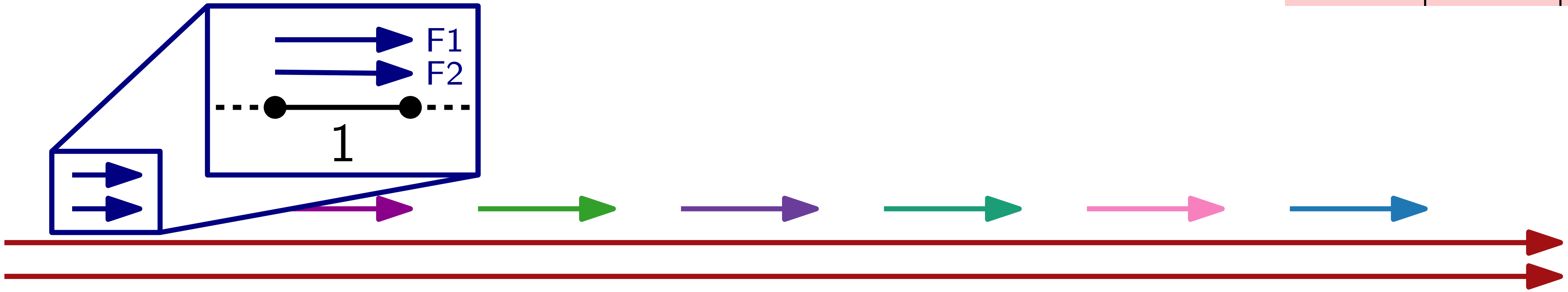
Filter Requests

- $\text{DTT} = 1 \Rightarrow \text{MTT} < 2$

P(F1)	P(F2)	——	D(F1)	D(F2)

Hard MinTurn

k	c	TW	SP	ST	SC
$=1$	$=2$	✗	✓	✓	✗



$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

Set service time $t_s = 1$ and service promise $\rho < 2$

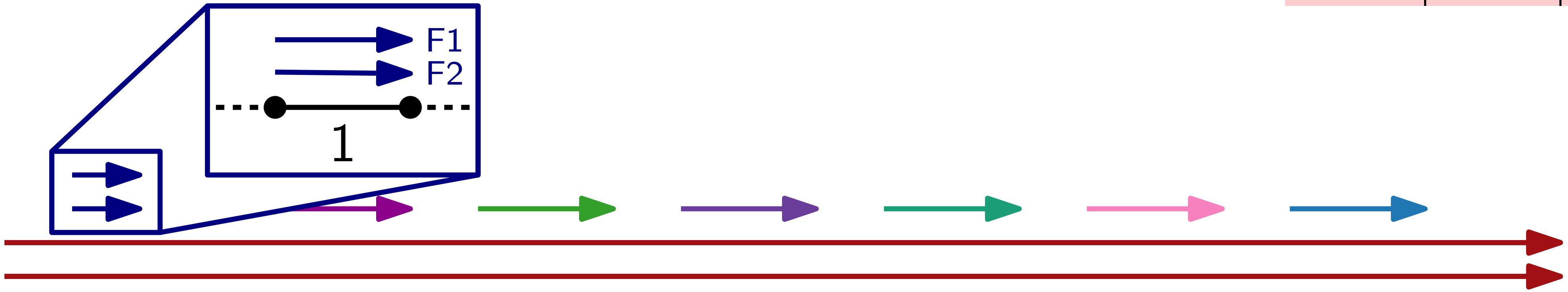
Filter Requests

- DTT $= 1 \Rightarrow$ MTT < 2

P(F1)	P(F2)	1	D(F1)	D(F2)
$2-3$	$3-4$		$5-6$	$6-7$

Hard MinTurn

k	c	TW	SP	ST	SC
$=1$	$=2$	✗	✓	✓	✗

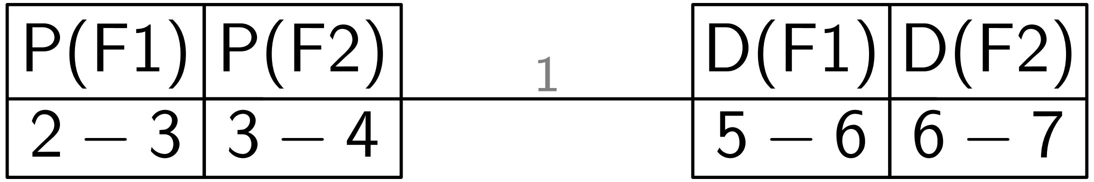


$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

Set service time $t_s = 1$ and service promise $\rho < 2$

Filter Requests

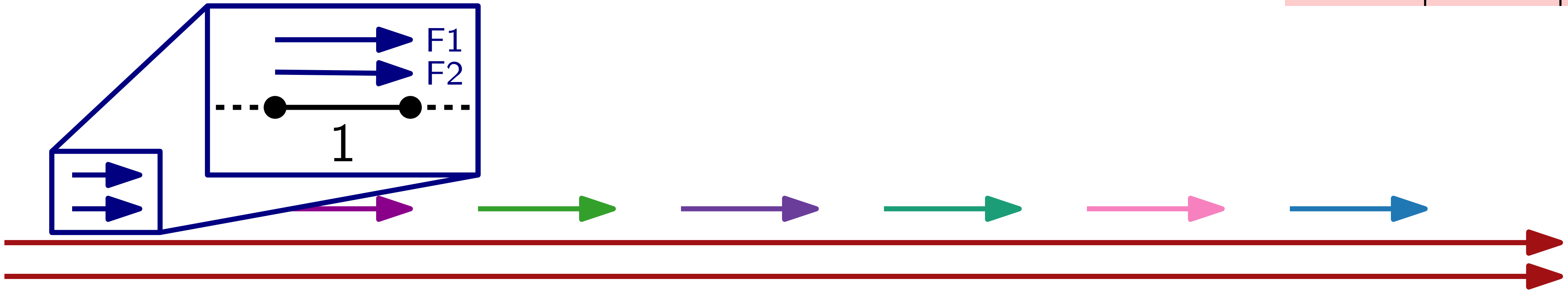
- DTT $= 1 \Rightarrow$ MTT < 2



TT(F1) $= 5 - 3 = 2 >$ MTT(F1)

