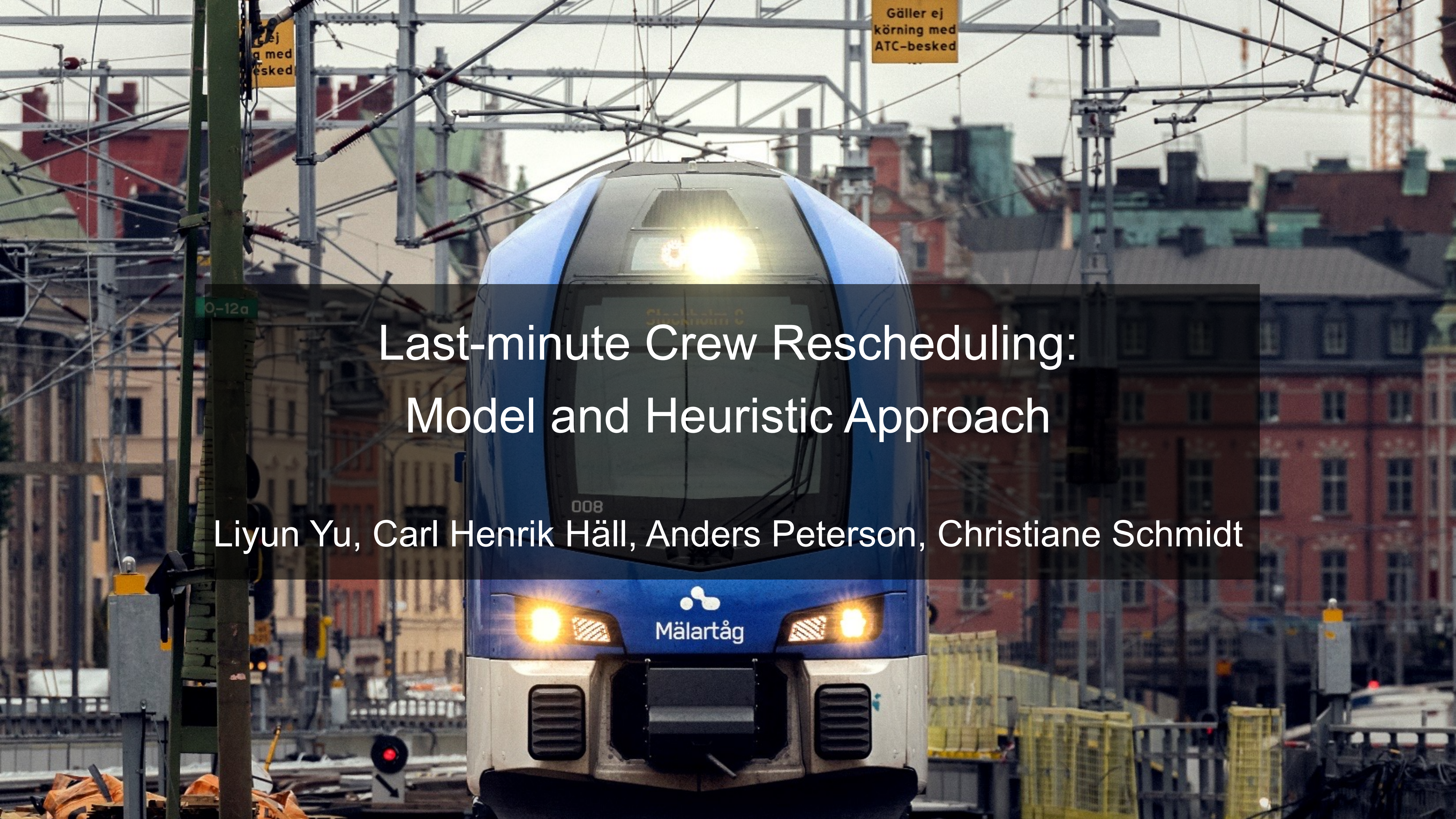
Last-minute Crew Rescheduling: Model and Heuristic Approach

Liyun Yu, Carl Henrik Häll, Anders Peterson, Christiane Schmidt



- Introduction
- Problem Description
- Methodology
- Case Study

Introduction

Introduction

Infrastructure failure

Bad weather

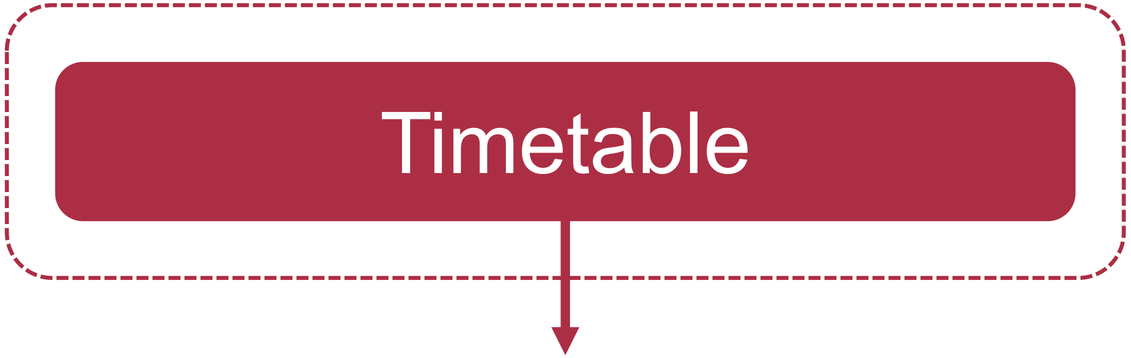
Rolling stock breakdown

Blocks on rail track

Infeasible vehicle schedule

Employee shortage

Introduction



Timetable

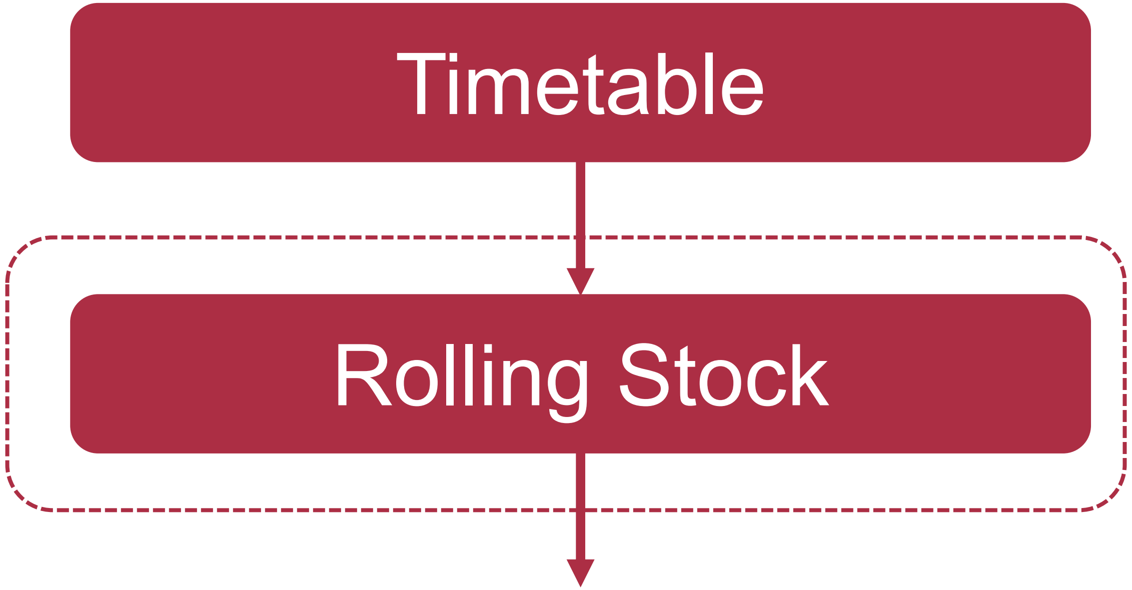
Infrastructure failure

Bad weather

Rolling stock breakdown

Blocks on rail track

Introduction



Infrastructure failure

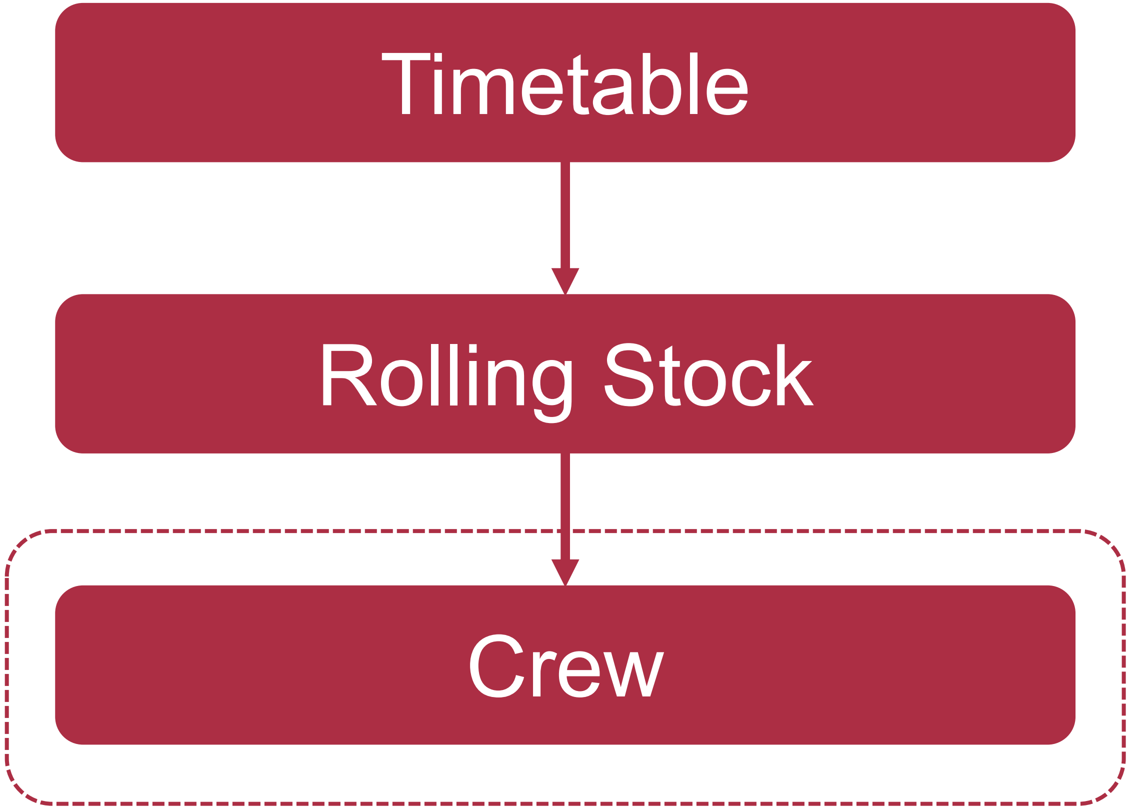
Bad weather

Rolling stock breakdown

Blocks on rail track

Infeasible vehicle schedule

Introduction



Infrastructure failure

Bad weather

Rolling stock breakdown

Blocks on rail track

Infeasible vehicle schedule

Employee shortage

Introduction

Employee shortage

Introduction

Driver

~~Employee~~ shortage

Introduction

Driver shortage

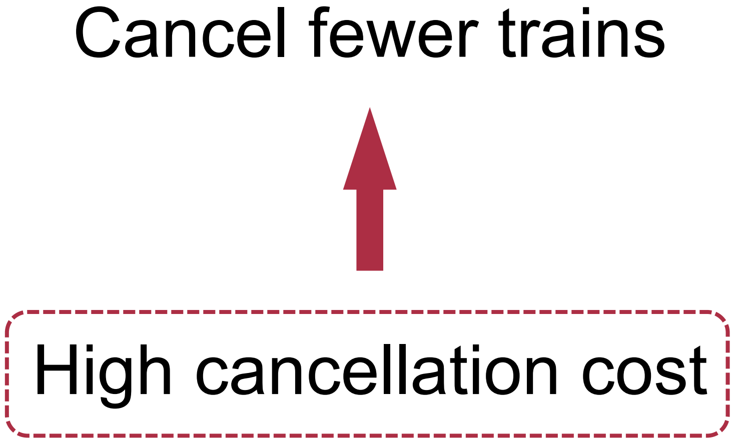
Introduction

Society

- Prefer as planned
- Less changes on short notice
- Fast-generated schedule after disruption

Introduction

Rail Undertaking



Cancel fewer trains

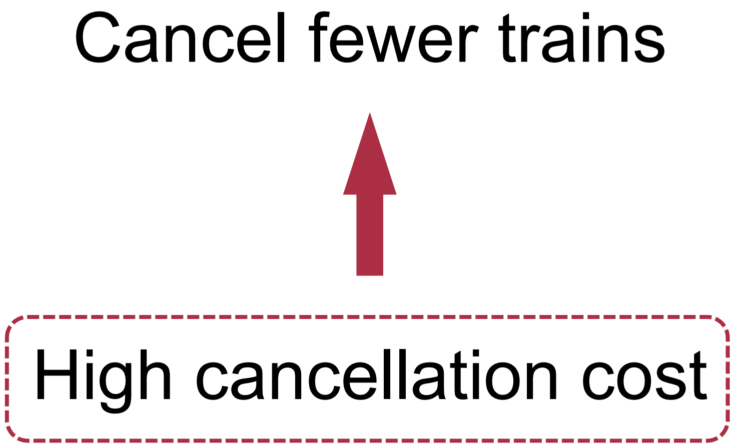
High cancellation cost

Short term

- Financial loss
- Employees’ working time loss

Introduction

Rail Undertaking



Cancel fewer trains

High cancellation cost

Long term

- Less trust from passengers
- Poor competitiveness

Problem Description

Problem Description

Our scenario

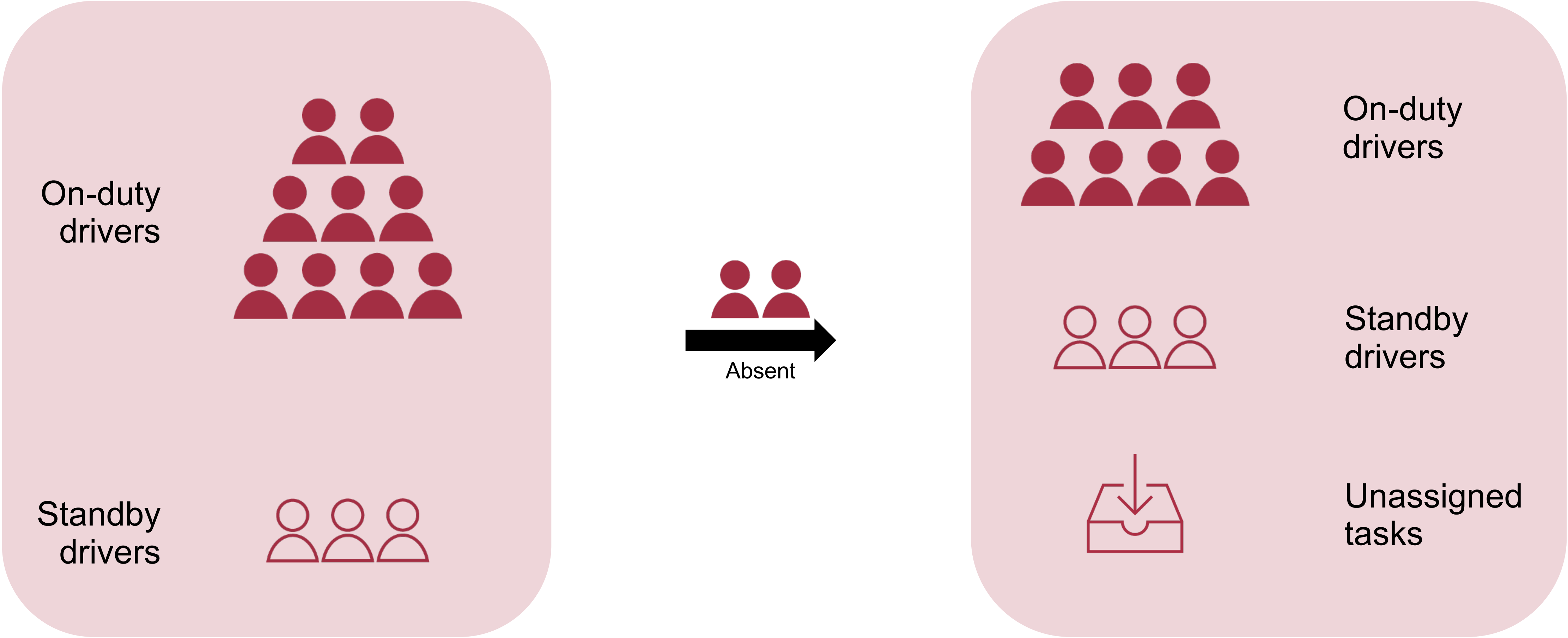
Daily driver shortage

- Take leaves in short notice
- Insufficient standby drivers

Goal

- # Task cancellations
- # Changed tasks

Problem Description



On-duty drivers

Standby drivers

Absent

On-duty drivers

Standby drivers

Unassigned tasks

Original

Now

Problem Description

On-duty drivers	Given (feasible) schedules
Standby drivers	Empty schedules with fixed working time and depot
Unassigned tasks	Set of unassigned tasks

