Very Large Neighborhood Search

Nuno Antunes Ribeiro

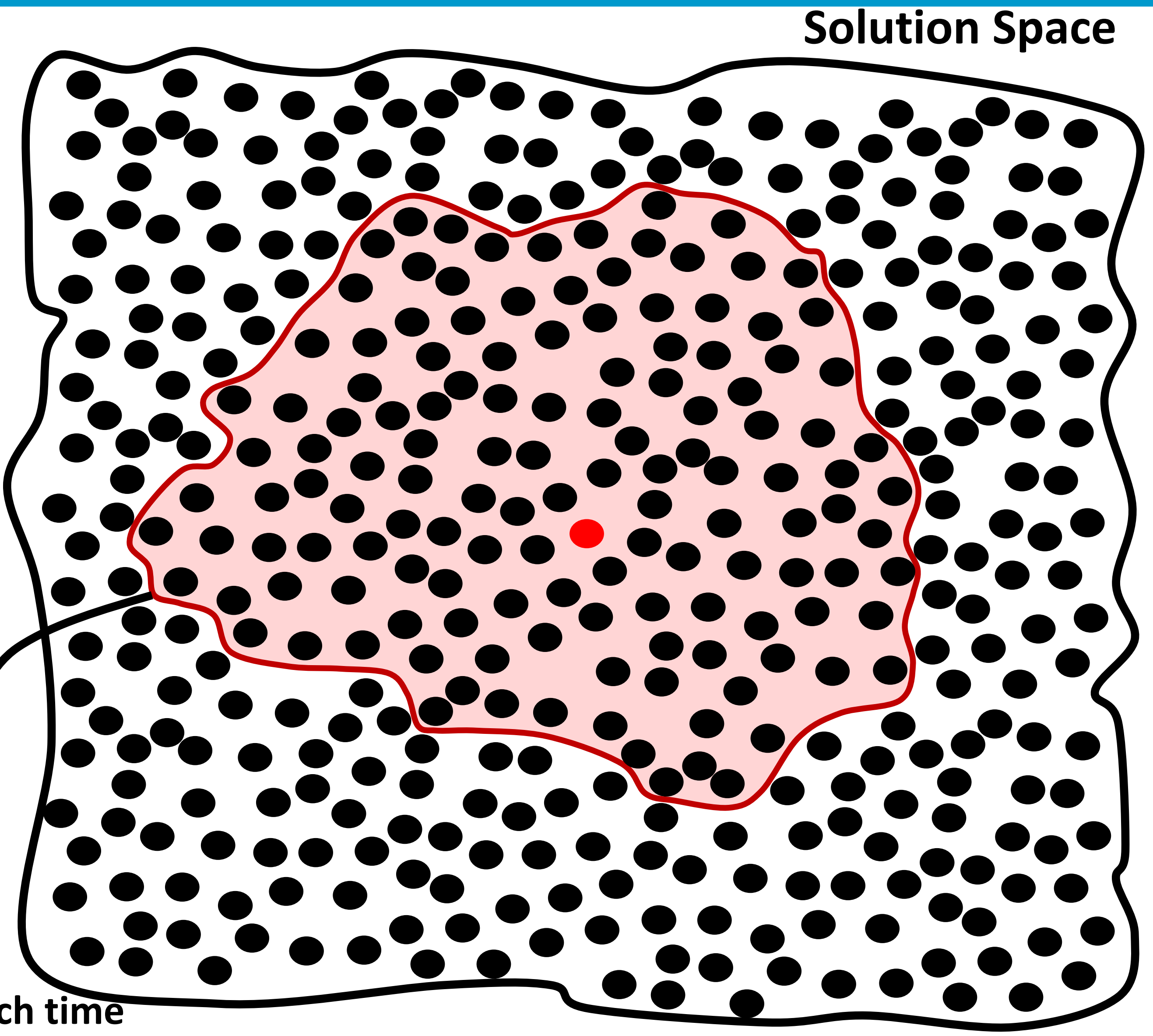
Assistant Professor

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Engineering Systems and Design

Very Large Neighborhood Search

- Explore neighbourhoods that would be impossible to analyse using exhaustive search
- Integrates exact methods of optimization and local search
  - Start with an initial feasible solution
  - Select a very large neighbourhood
  - Optimize the neighbourhood using exact methods of optimization
  - Repeat