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗

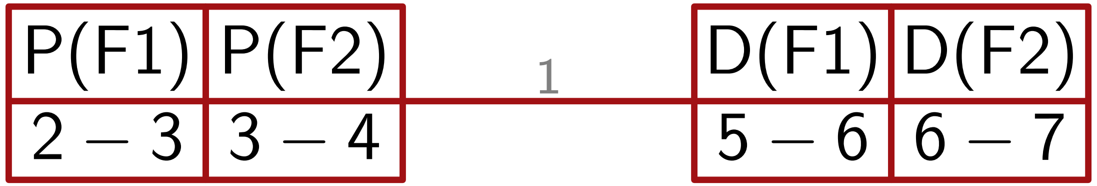


Set service time $t_s = 1$ and service promise $\rho < 2$

$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

Filter Requests

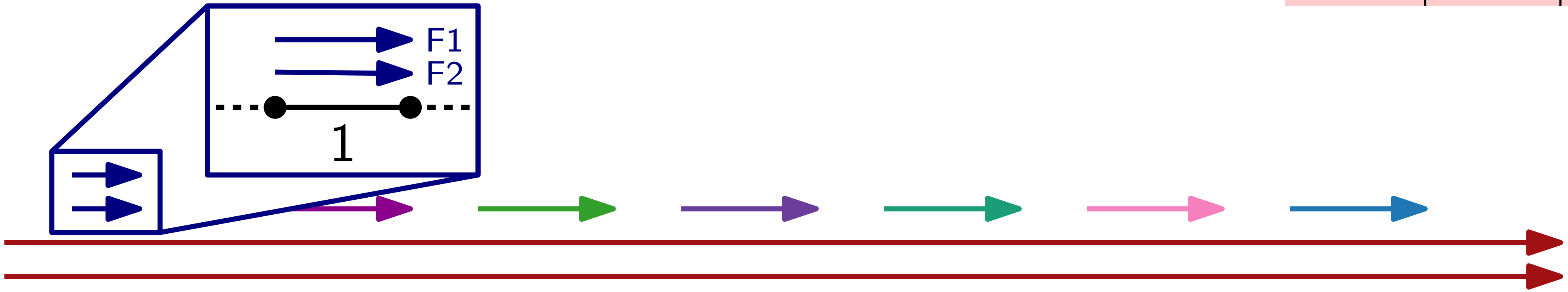
- $\text{DTT} = 1 \Rightarrow \text{MTT} < 2$



$\text{TT(F1)} = 5 - 3 = 2 > \text{MTT(F1)}$

Hard MinTurn

k	c	TW	SP	ST	SC
$=1$	$=2$	✗	✓	✓	✗

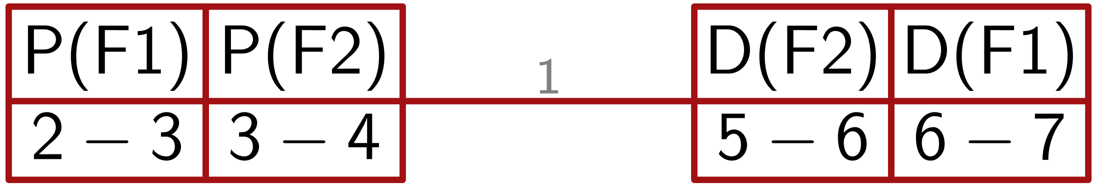


$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

Set service time $t_s = 1$ and service promise $\rho < 2$

Filter Requests

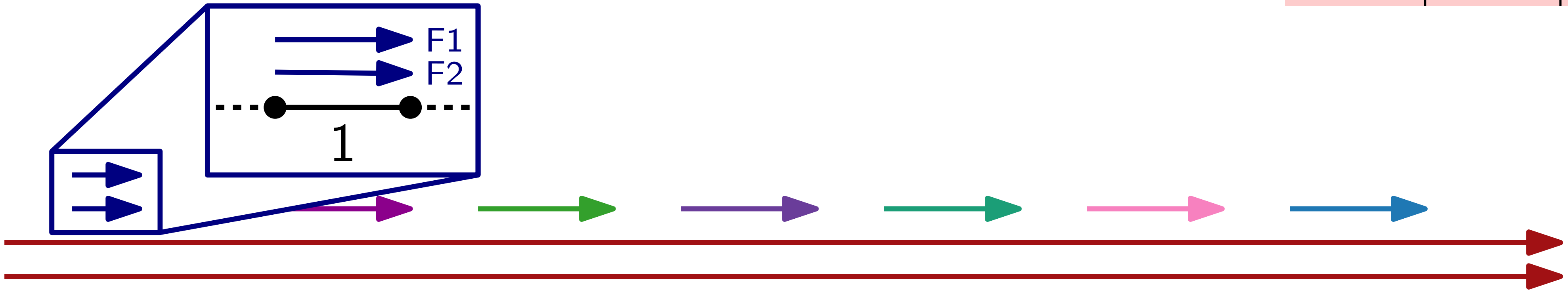
- DTT $= 1 \Rightarrow$ MTT < 2



TT(F1) $= 6 - 3 = 3 >$ MTT(F1)