Problem Description

On-duty drivers	Given (feasible) schedules
Standby drivers	Empty schedules with fixed working time and depot
Unassigned tasks	Set of unassigned tasks

Problem Description

On-duty drivers	Given (feasible) schedules
Standby drivers	Empty schedules with fixed working time and depot
Unassigned tasks	Set of unassigned tasks

Methodology

Methodology

MILP problem

Objective function

$$\text{minimize} \quad f\left(x_{g,d}, z_{g,d}\right)$$

Methodology

LINKÖPINGS UNIVERSITET

MILP problem

Objective function

$$\text{minimize} \quad f\left(x_{g,d}, z_{g,d}\right)$$

Total number of unassigned tasks

Methodology

MILP problem

Objective function

$$\text{minimize} \quad f\left(\underbrace{x_{g,d}}_{\text{Total number of unassigned tasks}}, \underbrace{z_{g,d}}_{\text{Total number of changed tasks}}\right)$$

Methodology

MILP problem

Constraints

- Consistent connections of time and geographical location
- Total working time
- Driver’s license
- Rest
- Break

Methodology

LINKÖPINGS UNIVERSITET

MILP problem

Constraints

- Consistent connections of time and geographical location
- Total working time
- Driver’s license
- Rest
- Break

Methodology

LINKÖPINGS UNIVERSITET

MILP problem

Constraints

- Consistent connections of time and geographical location
- Total working time
- Driver’s license
- Rest
- Break

MILP problem

Constraints

- Consistent connections of time and geographical location
- Total working time
- Driver’s license
- Rest
- Break

Methodology

MILP problem

Constraints

- Consistent connections of time and geographical location
- Total working time
- Driver’s license
- Rest
- Break
  - Break time duration
  - Maximum work hour without a break

Methodology

MILP Model with Commercial Solver

- to get the optimal solution

Approach Based on Tabu Search

- less computational time and space
- good enough result

Methodology

Tabu Search

A local search-based heuristic that avoids revisiting solutions by recording the recent history of the search in a short-time memory called Tabu List. [1]

[1] Froger, A. *et al*. (2016) ‘Maintenance scheduling in the Electricity Industry: A Literature Review’, *European Journal of Operational Research*, 251(3), pp. 695–706. doi:10.1016/j.ejor.2015.08.045.

Methodology

LINKÖPINGS UNIVERSITET

Tabu Search

A local search-based heuristic that avoids revisiting solutions by recording the recent history of the search in a short-time memory called Tabu List. [1]

[1] Froger, A. *et al.* (2016) 'Maintenance scheduling in the Electricity Industry: A Literature Review', *European Journal of Operational Research*, 251(3), pp. 695–706. doi:10.1016/j.ejor.2015.08.045.

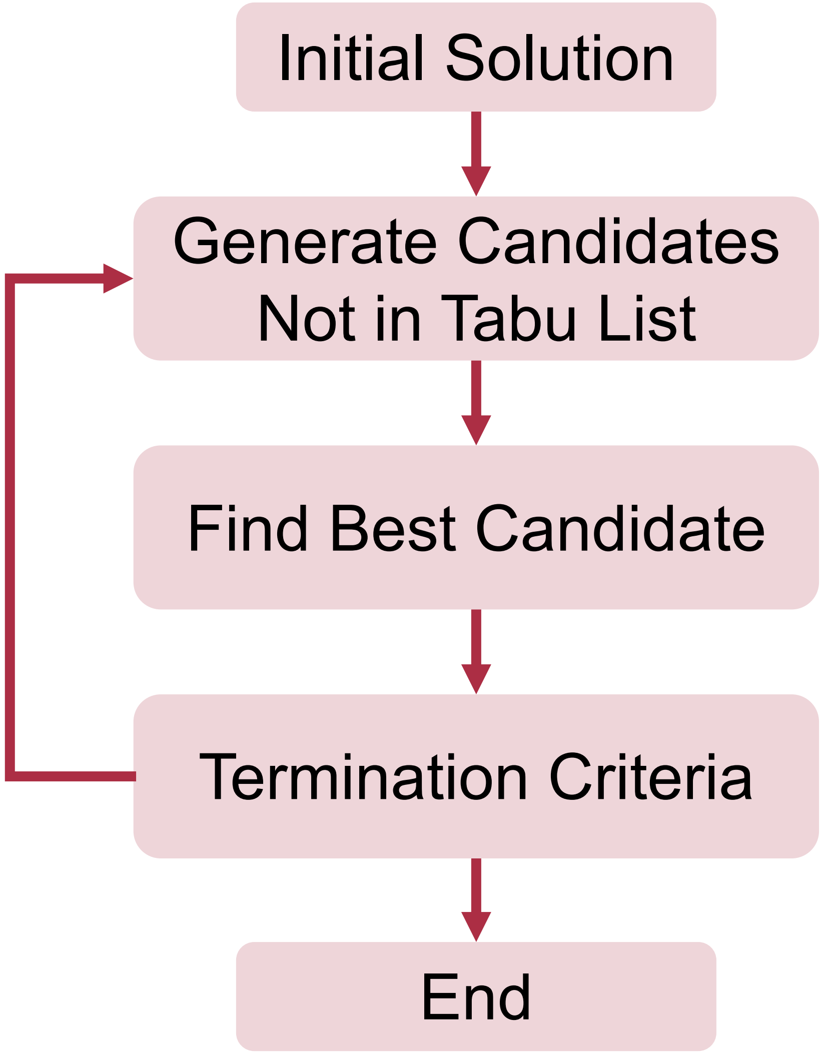
Methodology

Tabu Search

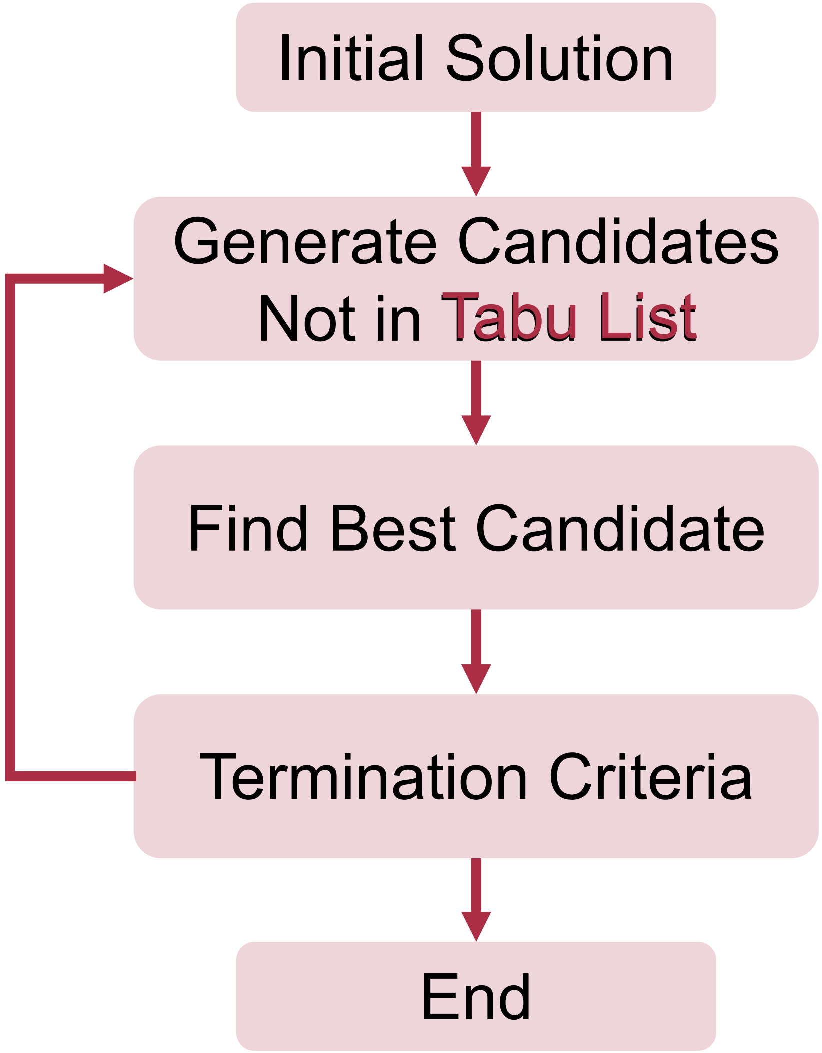
A local search-based heuristic that avoids revisiting solutions by recording the recent history of the search in a short-time memory called Tabu List. [1]

[1] Froger, A. *et al*. (2016) 'Maintenance scheduling in the Electricity Industry: A Literature Review', *European Journal of Operational Research*, 251(3), pp. 695–706. doi:10.1016/j.ejor.2015.08.045.

Methodology



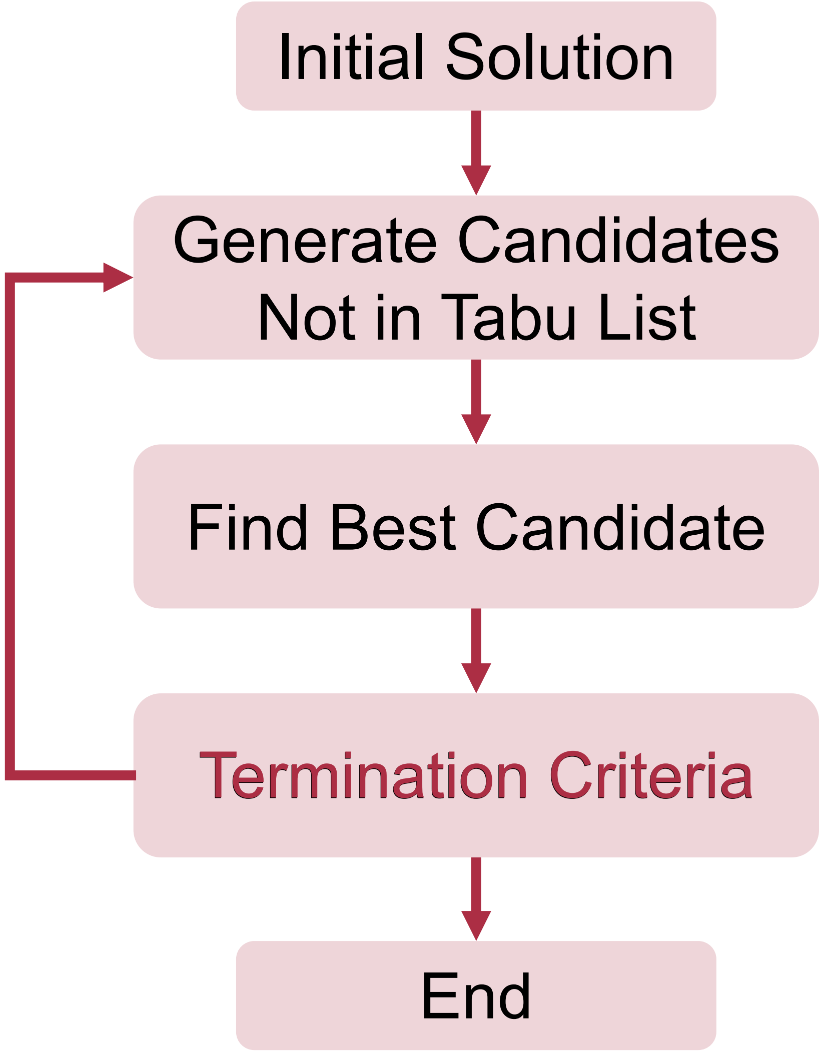
Methodology



Tabu List

- Short-time memory
- Avoiding local optimum

Methodology



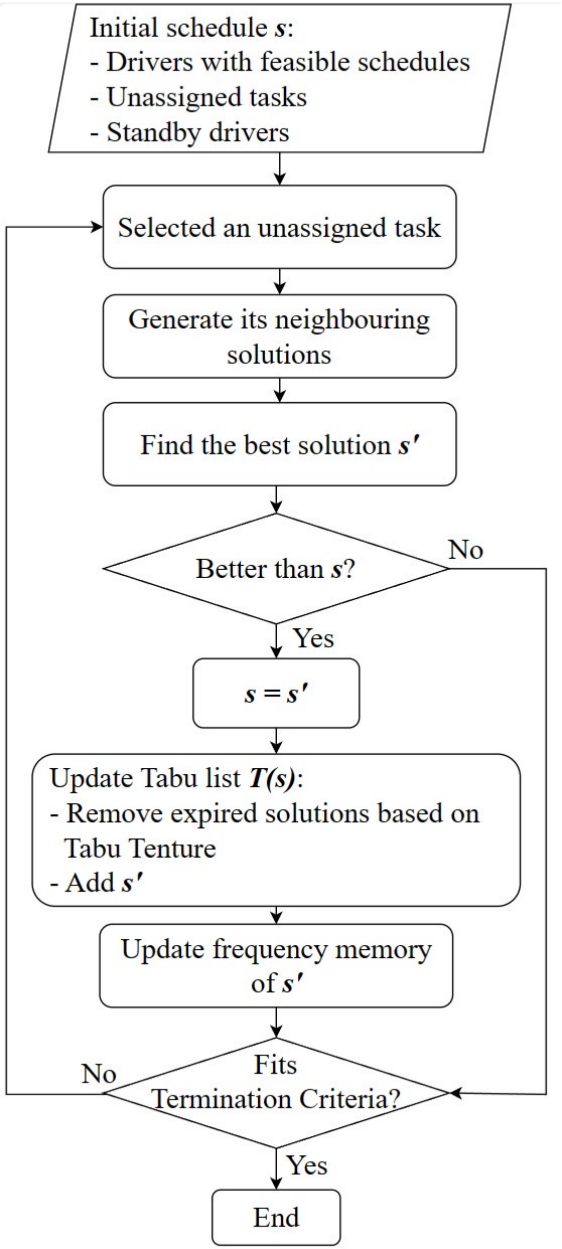
Tabu List

- Short-time memory
- Avoiding local optimum

Termination Criteria

- Terminating the approach

Approach: Tabu Search



Initial schedule s:
- Drivers with feasible schedules
- Unassigned tasks
- Standby drivers

Selected an unassigned task

Generate its neighbouring solutions

Find the best solution s'

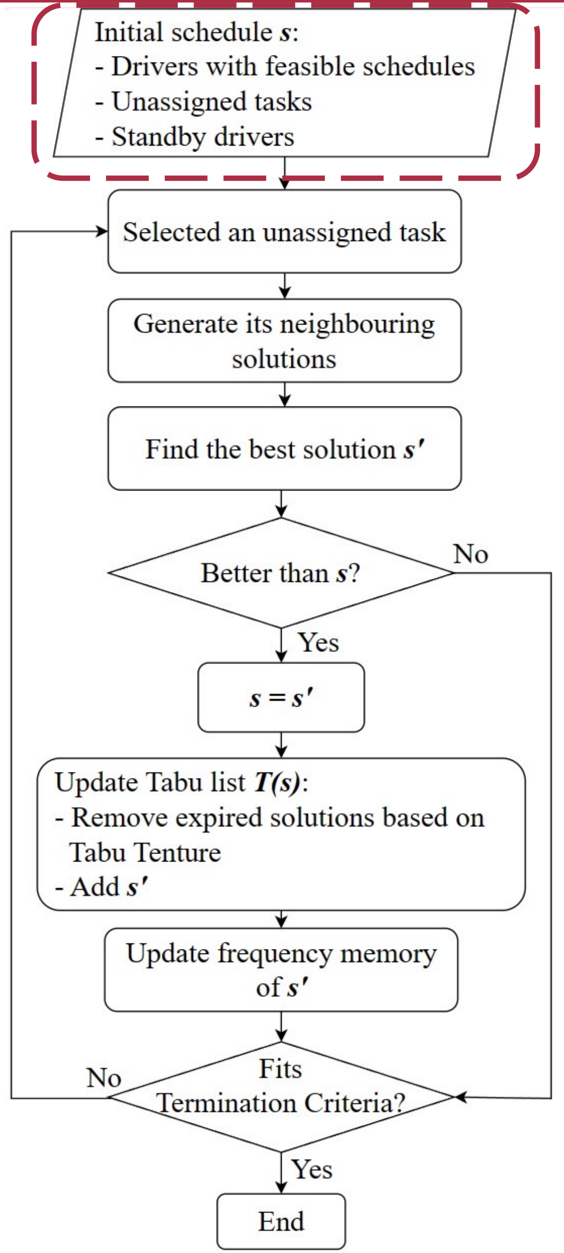
$s = s'$

Update Tabu list $T(s)$:
- Remove expired solutions based on Tabu Tenture
- Add s'

Update frequency memory of s'

End

Approach: Tabu Search



Initial schedule *s*:
- Drivers with feasible schedules
- Unassigned tasks
- Standby drivers

Selected an unassigned task

Generate its neighbouring solutions

Find the best solution ***s′***

Better than *s*?