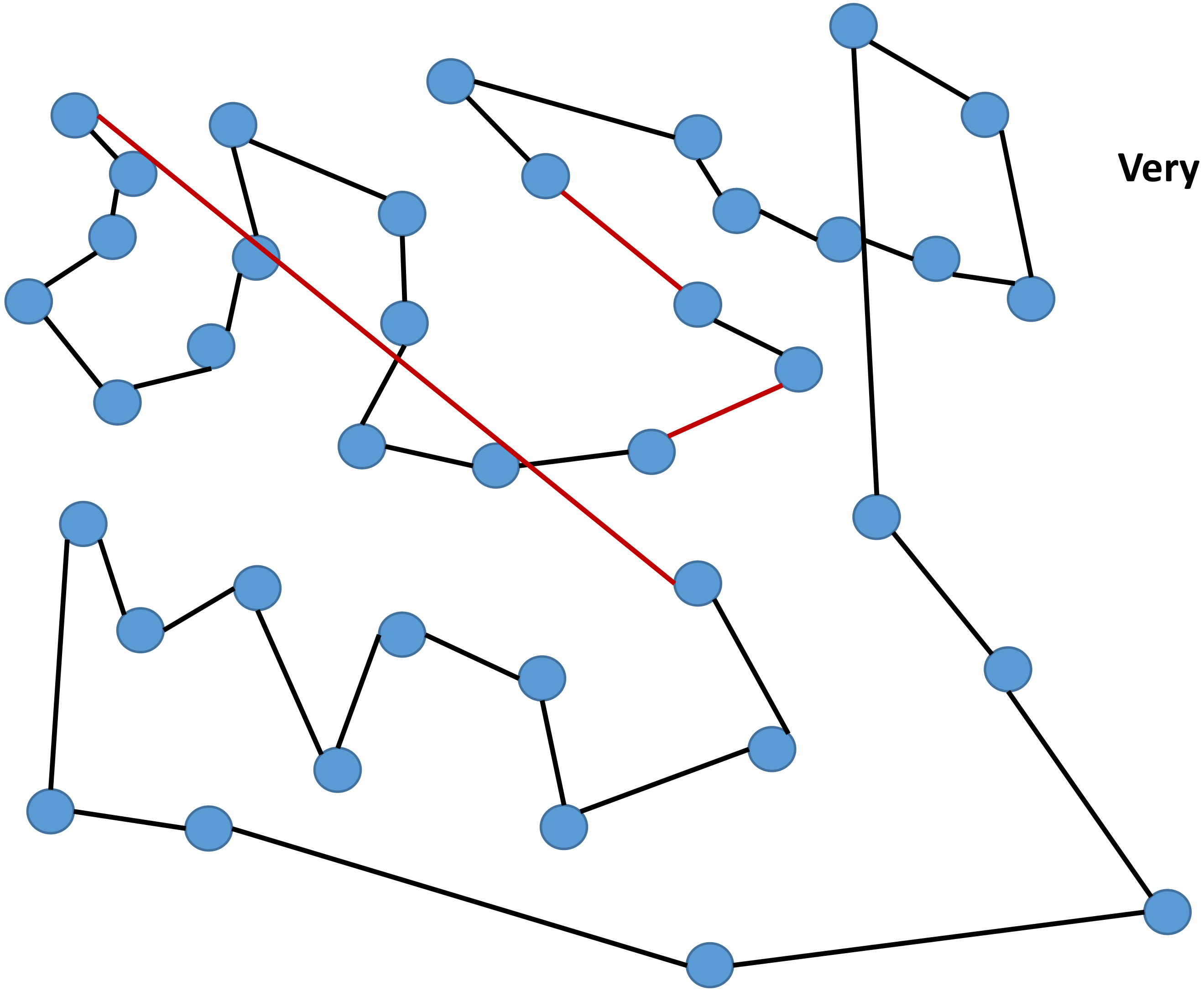
Very-large Neighbourhood
Optimize using exact methods
Exhaustive Search (best descent) would take too much time

Very Large Neighborhood Search

- Basic premise: There is a “limit” on the size and complexity to solve optimization problems using exact methods
- Runtime tends to increases **exponentially** with the size of the problem
- **Decomposition** of the problem:
  - Small enough subsets to ensure tractability of the model
  - Large enough subsets to capture interdependencies across variables

Travelling Salesman		
n	CPU Time	Opt. Value
10	1.29 s	34993
20		
25	1.21 s	39224
50	49 s	57546
75	91 s	70395
100	178 s	78357
200	2625 s	105404
500	**>6000 s**	**220704 (25.9%)**
-	-	-
-	-	-