Hard MinTurn

k	c	TW	SP	ST	SC
$=1$	$=2$	✗	✓	✓	✗



$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

Set service time $t_s = 1$ and service promise $\rho < 2$

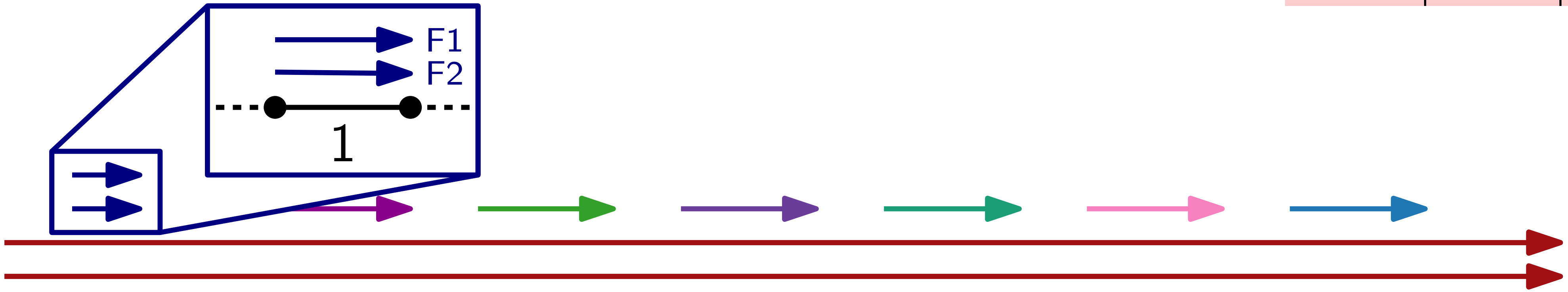
Filter Requests

- DTT $= 1 \Rightarrow$ MTT < 2

$\Rightarrow$ no two filter requests can be served in same subroute

Hard $\textsc{MinTurn}$

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗



$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

Set service time $t_s = 1$ and service promise $\rho < 2$

Filter Requests

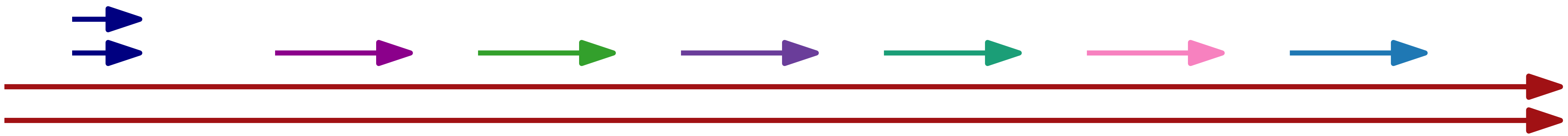
- DTT $= 1 \Rightarrow$ MTT < 2

$\Rightarrow$ no two filter requests can be served in same subroute

$\Rightarrow$ each of the m subroutes contains one filter and one promise request

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗



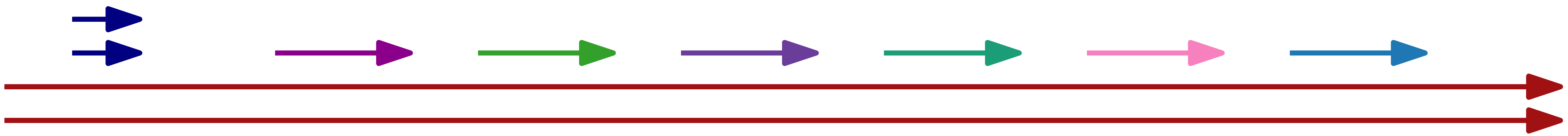
Set service time $t_s = 1$ and service promise $\rho < 2$

Value Requests

$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2, T = 15$

Hard MinTurn

k	c	TW	SP	ST	SC
$=1$	$=2$	✗	✓	✓	✗



$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

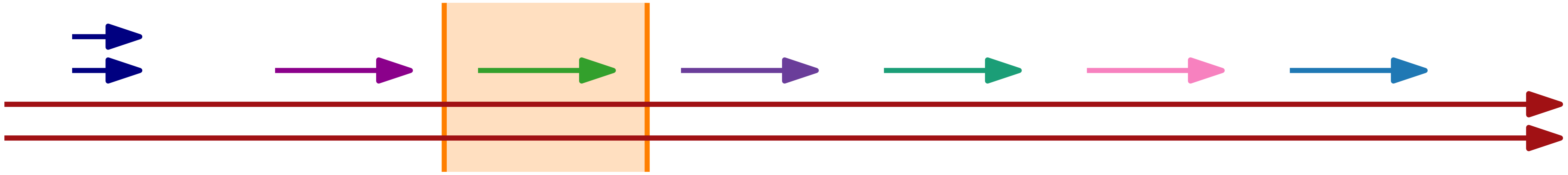
Set service time $t_s = 1$ and service promise $\rho < 2$

Value Requests

- value request v_i consists of s_i smaller requests

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗



Set service time $t_s = 1$ and service promise $\rho < 2$

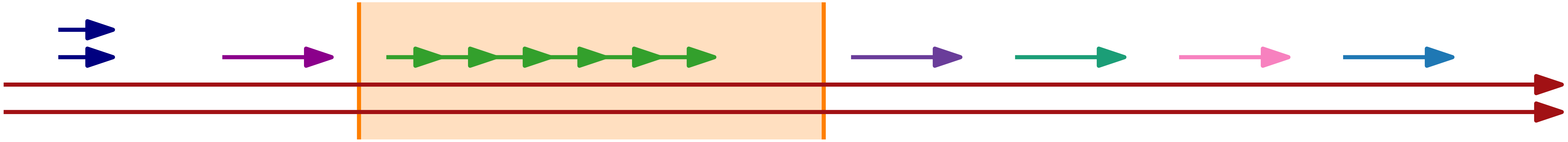
$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2, T = 15$

Value Requests

- value request v_i consists of s_i smaller requests

Hard MinTurn

k	c	TW	SP	ST	SC
$=1$	$=2$	✗	✓	✓	✗



Set service time $t_s = 1$ and service promise $\rho < 2$

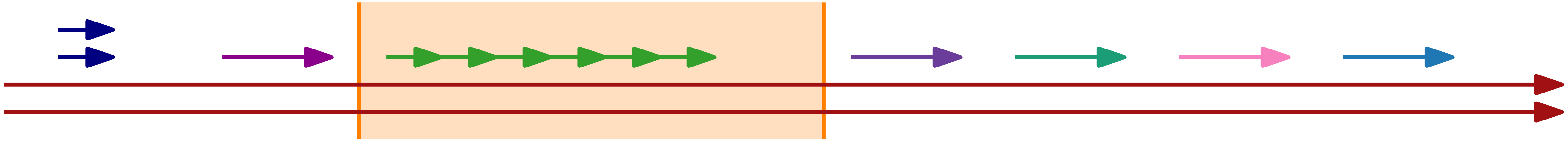
$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