No

Yes

s = s′

Update Tabu list ***T(s)***:
- Remove expired solutions based on Tabu Tenture
- Add ***s′***

Update frequency memory of ***s′***

Fits Termination Criteria?

No

Yes

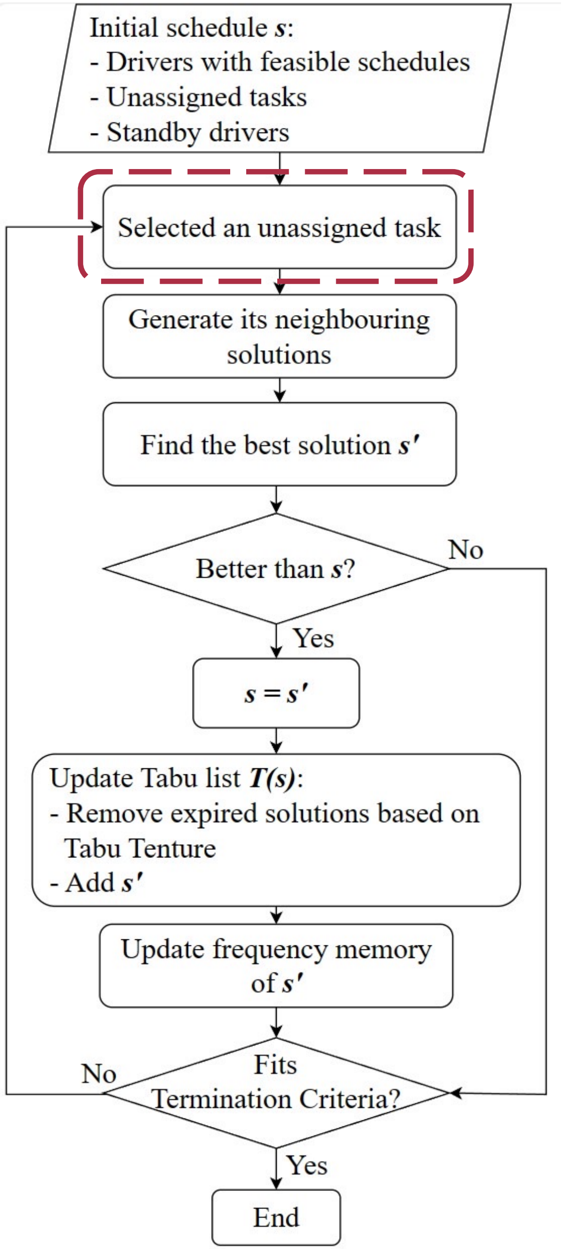
End

On-duty drivers

Standby drivers

Unassigned tasks

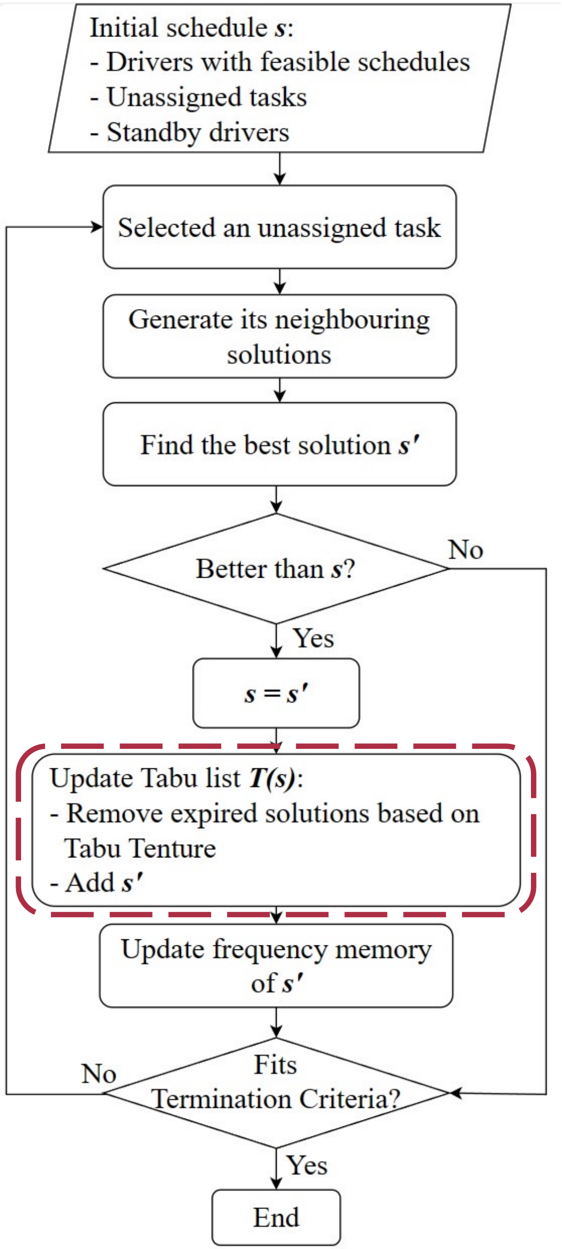
Approach: Tabu Search


Initial schedule s:
- Drivers with feasible schedules
- Unassigned tasks
- Standby drivers
Selected an unassigned task
Generate its neighbouring solutions
Find the best solution s′
Better than s?
No
Yes
s = s′
Update Tabu list T(s):
- Remove expired solutions based on Tabu Tenture
- Add s′
Update frequency memory of s′
Fits Termination Criteria?
No
Yes
End

Select an unassigned task:

- Randomly

Approach: Tabu Search

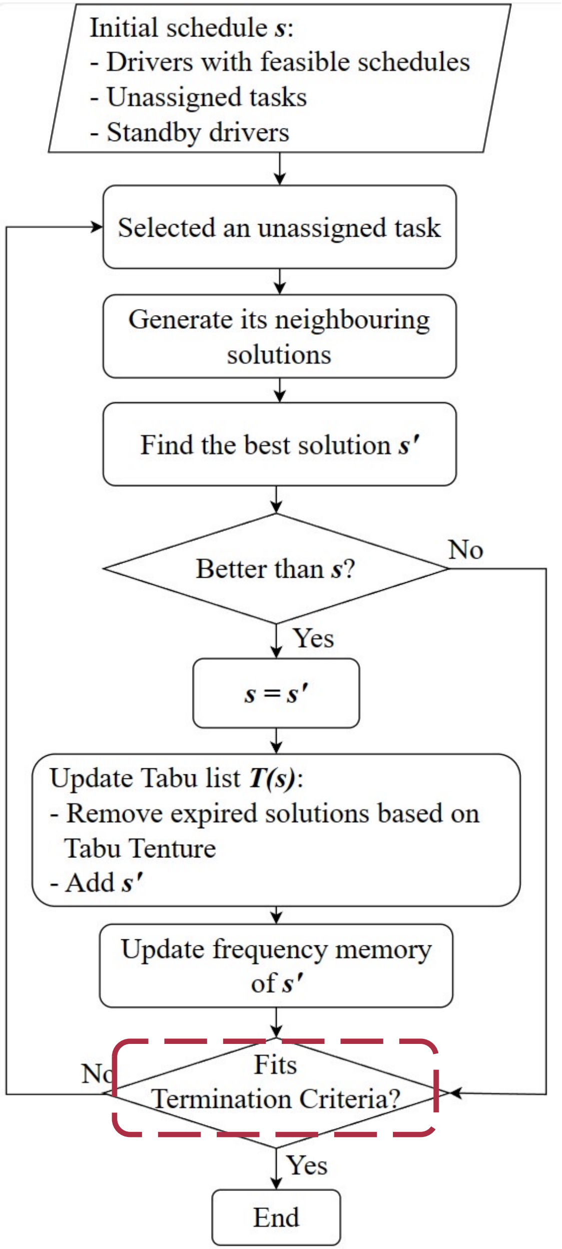


Initial schedule s:
- Drivers with feasible schedules
- Unassigned tasks
- Standby drivers
Selected an unassigned task
Generate its neighbouring solutions
Find the best solution s'
Better than s?
No
Yes
$s = s'$
Update Tabu list $T(s)$:
- Remove expired solutions based on Tabu Tenture
- Add s'
Update frequency memory of s'
Fits Termination Criteria?
No
Yes
End

Tabu List:

- The schedule of all drivers

Approach: Tabu Search



Initial schedule *s*:
- Drivers with feasible schedules
- Unassigned tasks
- Standby drivers

Selected an unassigned task

Generate its neighbouring solutions

Find the best solution s'

Better than *s*?

No

Yes

$s = s'$

Update Tabu list *T(s)*:
- Remove expired solutions based on Tabu Tenture
- Add s'

Update frequency memory of s'

Fits Termination Criteria?

No

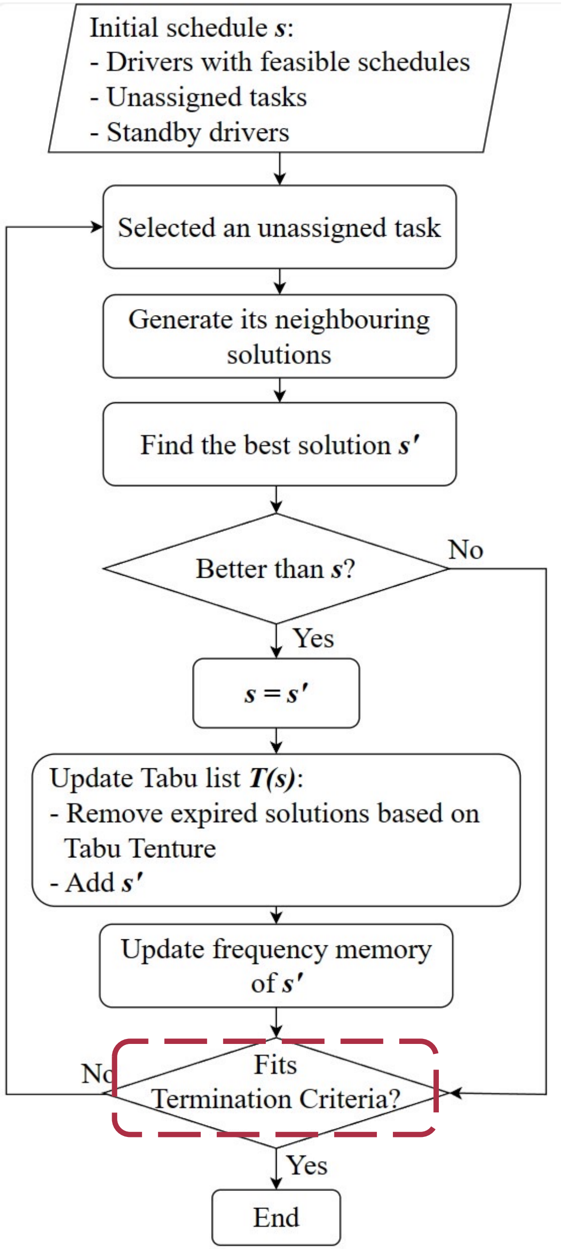
Yes

End

Termination Criteria:

- Maximum number of iteration
- The rest of all unassigned tasks cannot be assigned
- The unassigned task pool is empty

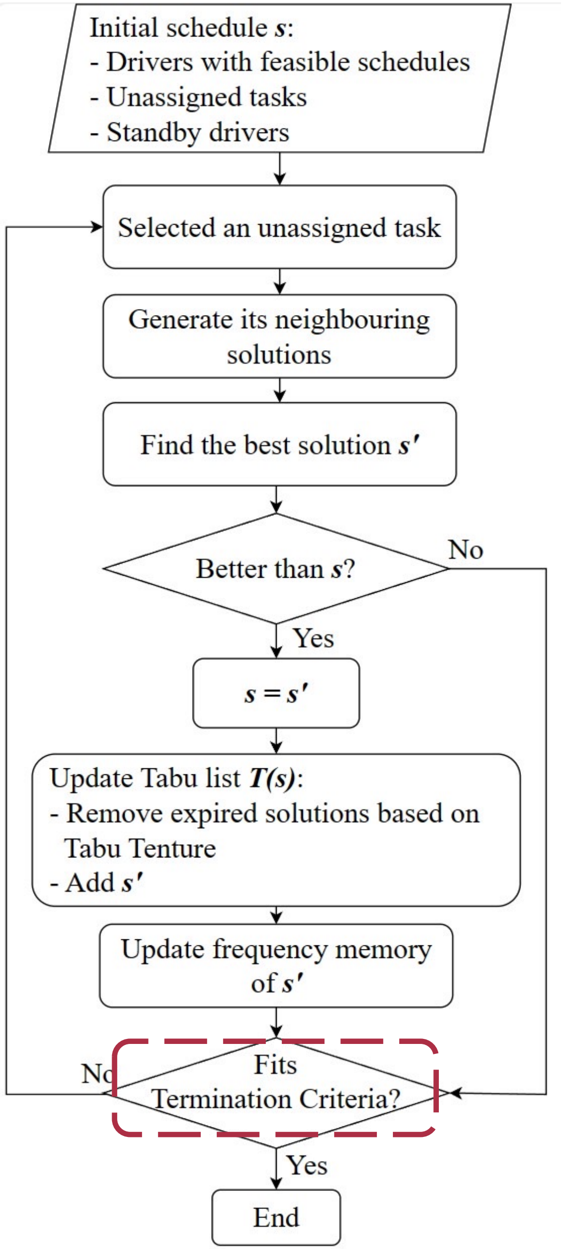
Approach: Tabu Search


Initial schedule *s*:
- Drivers with feasible schedules
- Unassigned tasks
- Standby drivers
Selected an unassigned task
Generate its neighbouring solutions
Find the best solution *s'*
Better than *s*?
No
Yes
s = s'
Update Tabu list *T(s)*:
- Remove expired solutions based on Tabu Tenture
- Add *s'*
Update frequency memory of *s'*
No
Fits Termination Criteria?
Yes
End

Termination Criteria:

- Maximum number of iteration
- The rest of all unassigned tasks cannot be assigned
- The unassigned task pool is empty

Approach: Tabu Search

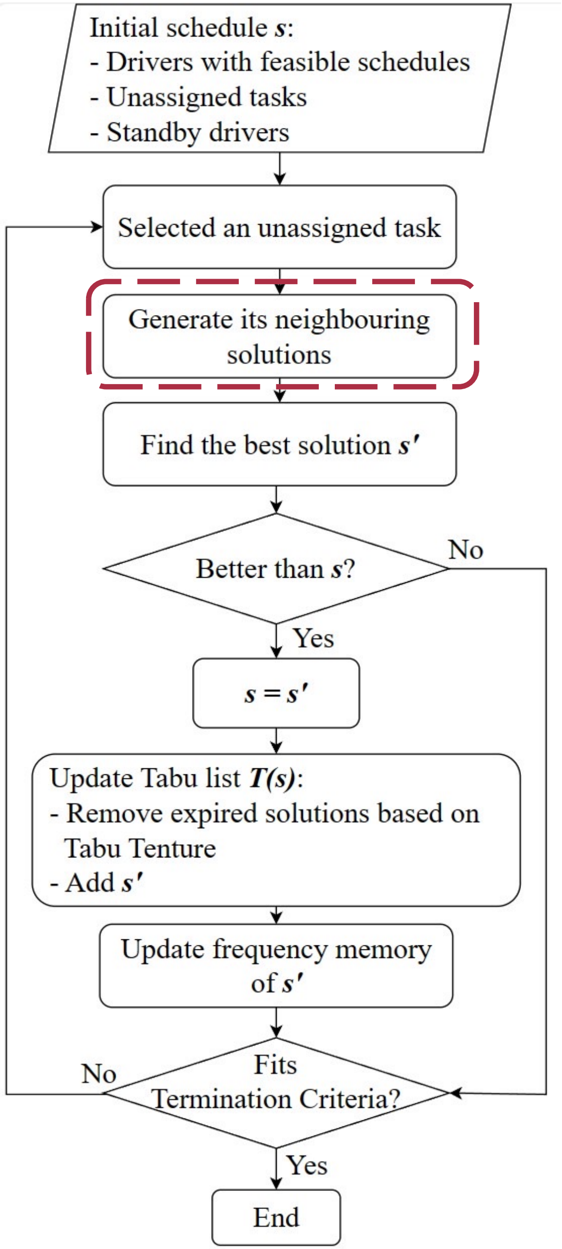


Initial schedule *s*:
- Drivers with feasible schedules
- Unassigned tasks
- Standby drivers
Selected an unassigned task
Generate its neighbouring solutions
Find the best solution *s′*
Better than *s*?
No
Yes
s = *s′*
Update Tabu list *T(s)*:
- Remove expired solutions based on Tabu Tenture
- Add *s′*
Update frequency memory of *s′*
No
Fits Termination Criteria?
Yes
End

Termination Criteria:

- Maximum number of iteration
- The rest of all unassigned tasks cannot be assigned
- The unassigned task pool is empty

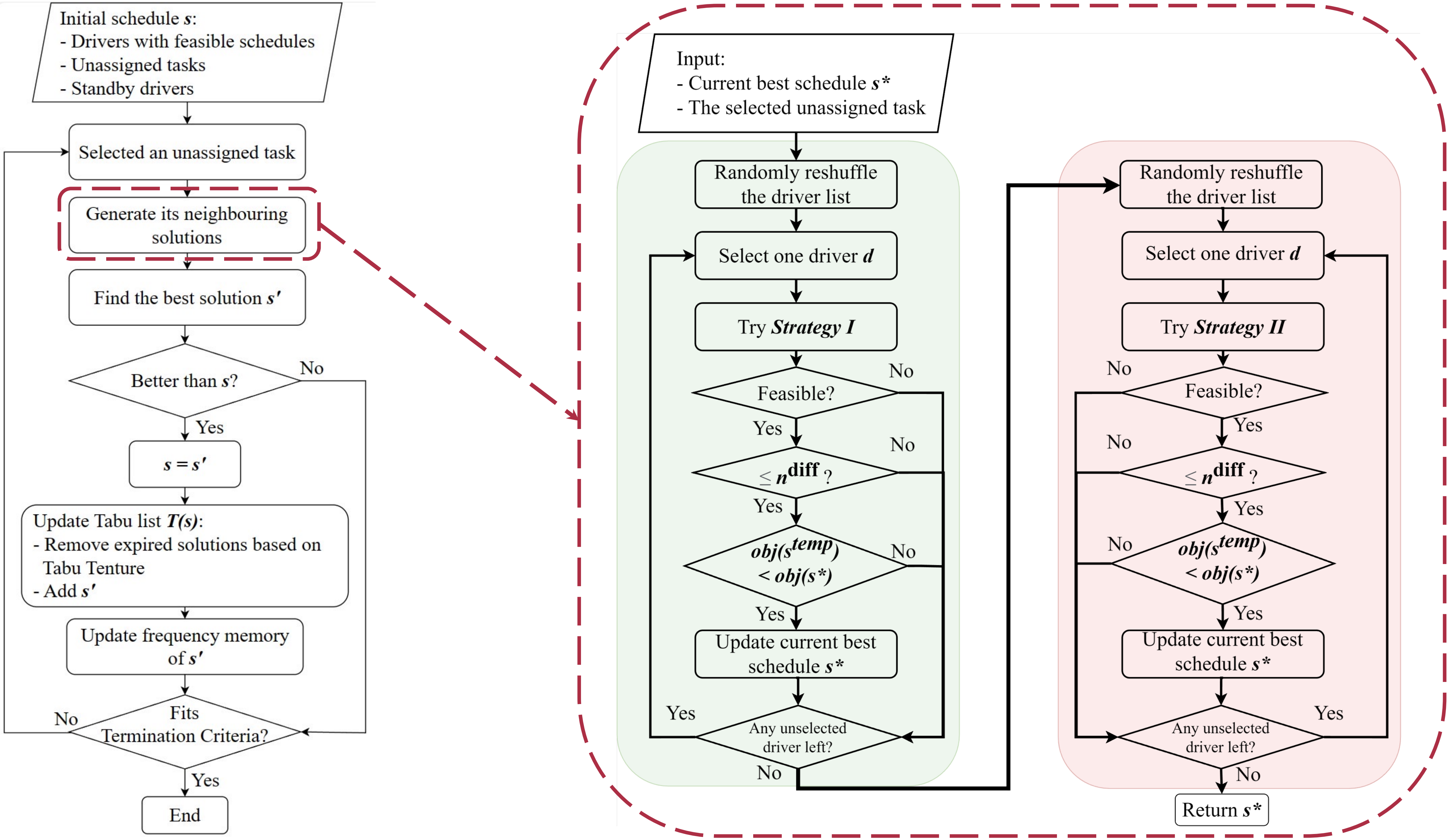
Approach: Tabu Search


Initial schedule *s*:
- Drivers with feasible schedules
- Unassigned tasks
- Standby drivers
Selected an unassigned task
Generate its neighbouring solutions
Find the best solution *s′*
Better than *s*?
No
Yes
s = *s′*
Update Tabu list *T(s)*:
- Remove expired solutions based on Tabu Tenture
- Add *s′*
Update frequency memory of *s′*
Fits Termination Criteria?
No
Yes
End

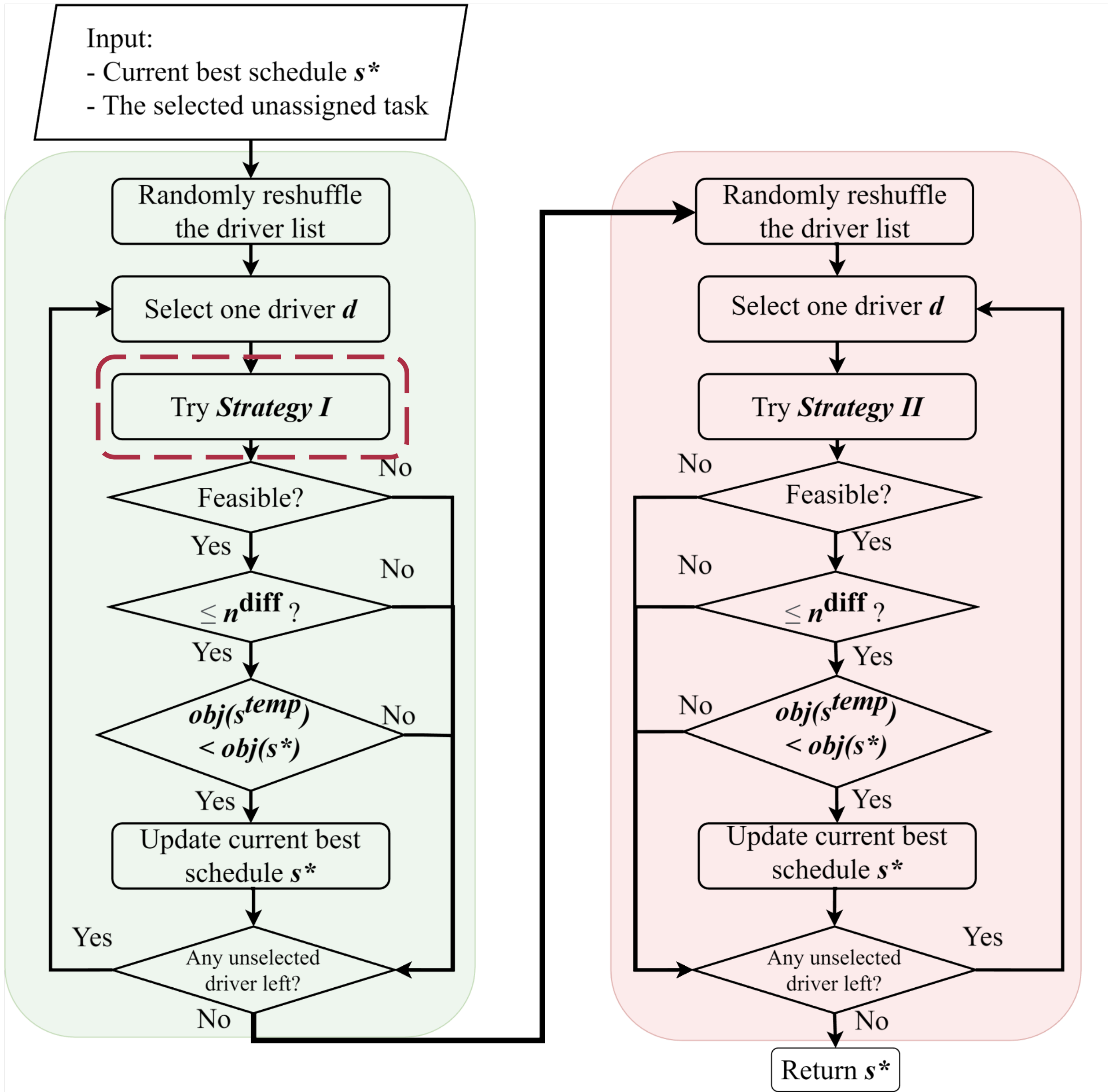
Neighboring Solutions:

- Deadheading
- Extra assign

Approach: Tabu Search


Initial schedule s:
- Drivers with feasible schedules
- Unassigned tasks
- Standby drivers
Selected an unassigned task
Generate its neighbouring solutions
Find the best solution s'
Better than s?
No
Yes
$s = s'$
Update Tabu list $T(s)$:
- Remove expired solutions based on Tabu Tenture
- Add s'
Update frequency memory of s'
Fits Termination Criteria?
No
Yes
End
Input:
- Current best schedule s^*
- The selected unassigned task
Randomly reshuffle the driver list
Select one driver d
Try Strategy I
Feasible?
No
Yes
$\leq n^{\text{diff}}$?
No
Yes
$obj(s^{temp}) < obj(s^*)$
No
Yes
Update current best schedule s^*
Any unselected driver left?
Yes
No
Randomly reshuffle the driver list
Select one driver d
Try Strategy II
Feasible?
No
Yes
$\leq n^{\text{diff}}$?
No
Yes
$obj(s^{temp}) < obj(s^*)$
No
Yes
Update current best schedule s^*
Any unselected driver left?
Yes
No
Return s^*

Approach: Neighboring Solutions



Input:
- Current best schedule ***s****
- The selected unassigned task

Randomly reshuffle the driver list

Select one driver ***d***

Try ***Strategy I***

Feasible?

Yes

No

$\leq n^{\mathbf{diff}}$?

Yes

No

$obj(s^{temp}) < obj(s^*)$

Yes

No

Update current best schedule ***s****

Any unselected driver left?

Yes

No

Randomly reshuffle the driver list

Select one driver ***d***

Try ***Strategy II***

Feasible?

Yes

No

$\leq n^{\mathbf{diff}}$?

Yes

No

$obj(s^{temp}) < obj(s^*)$

Yes

No

Update current best schedule ***s****

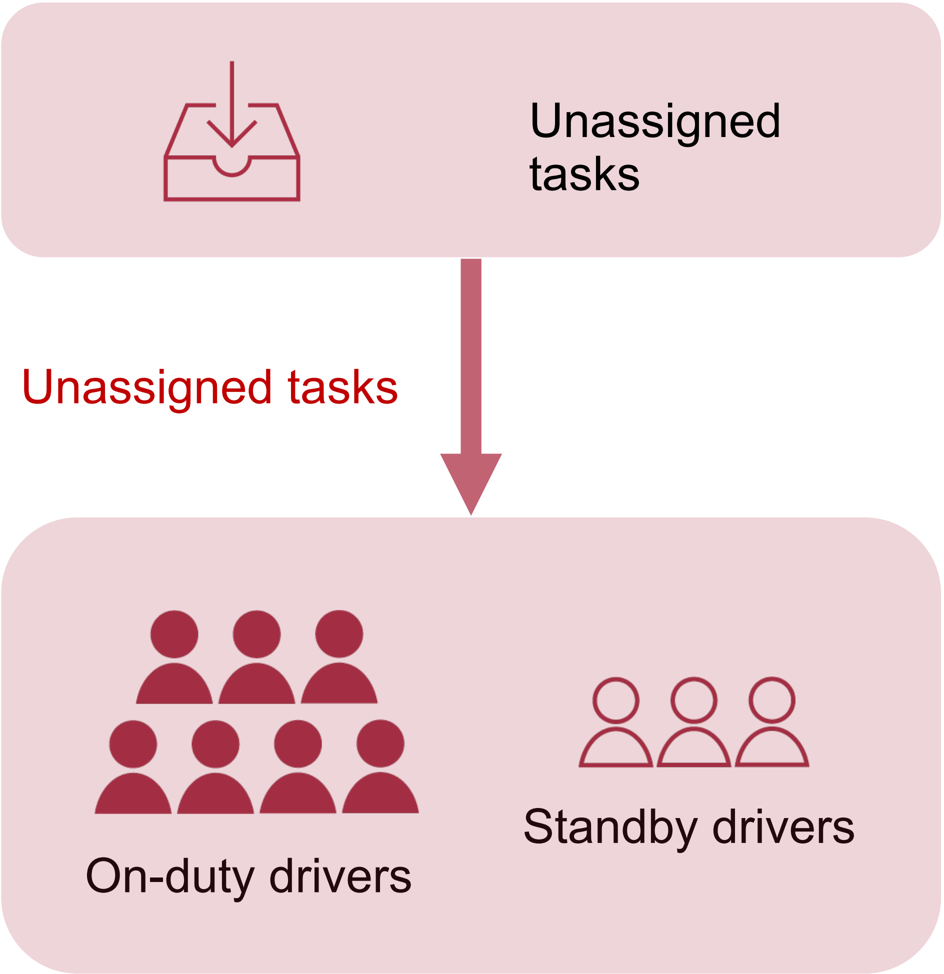
Any unselected driver left?