Value Requests

- value request v_i consists of s_i smaller requests

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗



Set service time $t_s = 1$ and service promise $\rho < 2$

$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

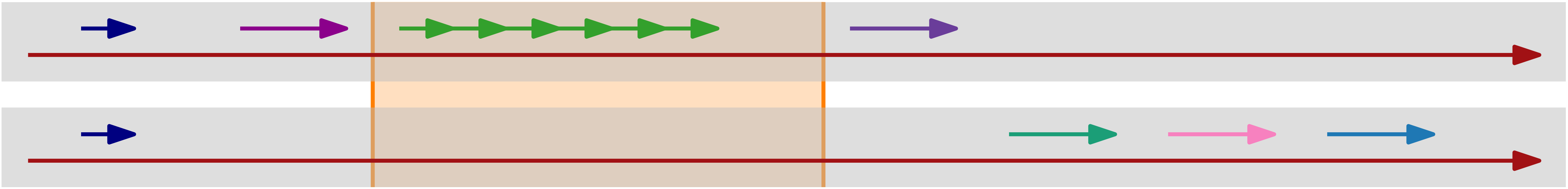
Value Requests

delay of $2s_i$

- value request v_i consists of s_i smaller requests

Hard $\textsc{MinTurn}$

k	c	TW	SP	ST	SC
$=1$	$=2$	✗	✓	✓	✗



Set service time $t_s = 1$ and service promise $\rho < 2$

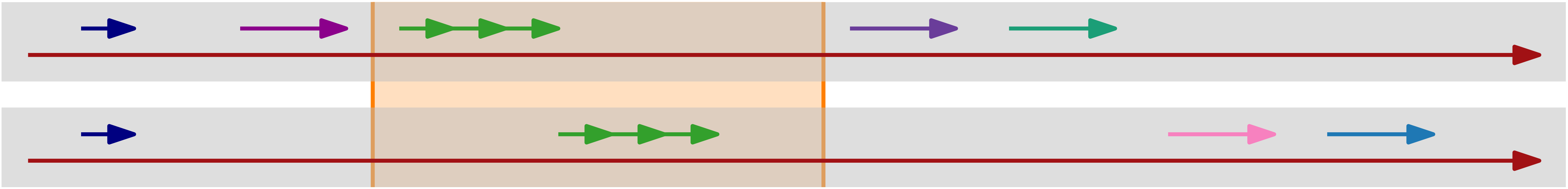
$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

Value Requests

- value request v_i consists of s_i smaller requests

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗



Set service time $t_s = 1$ and service promise $\rho < 2$

$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

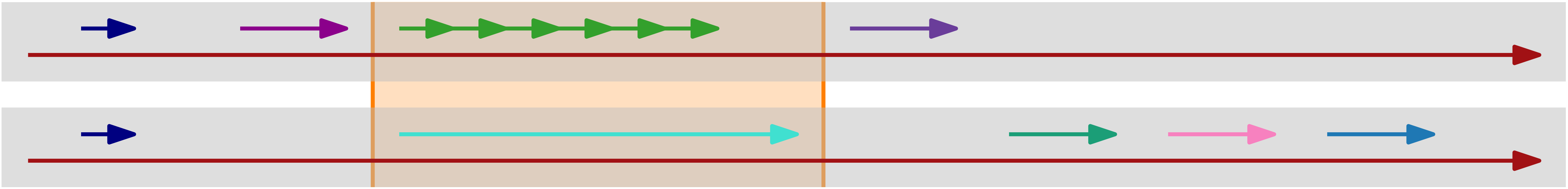
Value Requests

- value request v_i consists of s_i smaller requests

Problem: value request may be split between subroutes

Hard MinTurn

k	c	TW	SP	ST	SC
$=1$	$=2$	✗	✓	✓	✗



Set service time $t_s = 1$ and service promise $\rho < 2$

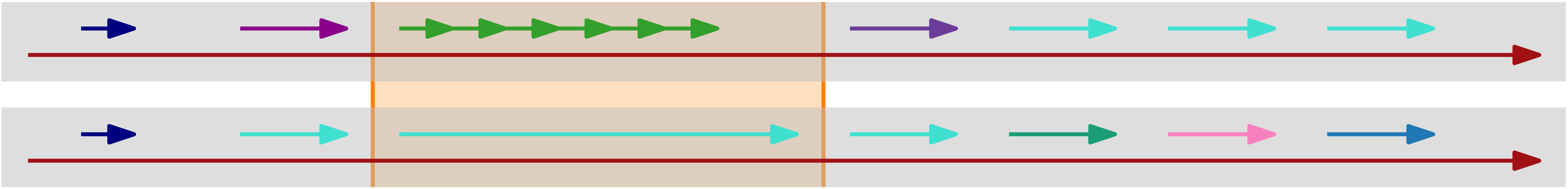
$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

Value Requests

- value request v_i consists of s_i smaller requests
- splitting of value requests is prevented by a plug request per subroute

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗



Set service time $t_s = 1$ and service promise $\rho < 2$

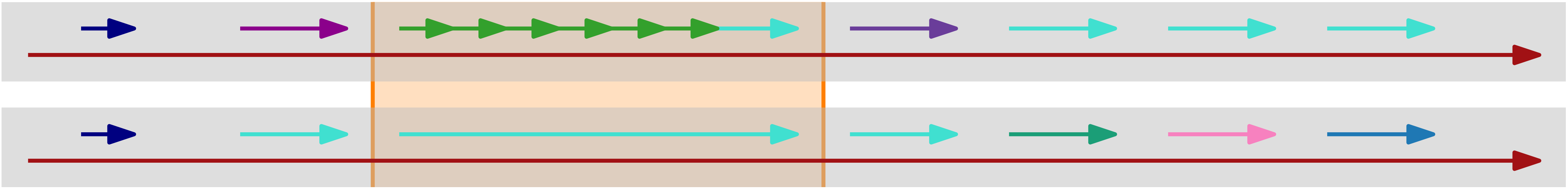
$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