Yes

No

Return ***s****

Strategy I: Directly assign to all drivers



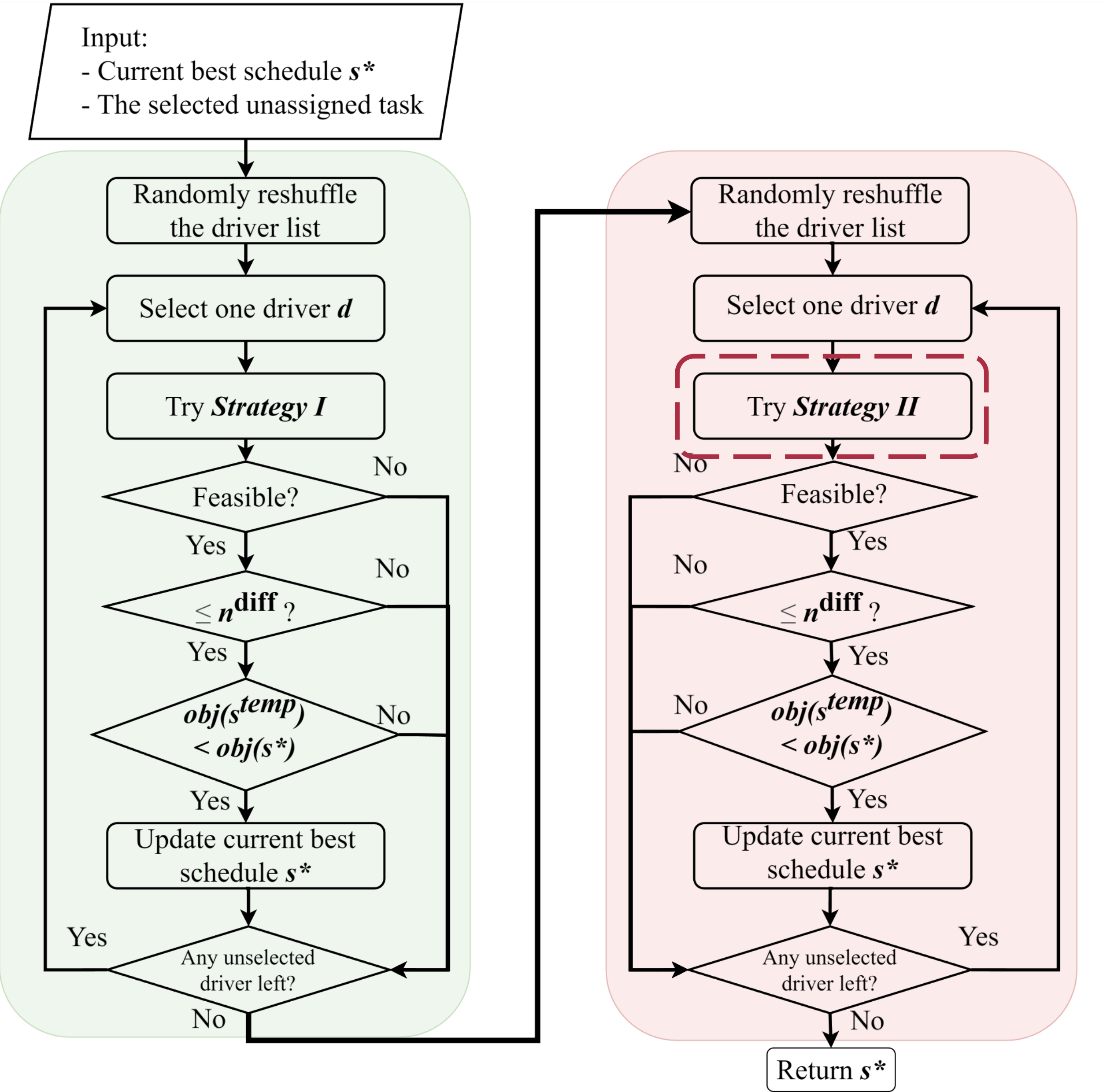
Unassigned tasks

Unassigned tasks

On-duty drivers

Standby drivers

Approach: Neighboring Solutions



Input:
- Current best schedule ***s****
- The selected unassigned task

Randomly reshuffle the driver list

Select one driver ***d***

Try ***Strategy I***

Feasible? Yes / No

$\leq n^{\textbf{diff}}$? Yes / No

$obj(s^{temp}) < obj(s^*)$ Yes / No

Update current best schedule ***s****

Any unselected driver left? Yes / No

Randomly reshuffle the driver list

Select one driver ***d***

Try ***Strategy II***

Feasible? Yes / No

$\leq n^{\textbf{diff}}$? Yes / No

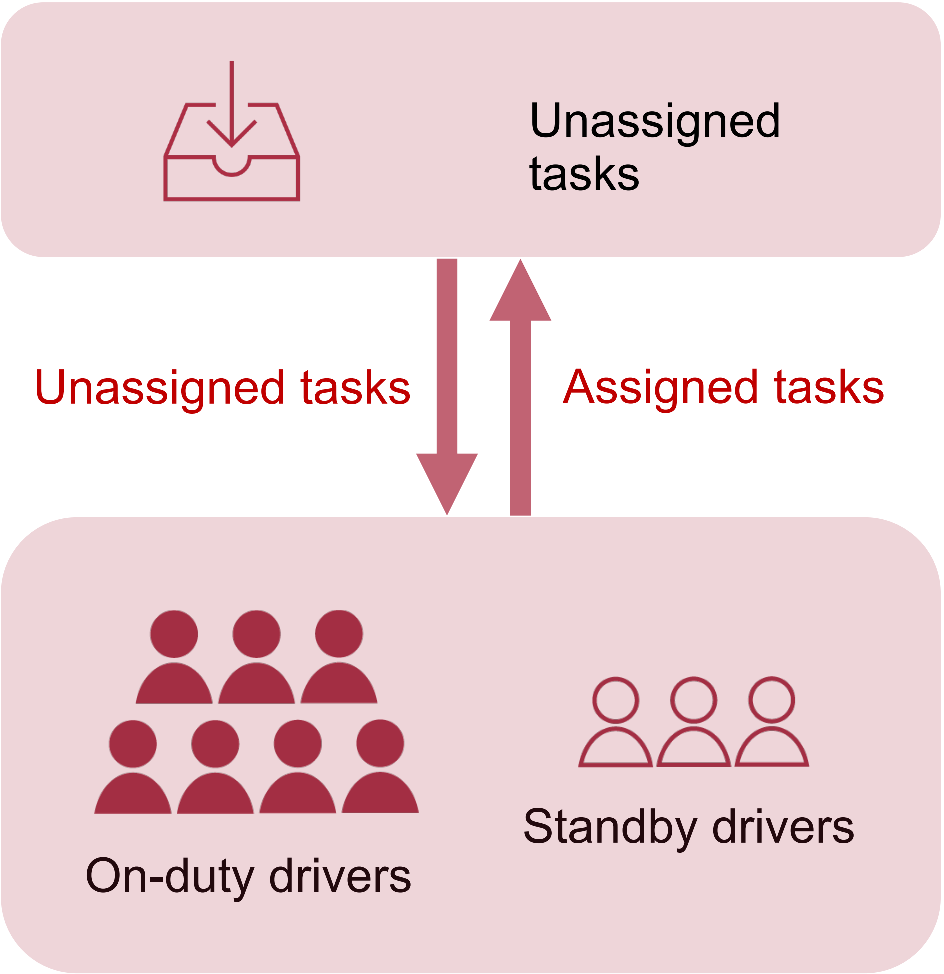
$obj(s^{temp}) < obj(s^*)$ Yes / No

Update current best schedule ***s****

Any unselected driver left? Yes / No

Return ***s****

Strategy II: Swap with assigned tasks



Unassigned tasks

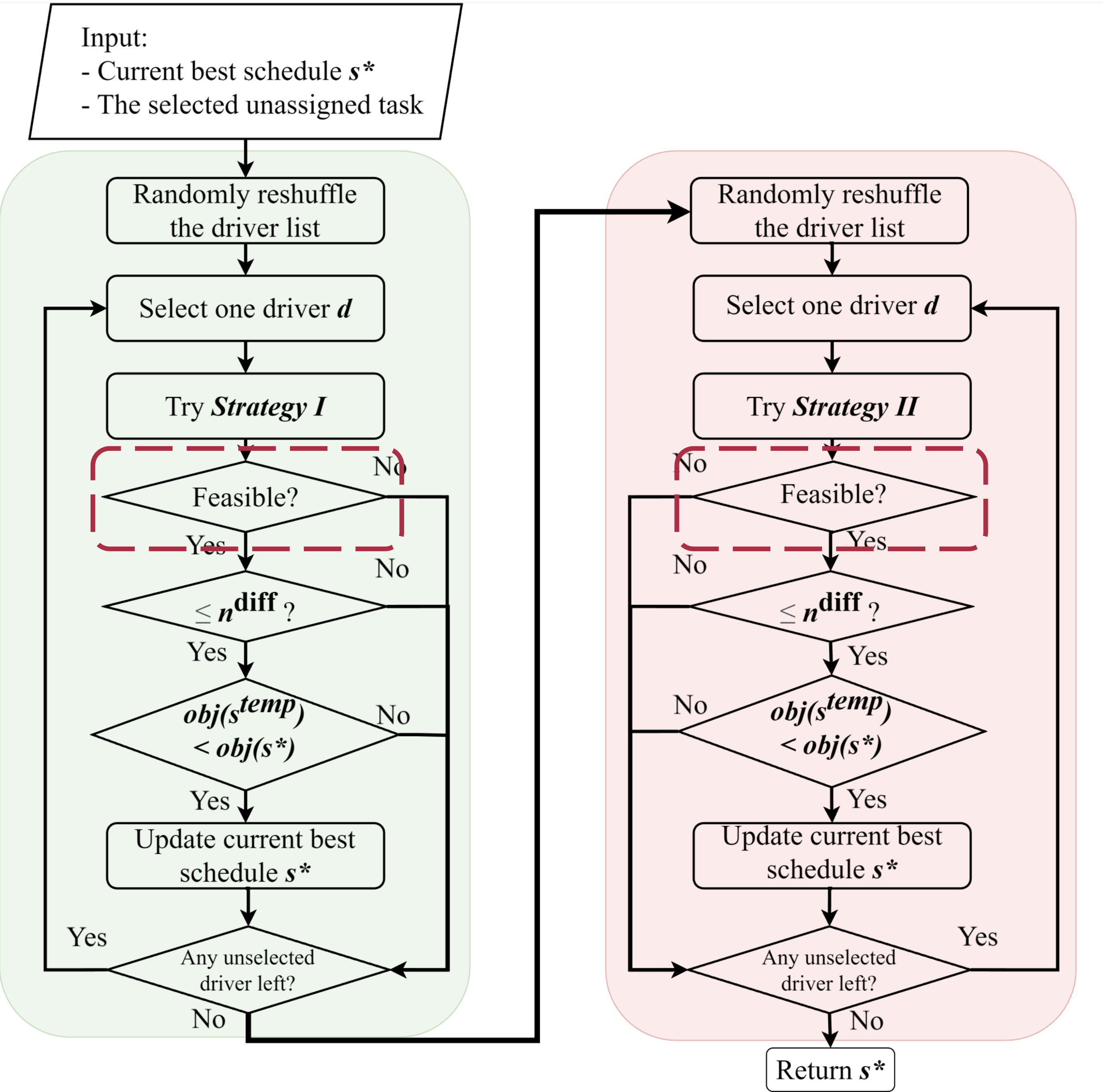
Unassigned tasks

Assigned tasks

On-duty drivers

Standby drivers

Approach: Neighboring Solutions

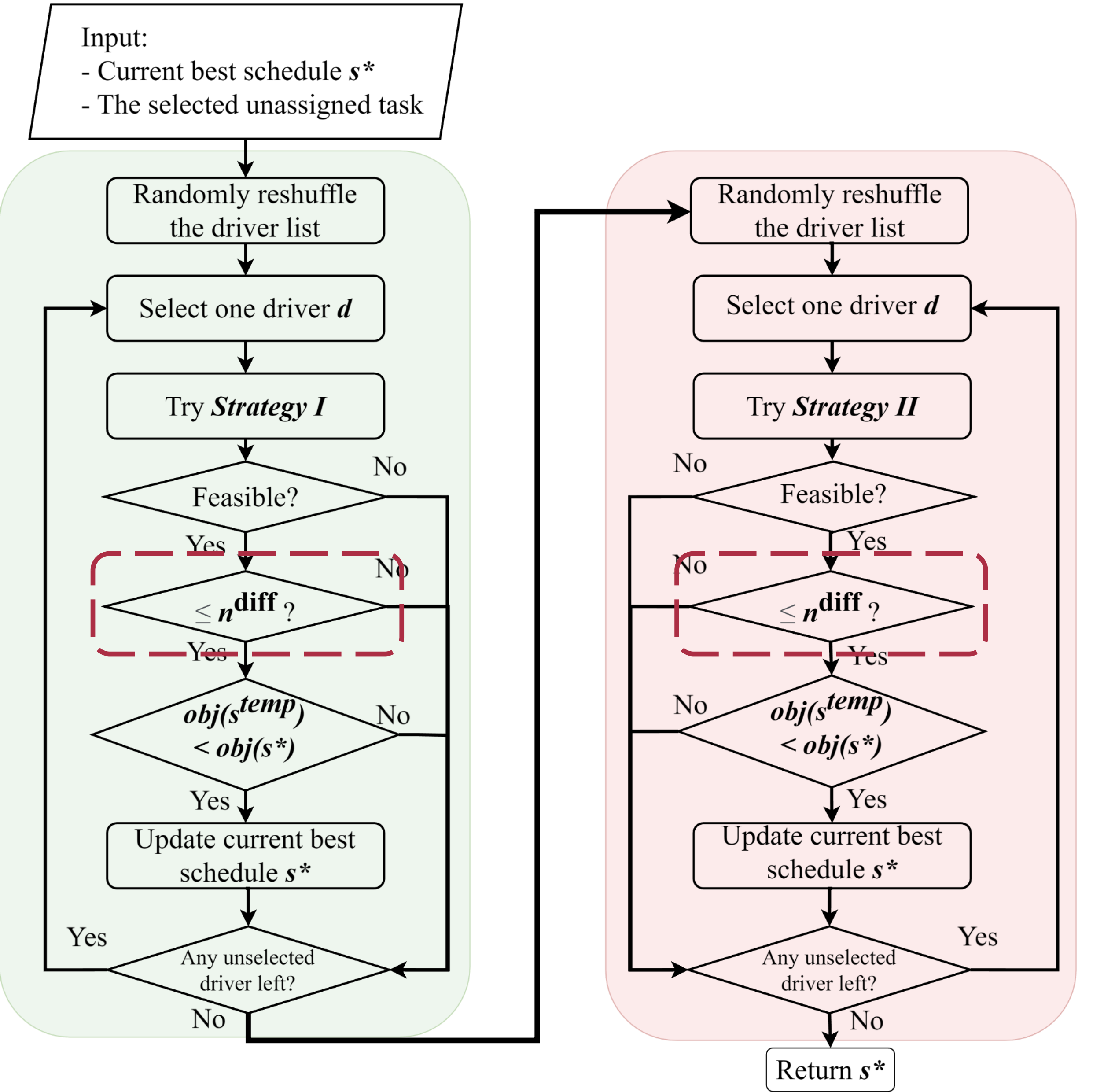

Input:
- Current best schedule *s**
- The selected unassigned task
Randomly reshuffle the driver list
Select one driver *d*
Try *Strategy I*
Feasible?
No
Yes
$\leq n^{\text{diff}}$?
No
Yes
$obj(s^{temp}) < obj(s^*)$
No
Yes
Update current best schedule *s**
Yes
Any unselected driver left?
No
Randomly reshuffle the driver list
Select one driver *d*
Try *Strategy II*
No
Feasible?
Yes
No
$\leq n^{\text{diff}}$?
Yes
No
$obj(s^{temp}) < obj(s^*)$
Yes
Update current best schedule *s**
Yes
Any unselected driver left?
No
Return *s**

Feasibility Check

Equivalent with the constraints in MILP

Approach: Neighboring Solutions

LINKÖPINGS UNIVERSITET



Input:
- Current best schedule *s**
- The selected unassigned task
Randomly reshuffle the driver list
Select one driver *d*
Try *Strategy I*
Feasible?
No
Yes
$\leq n^{\text{diff}}$?
No
Yes
$obj(s^{temp}) < obj(s^*)$
No
Yes
Update current best schedule *s**
Any unselected driver left?
Yes
No
Randomly reshuffle the driver list
Select one driver *d*
Try *Strategy II*
Feasible?
No
Yes
$\leq n^{\text{diff}}$?
No
Yes
$obj(s^{temp}) < obj(s^*)$
No
Yes
Update current best schedule *s**
Any unselected driver left?
Yes
No
Return *s**

Level of freedom

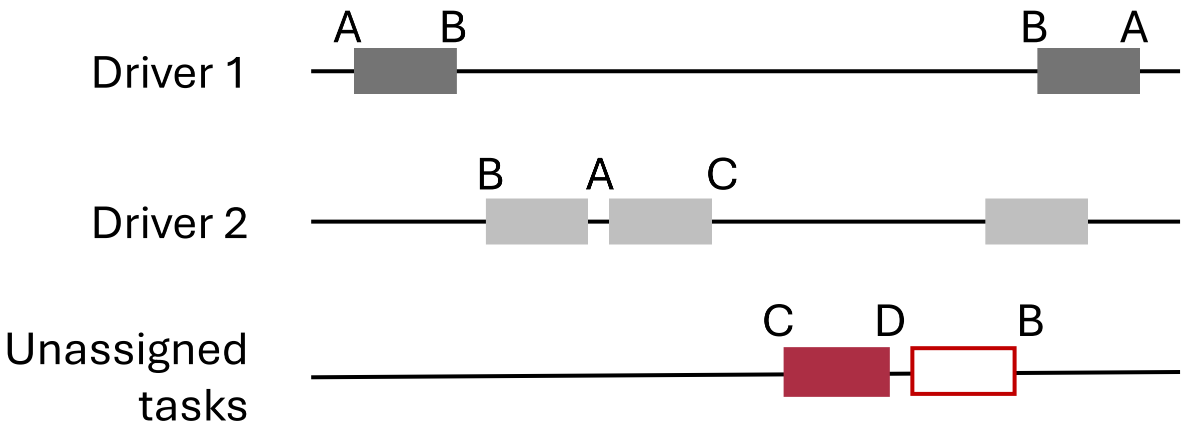
n^{diff}: maximum allowed difference between
# tasks unassigned from driver d and
# tasks assigned to driver d.