Value Requests

- value request v_i consists of s_i smaller requests
- splitting of value requests is prevented by a plug request per subroute

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗



$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

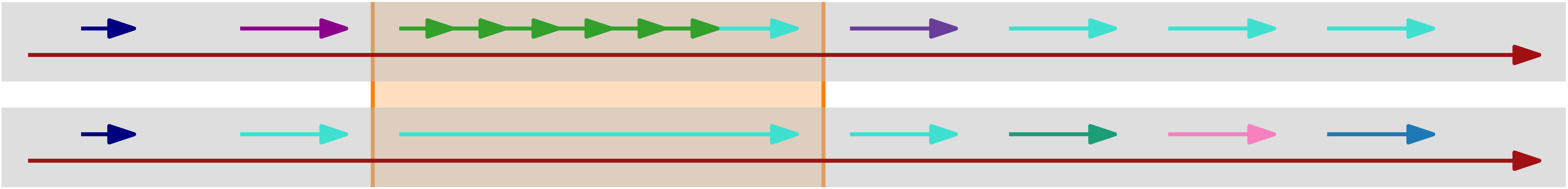
Set service time $t_s = 1$ and service promise $\rho < 2$

Value Requests

- value request v_i consists of s_i smaller requests
- splitting of value requests is prevented by a plug request per subroute

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗



Set service time $t_s = 1$ and service promise $\rho < 2$

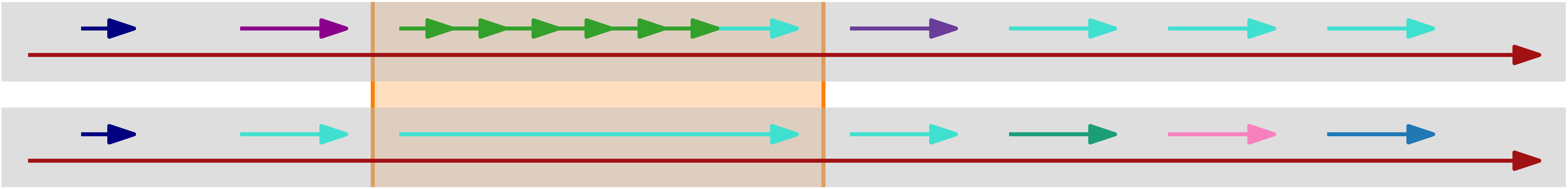
Each subroute should contain:

$S = \{1, 6, 8, 3, 7, 5\}$

$m = 2$, $T = 15$

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗



Set service time $t_s = 1$ and service promise $\rho < 2$

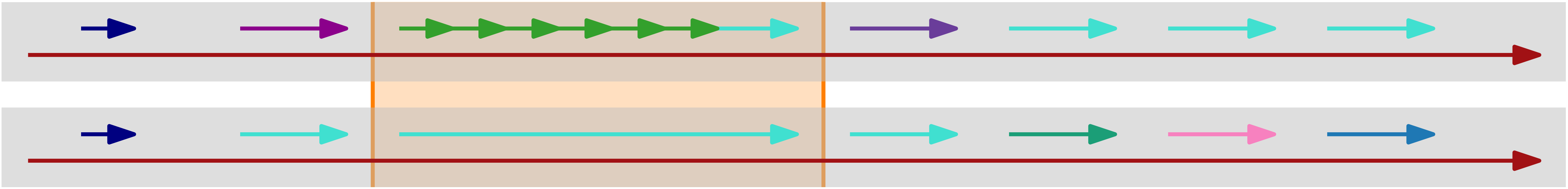
Each subroute should contain:

- 1 promise request

$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2,\ T = 15$

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗



$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

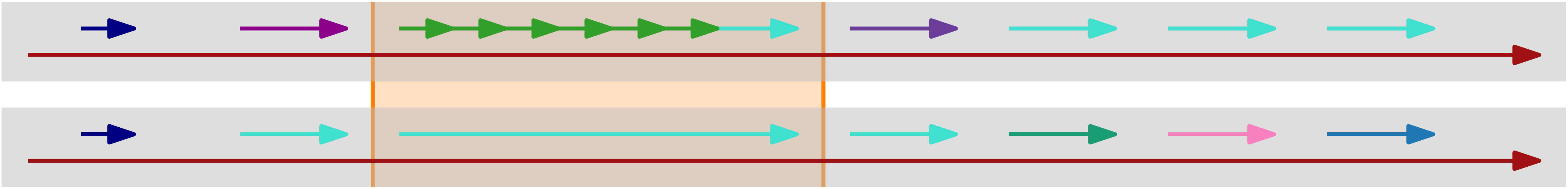
Set service time $t_s = 1$ and service promise $\rho < 2$

Each subroute should contain:

- 1 promise request
- 1 filter request

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗



Set service time $t_s = 1$ and service promise $\rho < 2$

Each subroute should contain:

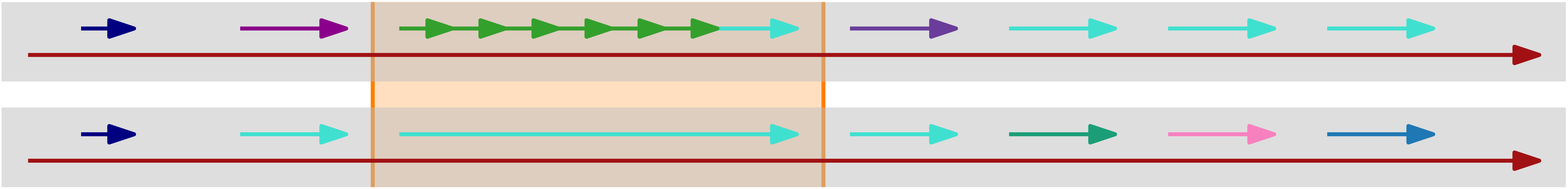
- 1 promise request
- 1 filter request
- T short value requests