Approach: Neighboring Solutions

Strategy I: Directly assign to all drivers

🟥 Selected unassigned task

□ Another unassigned task

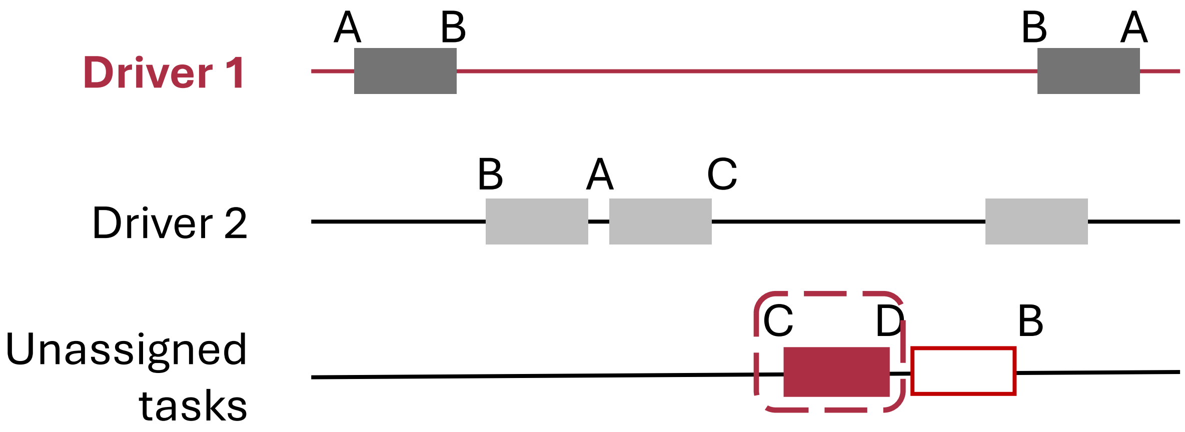


Approach: Neighboring Solutions

Strategy I: Directly assign to all drivers

■ Selected unassigned task

□ Another unassigned task

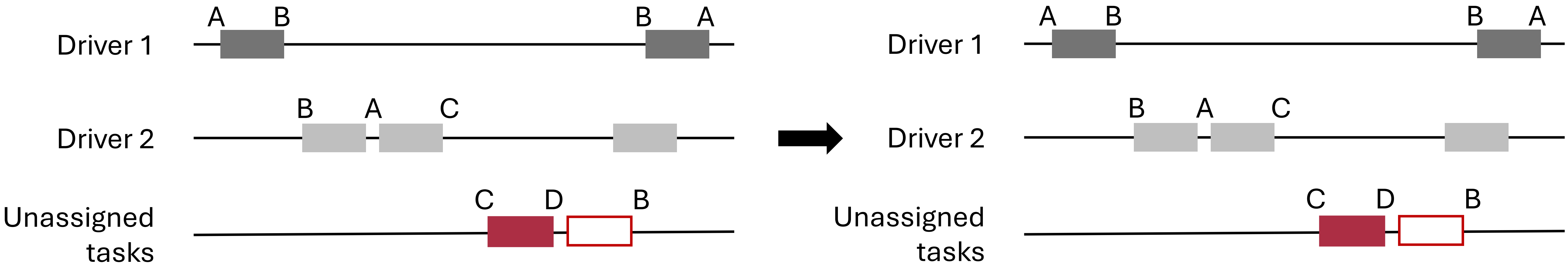


Approach: Neighboring Solutions

Strategy I: Directly assign to all drivers

🟥 Selected unassigned task

□ Another unassigned task

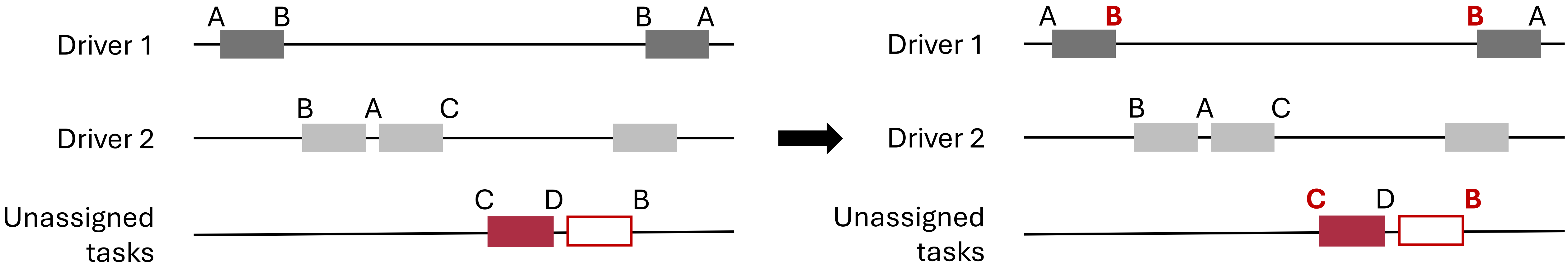


Approach: Neighboring Solutions

Strategy I: Directly assign to all drivers

🟥 Selected unassigned task

□ Another unassigned task



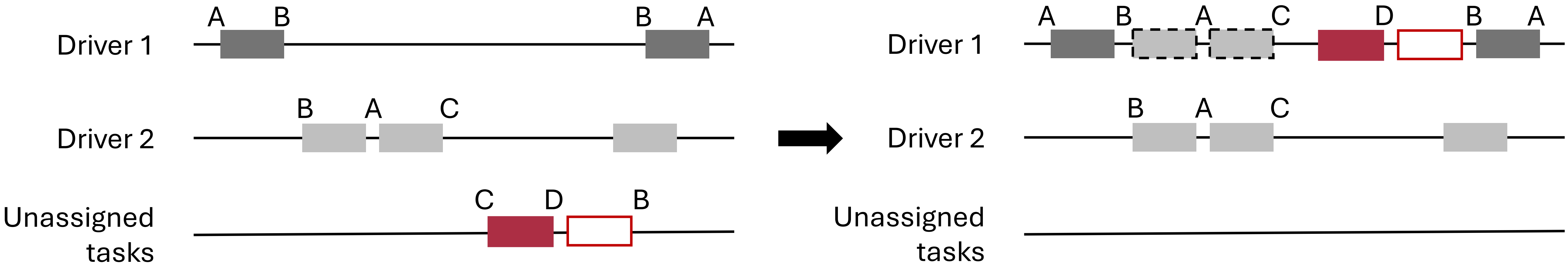
Approach: Neighboring Solutions

Strategy I: Directly assign to all drivers

🟥 Selected unassigned task

□ Another unassigned task

▨ Deadheading assigned task

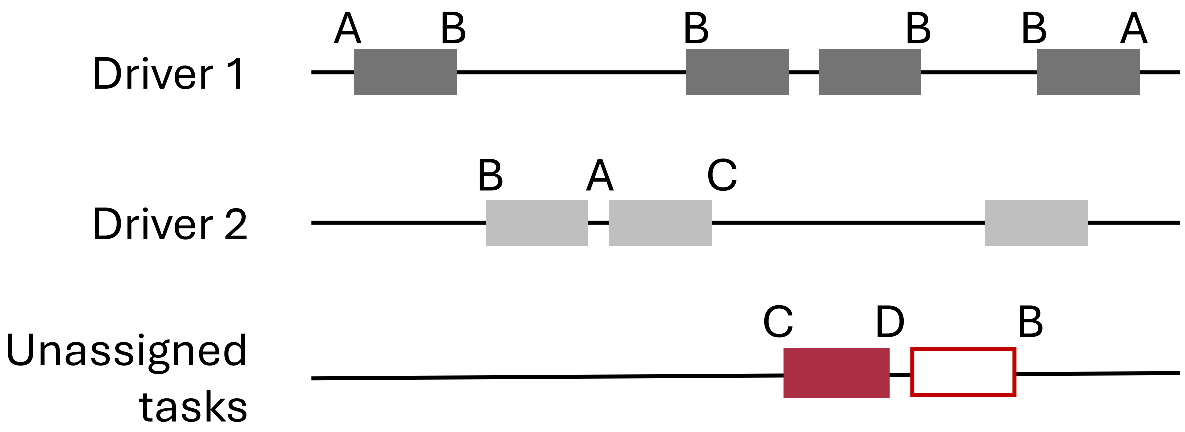


Approach: Neighboring Solutions

Strategy II: Swap with assigned tasks

🟥 Selected unassigned task

□ Another unassigned task

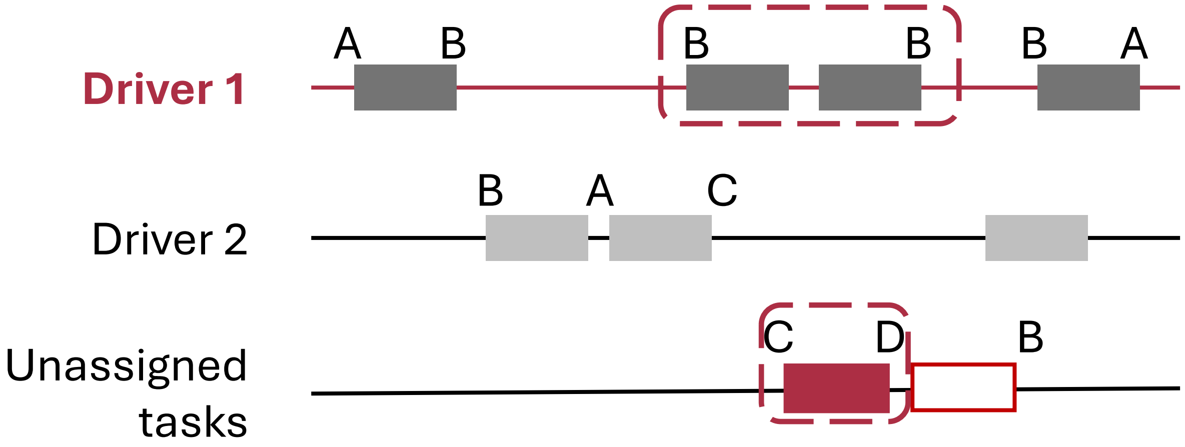


Approach: Neighboring Solutions

Strategy II: Swap with assigned tasks

🟥 Selected unassigned task

□ Another unassigned task

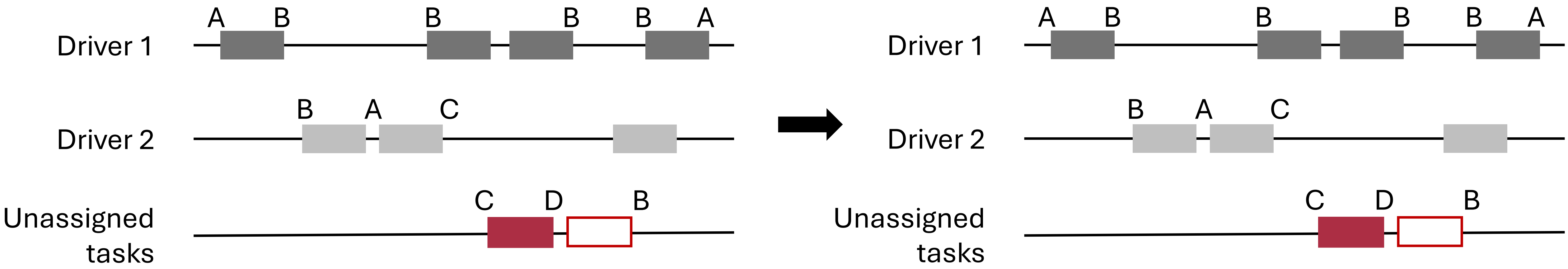


Approach: Neighboring Solutions

Strategy II: Swap with assigned tasks

🟥 Selected unassigned task

□ Another unassigned task

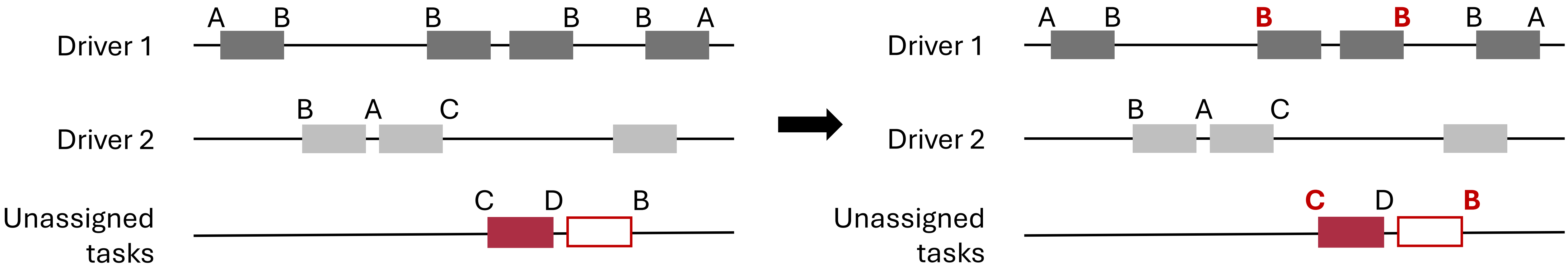


Approach: Neighboring Solutions

Strategy II: Swap with assigned tasks

🟥 Selected unassigned task

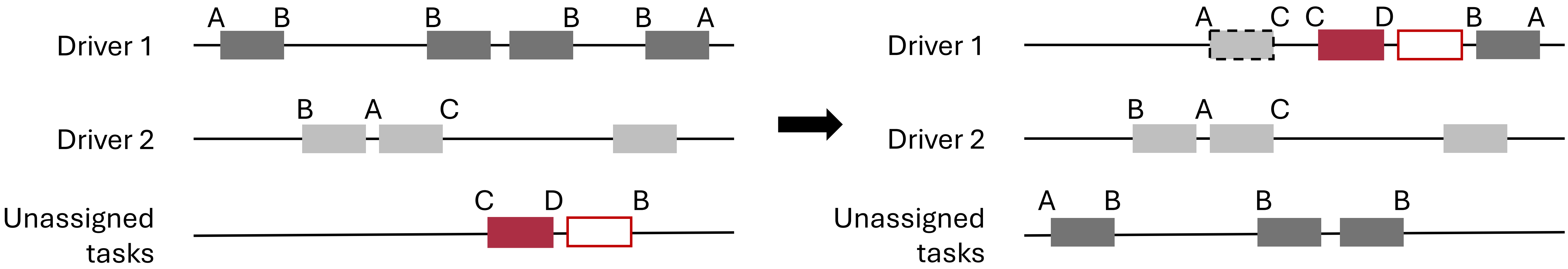
□ Another unassigned task



Approach: Neighboring Solutions

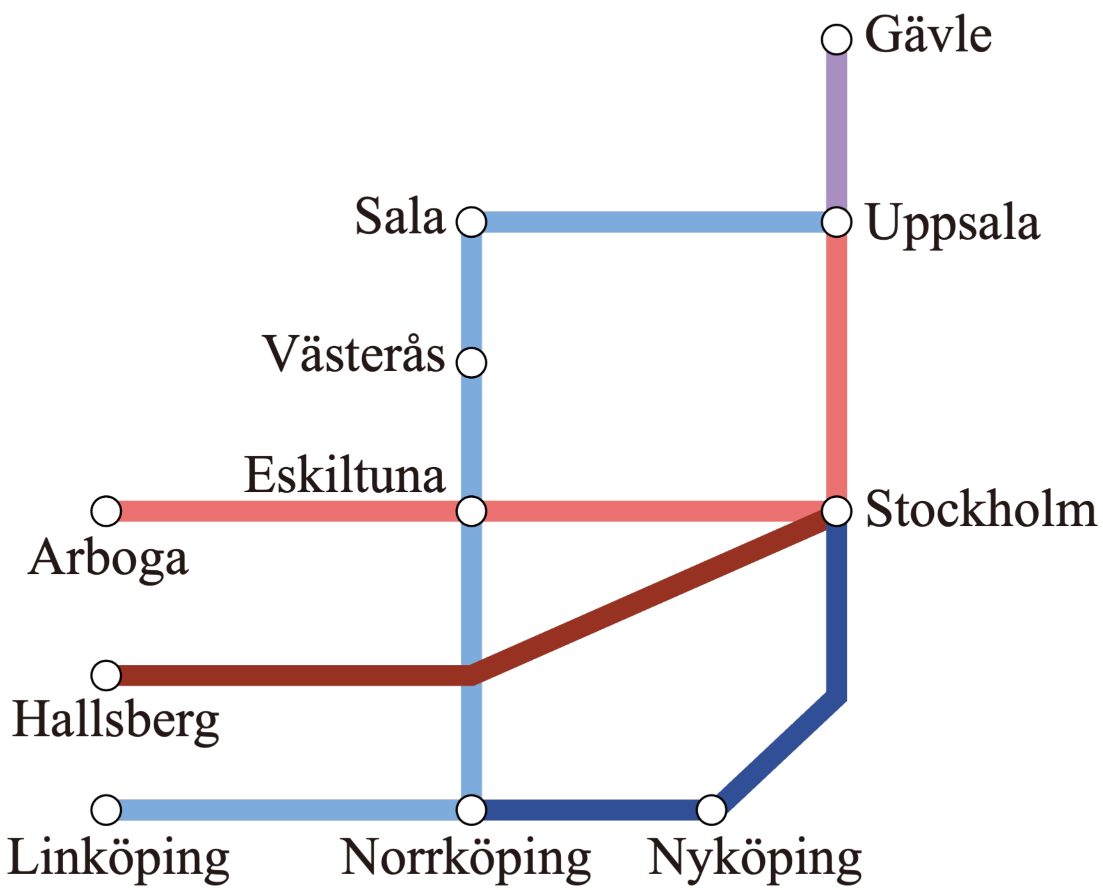
Strategy II: Swap with assigned tasks