$S = \{1, 6, 8, 3, 7, 5\}$

$m = 2, T = 15$

Hard MinTurn

k	c	TW	SP	ST	SC
$=1$	$=2$	✗	✓	✓	✗



$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

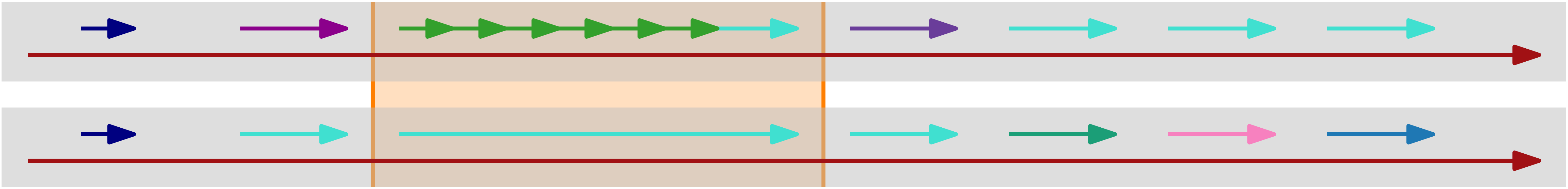
Set service time $t_s = 1$ and service promise $\rho < 2$

Each subroute should contain:

- 1 promise request
- 1 filter request
- T short value requests
- $3m$ plug requests

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗



Set service time $t_s = 1$ and service promise $\rho < 2$

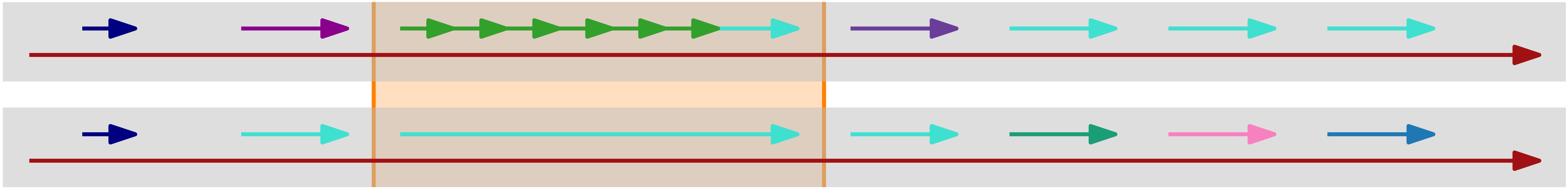
Each subroute should contain:

- 1 promise request ✓
- 1 filter request ✓
- T short value requests
- $3m$ plug requests

$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2, T = 15$

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗



Set service time $t_s = 1$ and service promise $\rho < 2$

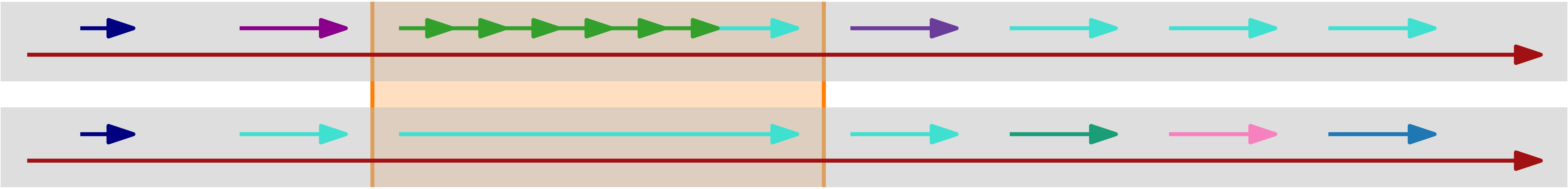
$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2, T = 15$

Each subroute should contain:

- 1 promise request ✓
- 1 filter request ✓
- T short value requests
- $3m$ plug requests ✓

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗



Set service time $t_s = 1$ and service promise $\rho < 2$

Each subroute should contain:

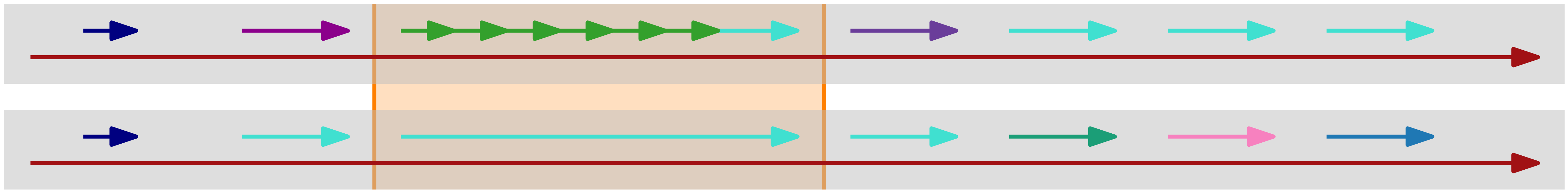
- 1 promise request ✓
- 1 filter request ✓
- T short value requests ⟵ enforced by service promise on promise requests
- $3m$ plug requests ✓

$S = \{1, 6, 8, 3, 7, 5\}$

$m = 2$, $T = 15$

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗



Set service time $t_s = 1$ and service promise $\rho < 2$

Each subroute should contain:

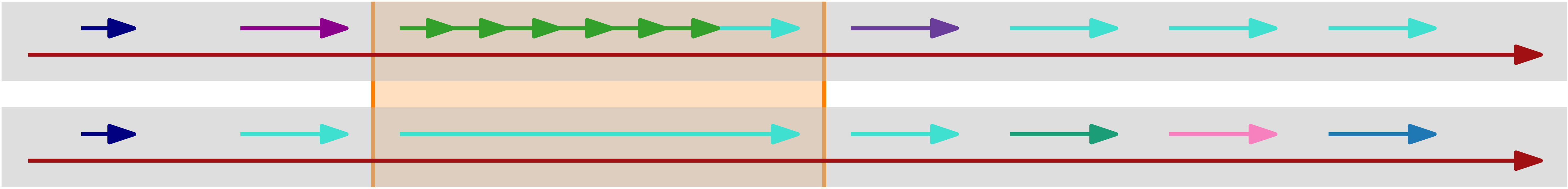
- 1 promise request
- 1 filter request
- T short value requests
- $3m$ plug requests

$S = \{1, 6, 8, 3, 7, 5\}$

$m = 2$, $T = 15$

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗



Set service time $t_s = 1$ and service promise $\rho < 2$

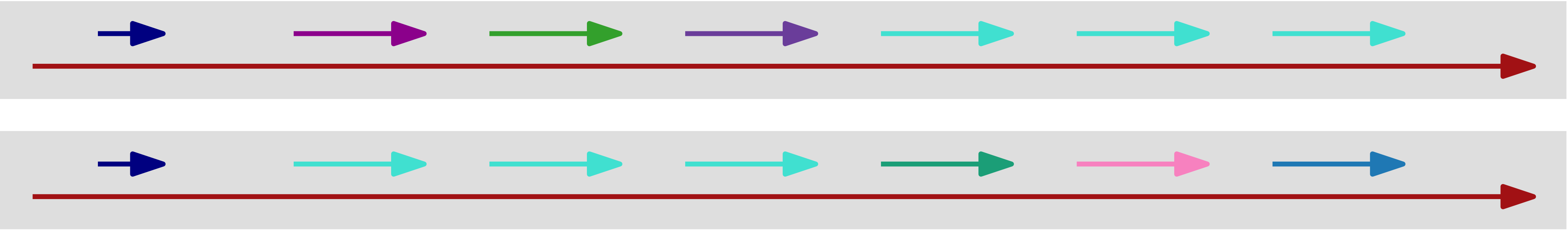
$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

Each subroute should contain:

$$\left.\begin{array}{l} \text{1 promise request} \\ \text{1 filter request} \\ T \text{ short value requests} \\ 3m \text{ plug requests} \end{array}\right\} \text{delay: } 2(1 + T + 3m)$$

Hard $\textsc{MinTurn}$

k	c	TW	SP	ST	SC
$=1$	$=2$	✗	✓	✓	✗



$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

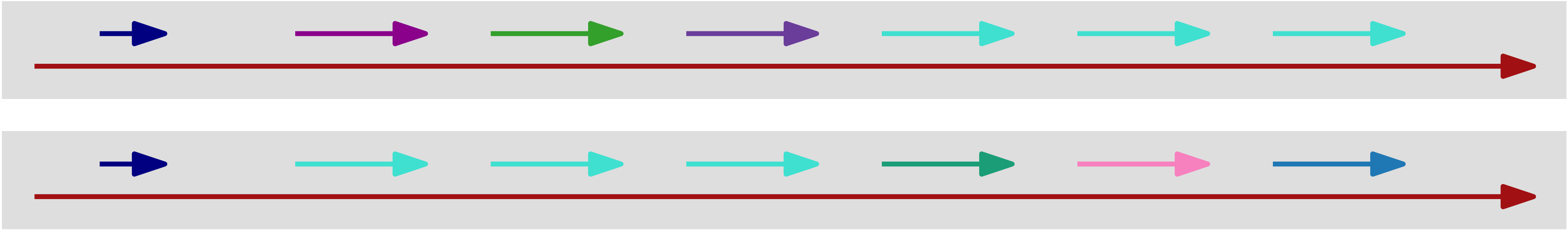
Set service time $t_s = 1$ and service promise $\rho < 2$

Promise Requests

- need $\Delta = 2(1 + T + 3m)$

Hard MinTurn

k	c	TW	SP	ST	SC
$=1$	$=2$	✗	✓	✓	✗



Set service time $t_s = 1$ and service promise $\rho < 2$

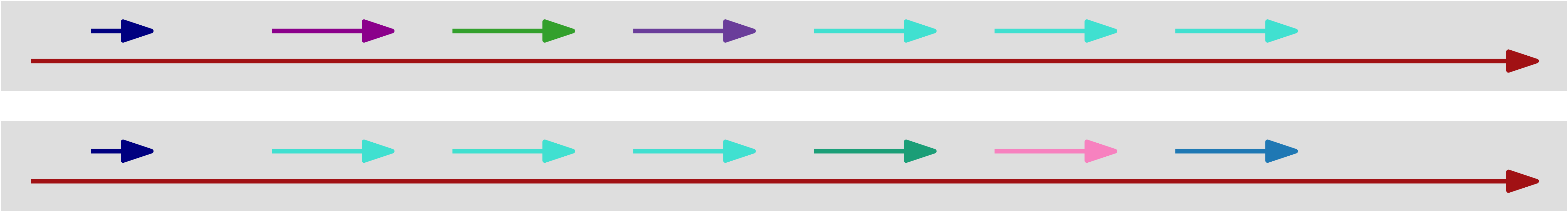
$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2,\ T = 15$

Promise Requests

- need $\Delta = 2(1 + T + 3m)$ $(\rho - 1) \cdot \mathrm{DTT}$

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗



Set service time $t_s = 1$ and service promise $\rho < 2$

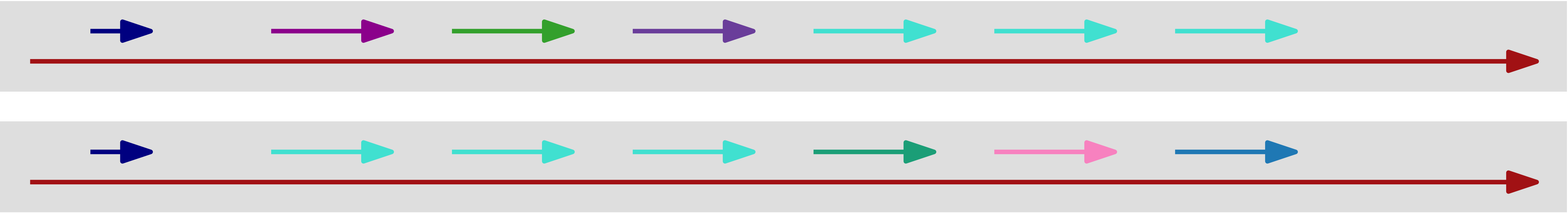
$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