- 🟥 Selected unassigned task
- □ Another unassigned task
- ▨ Deadheading assigned task



Case Study

Case Study


Gävle
Sala
Uppsala
Västerås
Eskiltuna
Stockholm
Arboga
Hallsberg
Linköping
Norrköping
Nyköping

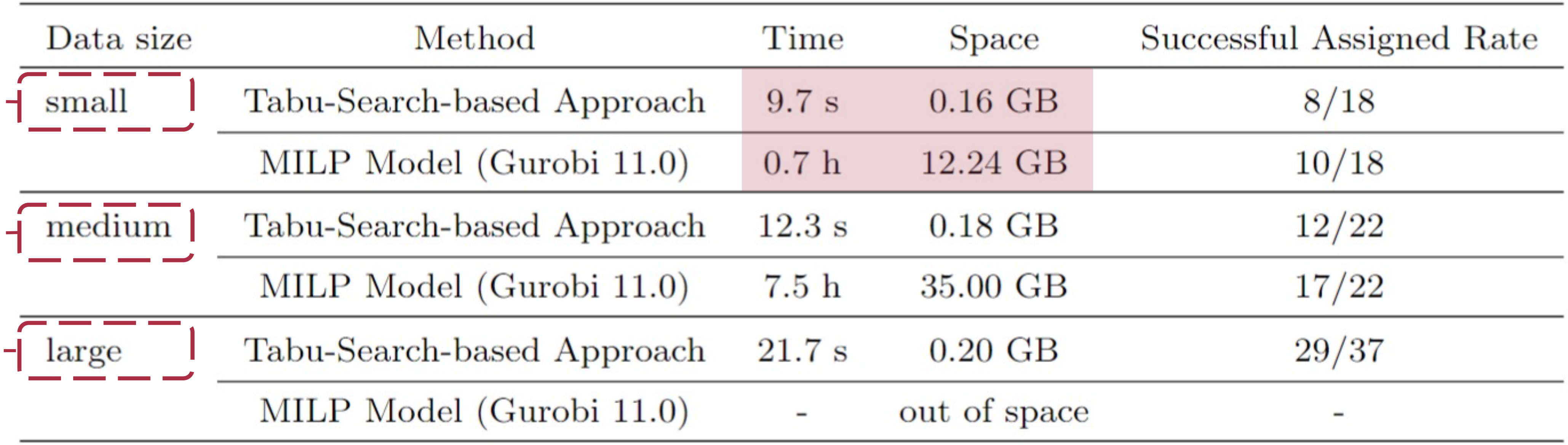
Case Study

MILP Model vs Approach

0.50*Large

0.75*Large

One-day Schedule



Data size	Method	Time	Space	Successful Assigned Rate
small	Tabu-Search-based Approach	9.7 s	0.16 GB	8/18
	MILP Model (Gurobi 11.0)	0.7 h	12.24 GB	10/18
medium	Tabu-Search-based Approach	12.3 s	0.18 GB	12/22
	MILP Model (Gurobi 11.0)	7.5 h	35.00 GB	17/22
large	Tabu-Search-based Approach	21.7 s	0.20 GB	29/37
	MILP Model (Gurobi 11.0)	-	out of space	-

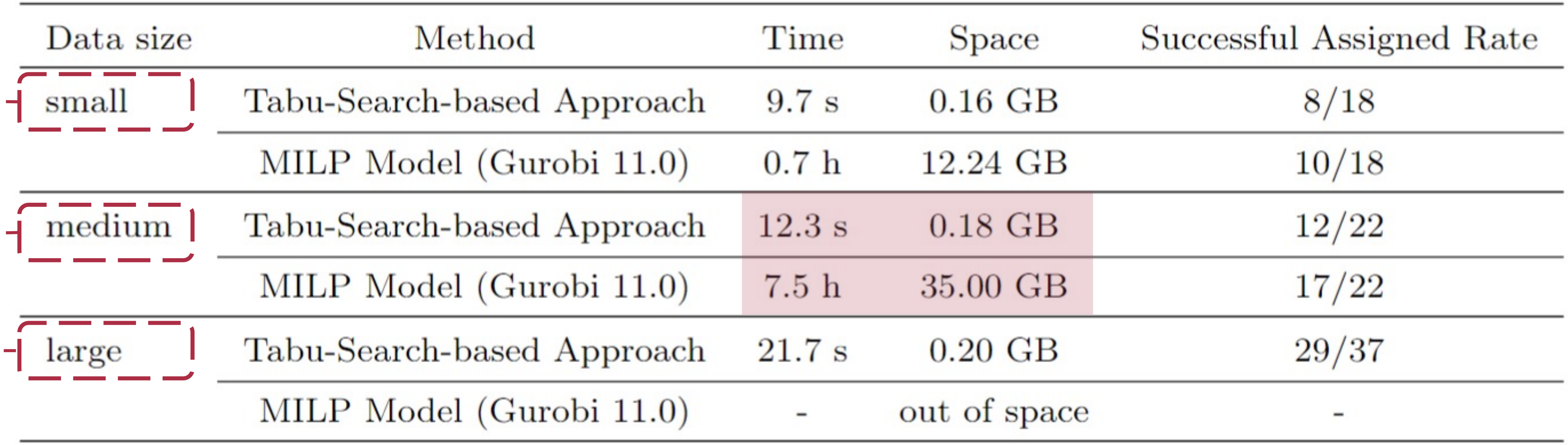
Case Study

MILP Model vs Approach

0.50*Large

0.75*Large

One-day Schedule



Data size	Method	Time	Space	Successful Assigned Rate
small	Tabu-Search-based Approach	9.7 s	0.16 GB	8/18
	MILP Model (Gurobi 11.0)	0.7 h	12.24 GB	10/18
medium	Tabu-Search-based Approach	12.3 s	0.18 GB	12/22
	MILP Model (Gurobi 11.0)	7.5 h	35.00 GB	17/22
large	Tabu-Search-based Approach	21.7 s	0.20 GB	29/37
	MILP Model (Gurobi 11.0)	-	out of space	-

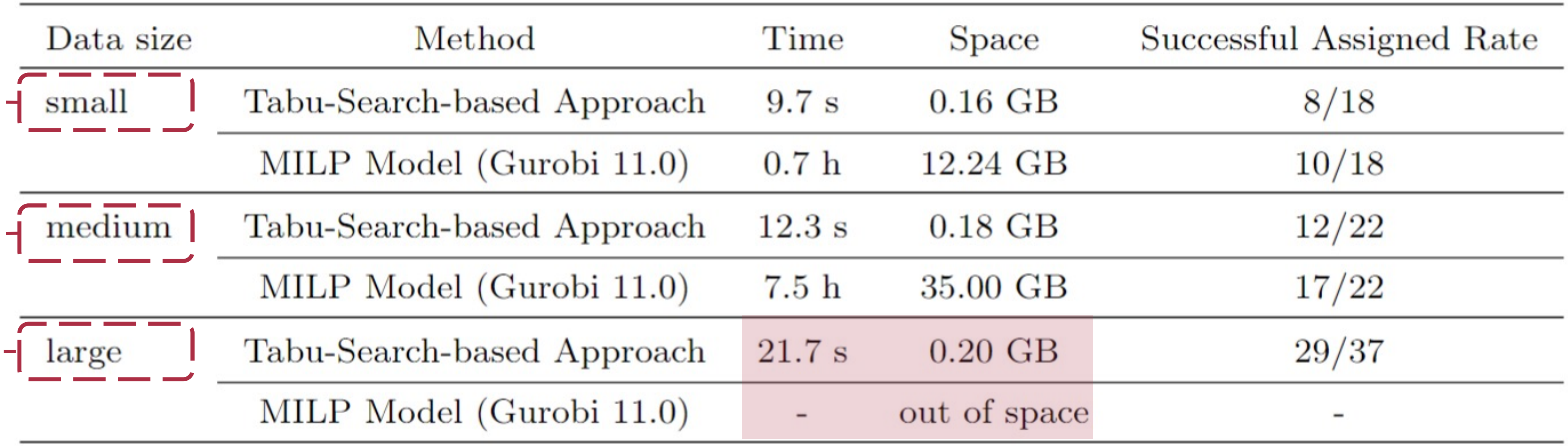
Case Study

MILP Model vs Approach

0.50*Large

0.75*Large

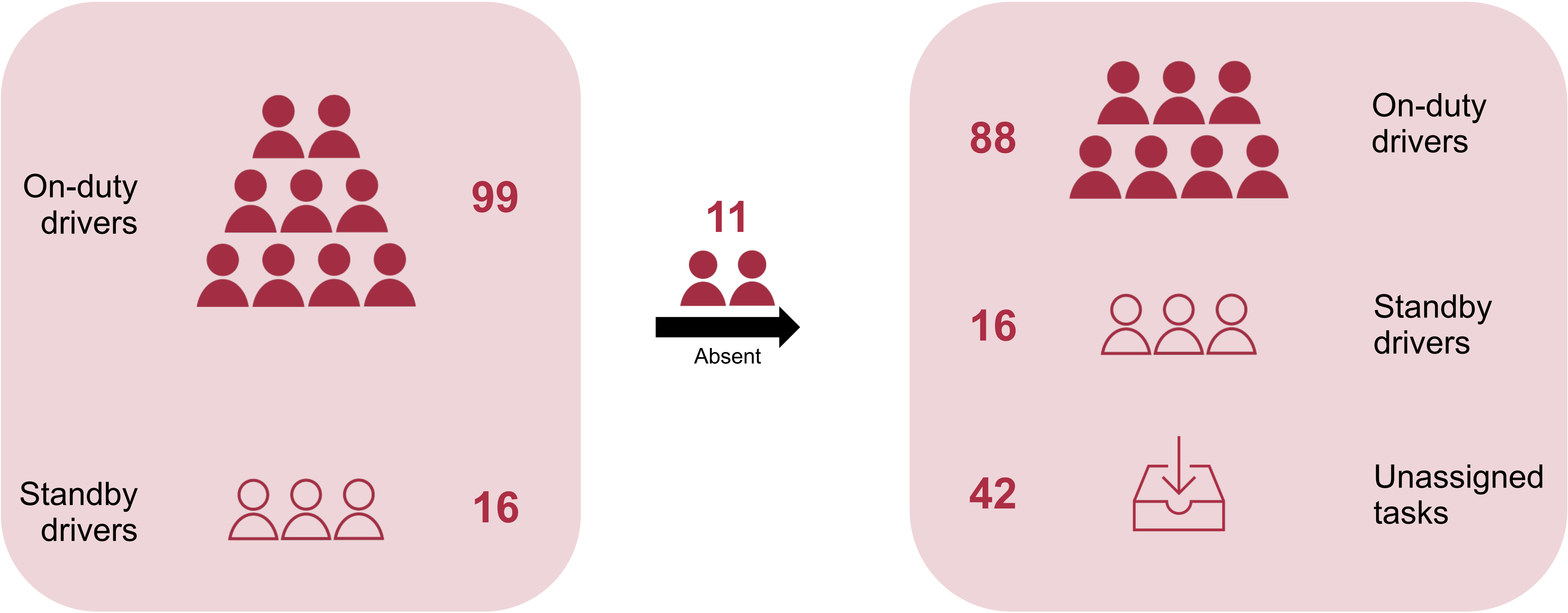
One-day Schedule



Data size	Method	Time	Space	Successful Assigned Rate
small	Tabu-Search-based Approach	9.7 s	0.16 GB	8/18
	MILP Model (Gurobi 11.0)	0.7 h	12.24 GB	10/18
medium	Tabu-Search-based Approach	12.3 s	0.18 GB	12/22
	MILP Model (Gurobi 11.0)	7.5 h	35.00 GB	17/22
large	Tabu-Search-based Approach	21.7 s	0.20 GB	29/37
	MILP Model (Gurobi 11.0)	-	out of space	-