Promise Requests

- need $\Delta = 2(1 + T + 3m)$ $(\rho - 1) \cdot \mathrm{DTT}$
- make DTT large enough such that $\rho < 2$

Hard MinTurn

k	c	TW	SP	ST	SC
$=1$	$=2$	✗	✓	✓	✗



Set service time $t_s = 1$ and service promise $\rho < 2$

$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2,\ T = 15$

Promise Requests

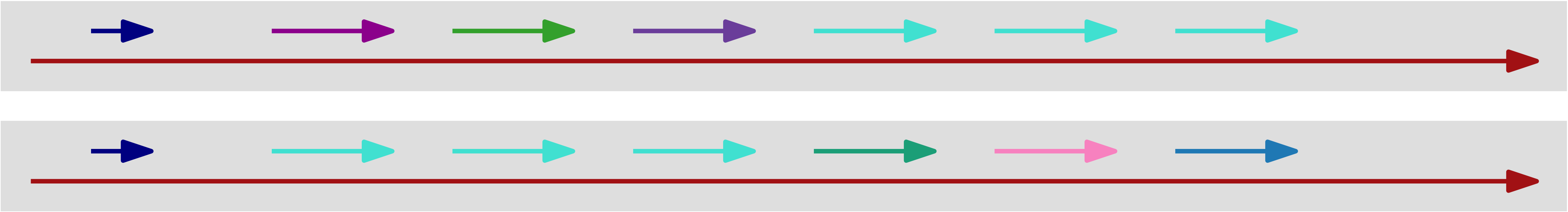
$(\rho - 1) \cdot \mathrm{DTT}$

- need $\Delta = 2(1 + T + 3m)$
- make DTT large enough such that $\rho < 2$

Goal: grouping of value requests into m subroutes corresponds to 3-partition of S ✓

Hard MinTurn

k	c	TW	SP	ST	SC
$= 1$	$= 2$	✗	✓	✓	✗



$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

Set service time $t_s = 1$ and service promise $\rho < 2$

Promise Requests

$(\rho - 1) \cdot \text{DTT}$

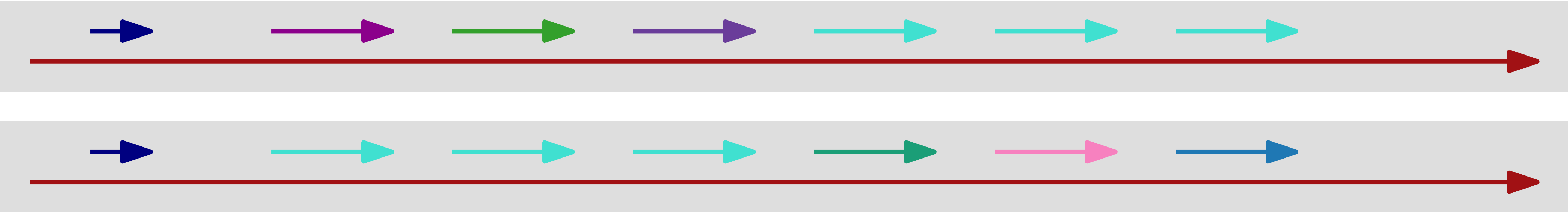
- need $\Delta = 2(1 + T + 3m)$
- make DTT large enough such that $\rho < 2$

Goal: grouping of value requests into m subroutes corresponds to 3-partition of S ✓

$\tau = 2m - 1 \Leftrightarrow S$ has 3-partition

Hard MinTurn

k	c	TW	SP	ST	SC
$=1$	$=2$	✗	✓	✓	✗



Set service time $t_s = 1$ and service promise $\rho < 2$

$S = \{1, 6, 8, 3, 7, 5\}$
$m = 2$, $T = 15$

Promise Requests

$(\rho - 1) \cdot \mathrm{DTT}$

- need $\Delta = 2(1 + T + 3m)$
- make DTT large enough such that $\rho < 2$

Goal: grouping of value requests into m subroutes corresponds to 3-partition of S ✓

$\tau = 2m - 1 \Leftrightarrow S$ has 3-partition

Theorem
MinTurn with service promise & service time is $\mathcal{NP}$-hard

Conclusion

Parameterized Algorithms

- FPT-algorithm for liDARP and MinTurn

  $$O^*((h^2 \cdot t^3 \cdot c \cdot k)^{2 \cdot t \cdot c \cdot k})$$

- XP-algorithm for MinTurn without time windows

  $$O^*(n^{h^2} \cdot h^{4 \cdot c \cdot h})$$

Complexity

liDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓
≥ 1	≥ 1	✓	✗	✗	✗

MinTurn

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✗	✗
≥ 1	≥ 1	✗	✗	✓	✓
≥ 1	1	✗	✓	✓	✓
≥ 1	≥ 2	✗	✓	✗	✓
≥ 1	≥ 2	✗	✓	✓	✗
≥ 1	≥ 1	✓	✗	✗	✗

Conclusion

Parameterized Algorithms

- FPT-algorithm for liDARP and MinTurn

  $$O^*((h^2 \cdot t^3 \cdot c \cdot k)^{2 \cdot t \cdot c \cdot k})$$

- XP-algorithm for MinTurn without time windows

  $$O^*(n^{h^2} \cdot h^{4 \cdot c \cdot h})$$

Open Problems

- Can the parameterized algorithms be improved? Is there an FPT algorithm for MinTurn without time windows?
- What about heuristics / approximation algorithms for MinTurn?

Complexity

liDARP

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✓	✓
≥ 1	≥ 1	✓	✗	✗	✗

MinTurn

k	c	TW	SP	ST	SC
≥ 1	≥ 1	✗	✓	✗	✗
≥ 1	≥ 1	✗	✗	✓	✓
≥ 1	1	✗	✓	✓	✓
≥ 1	≥ 2	✗	✓	✗	✓
≥ 1	≥ 2	✗	✓	✓	✗
≥ 1	≥ 1	✓	✗	✗	✗