Case Study

Approach: one-day schedule



On-duty drivers 99

Standby drivers 16

11

Absent

88 On-duty drivers

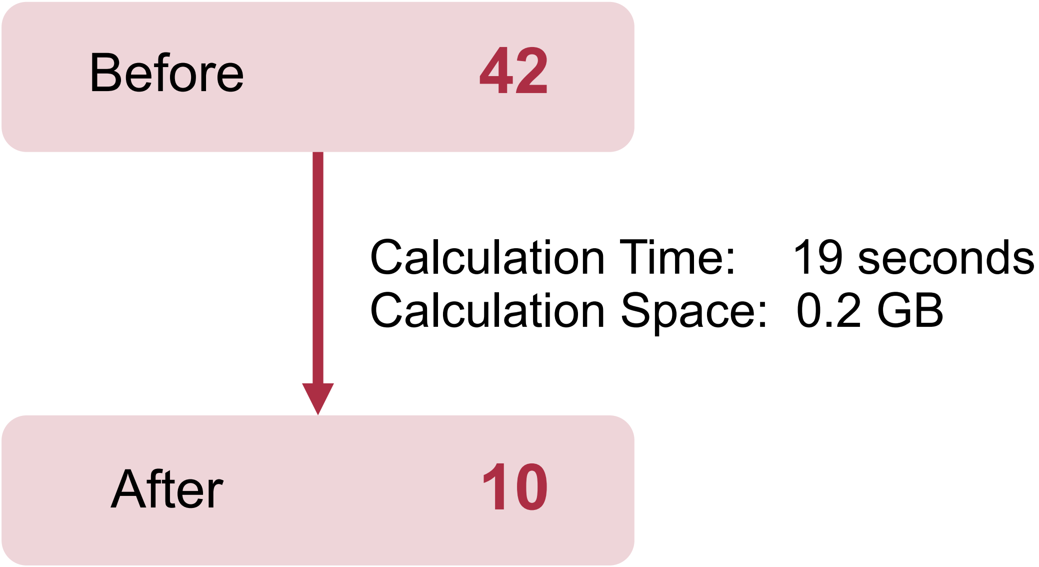
16 Standby drivers

42 Unassigned tasks

Case Study

Approach: one-day schedule

Unassigned tasks



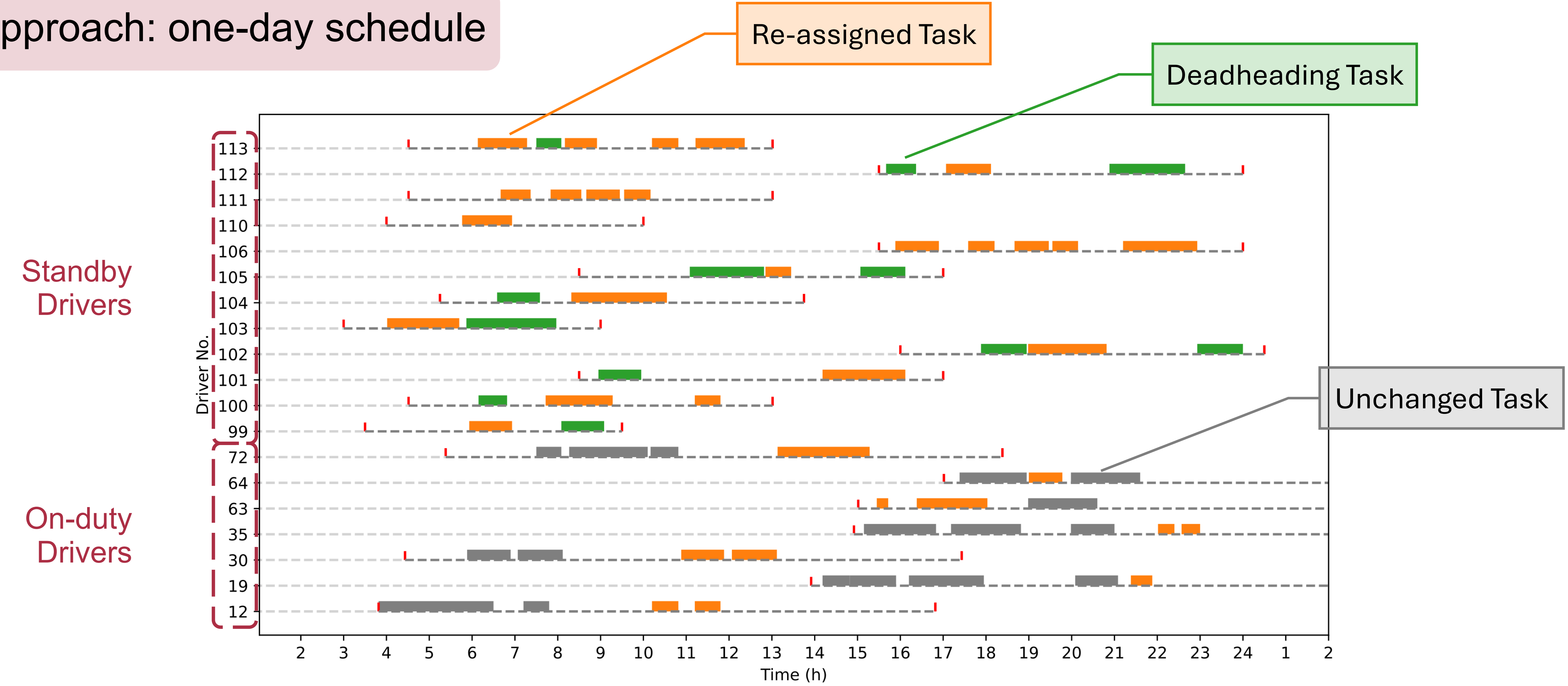
Before **42**

Calculation Time: 19 seconds
Calculation Space: 0.2 GB

After **10**

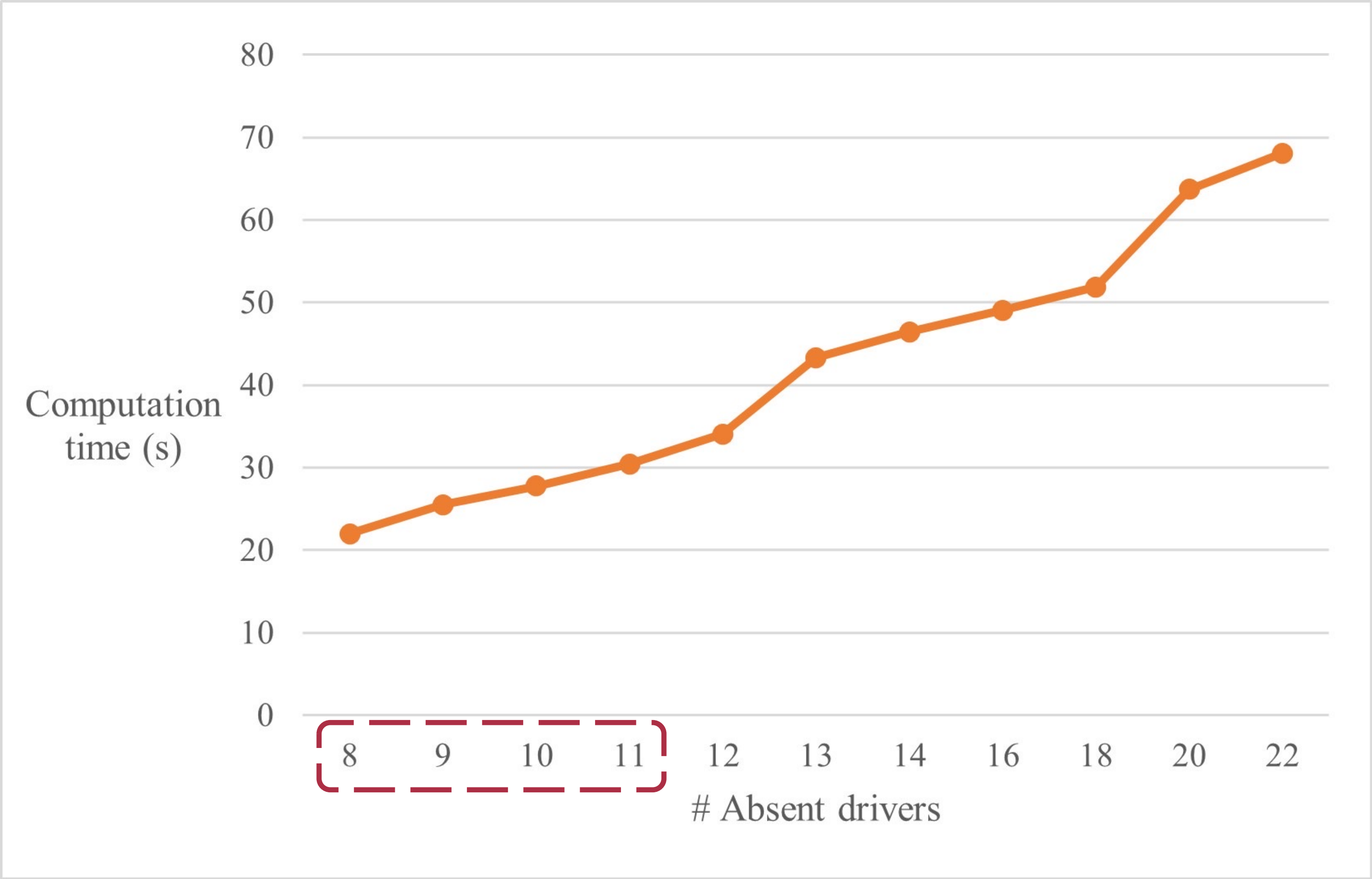
Case Study

Approach: one-day schedule



Case Study

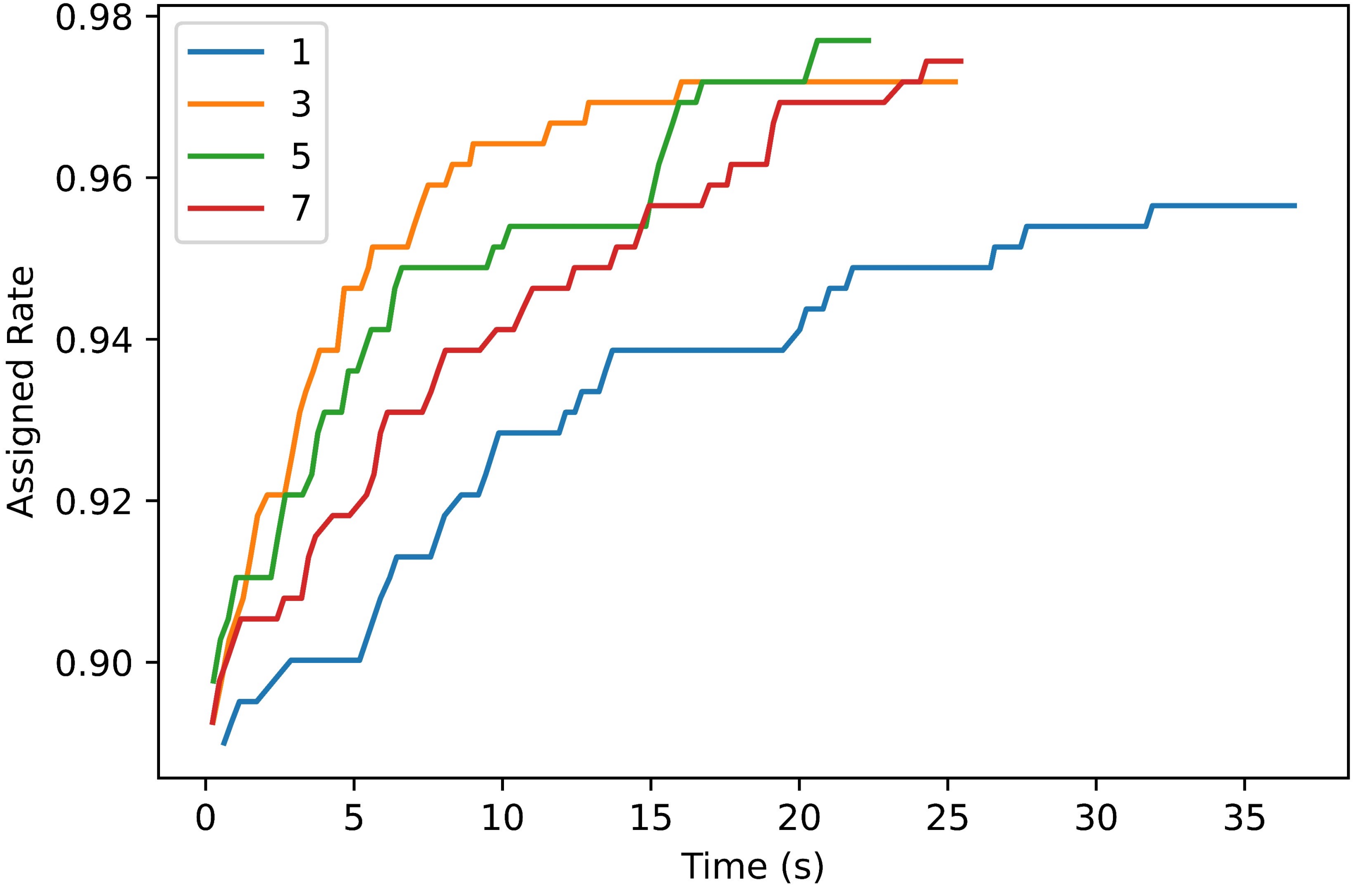
Approach: performance



Case Study

Approach: performance

Assigned Rate Changes While Calculating with Different Value of n^{diff}



Level of freedom

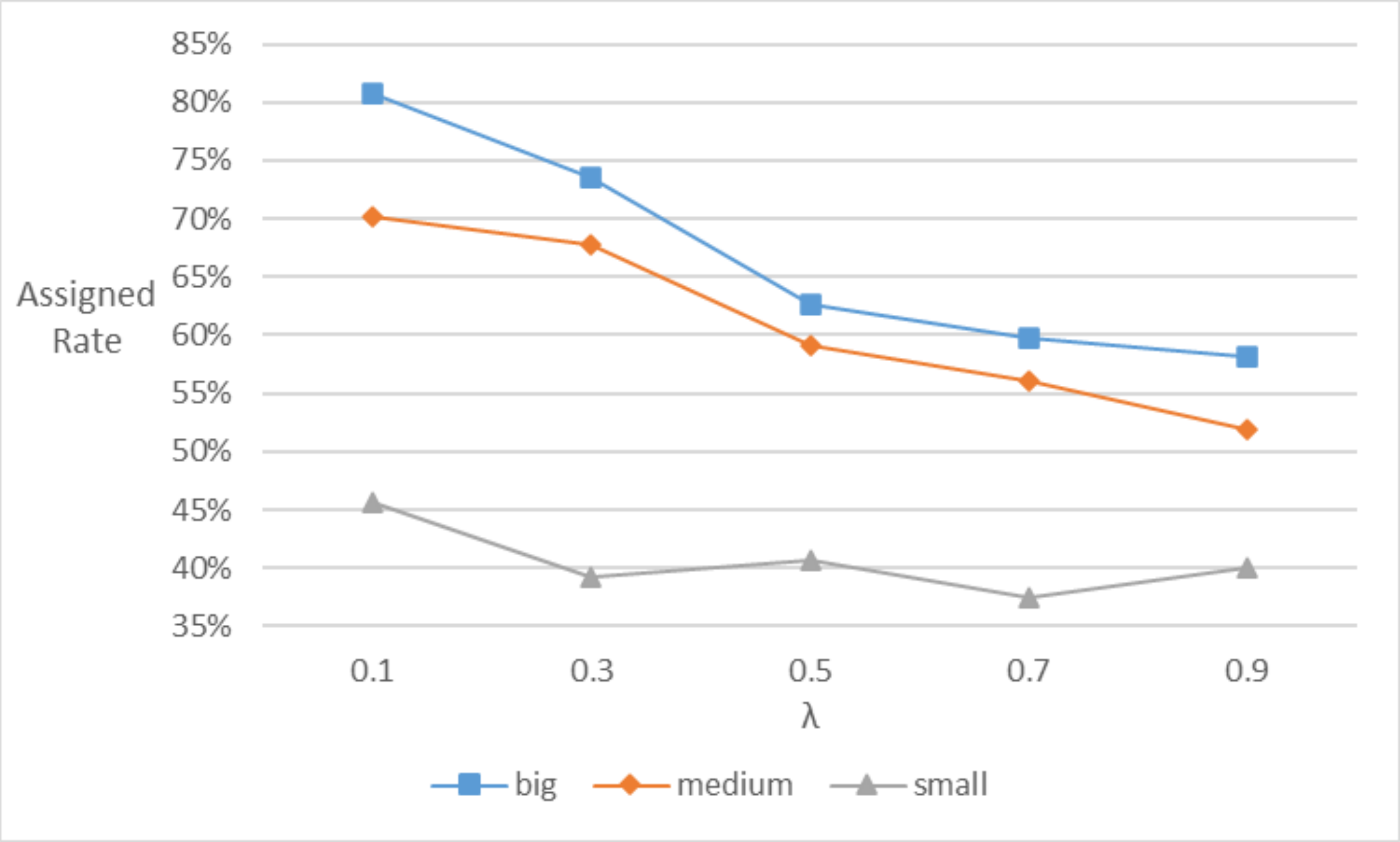
n^{diff}: maximum allowed difference between # tasks unassigned from driver d and # tasks assigned to driver d.

Case Study

LINKÖPINGS UNIVERSITET

Approach: performance

$$\text{minimize } f(x_{g,d}, z_{g,d}) = \boxed{\lambda} \sum_{g \in G} \sum_{d \in D \cap \hat{D}} z_{g,d} + (1 - \boxed{\lambda}) \sum_{d \in D^u} x_{g,d}$$



Tack så mycket!