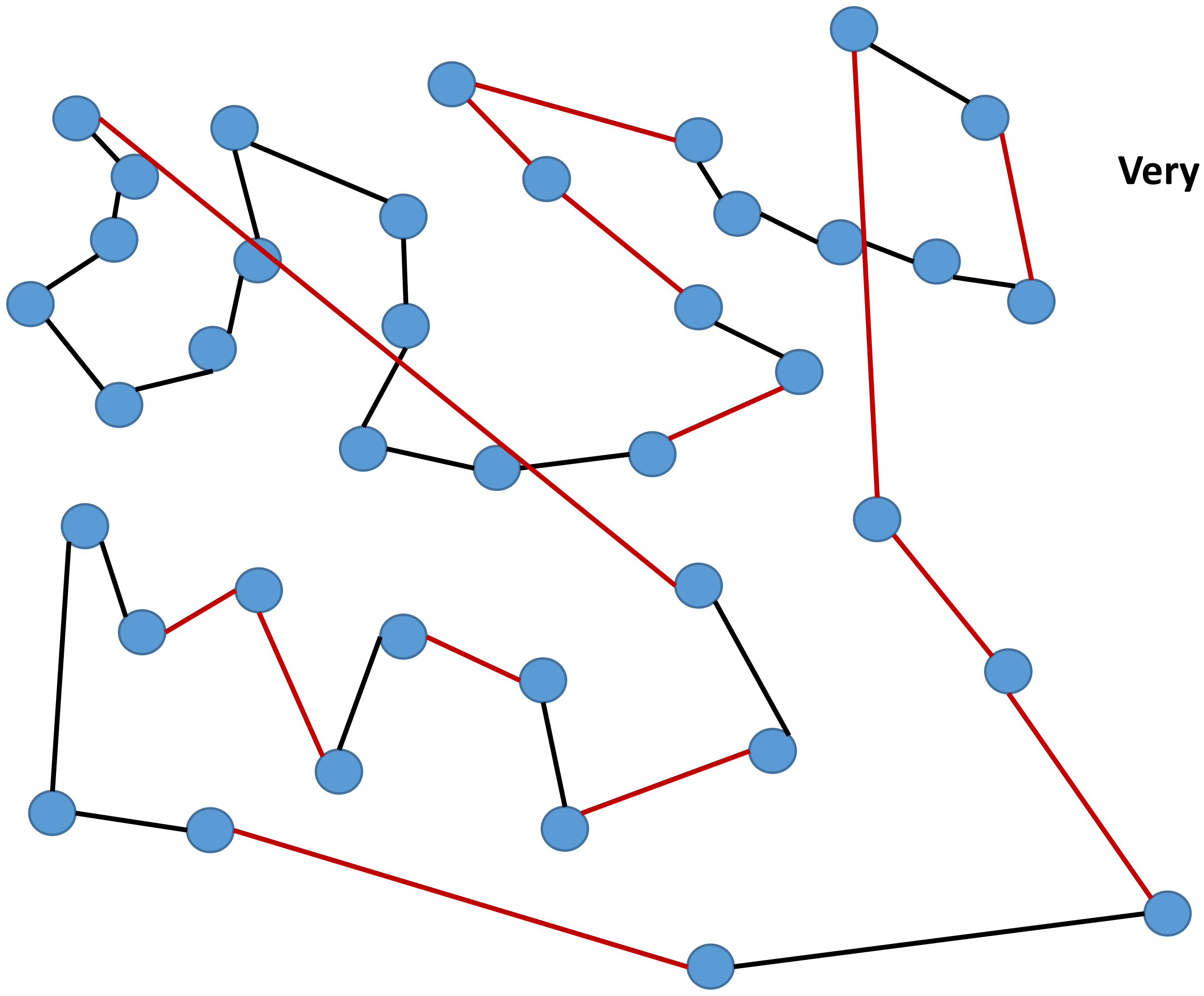
Destroy and Repair Approach



3-Opt - we select 3 arcs

Very Large Neighbourhood Search - we select n arcs

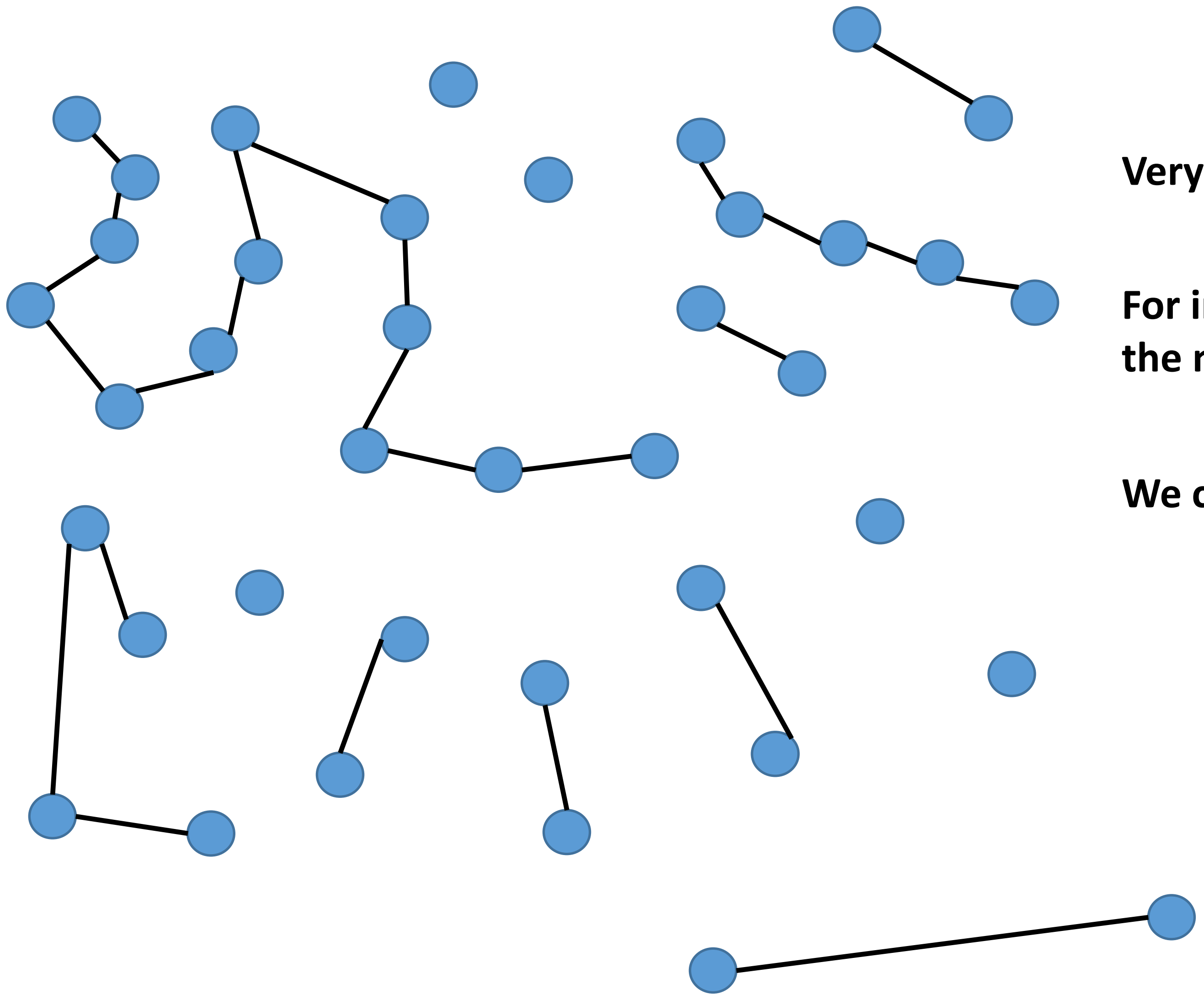
Destroy and Repair Approach



3-Opt - we select 3 arcs

Very Large Neighbourhood Search - we select n arcs

Destroy and Repair Approach



3-Opt - we select 3 arcs

Very Large Neighbourhood Search - we select n arcs

For instance, 14 arcs – that means 14! Solutions in the neighbourhood = 87178291200 solutions

We can optimize those 14 arcs using exact methods

Very Large Neighborhood Search

- In designing local search metaheuristics, there is often a compromise between the **size of the neighbourhood** to use and the **computational complexity** to explore it.

- Considering **small neighbourhoods bears two major risks.**
  - First, the algorithms may converge to **local optima**, although this can be mitigated by introducing random perturbations.
  - Second, they might be **ineffective for tightly constrained problems**, where any deviation from a feasible solution involves complex interaction effects across many decision variables.

Difficulty to escape local optima

- Multimodal with **plateaus**

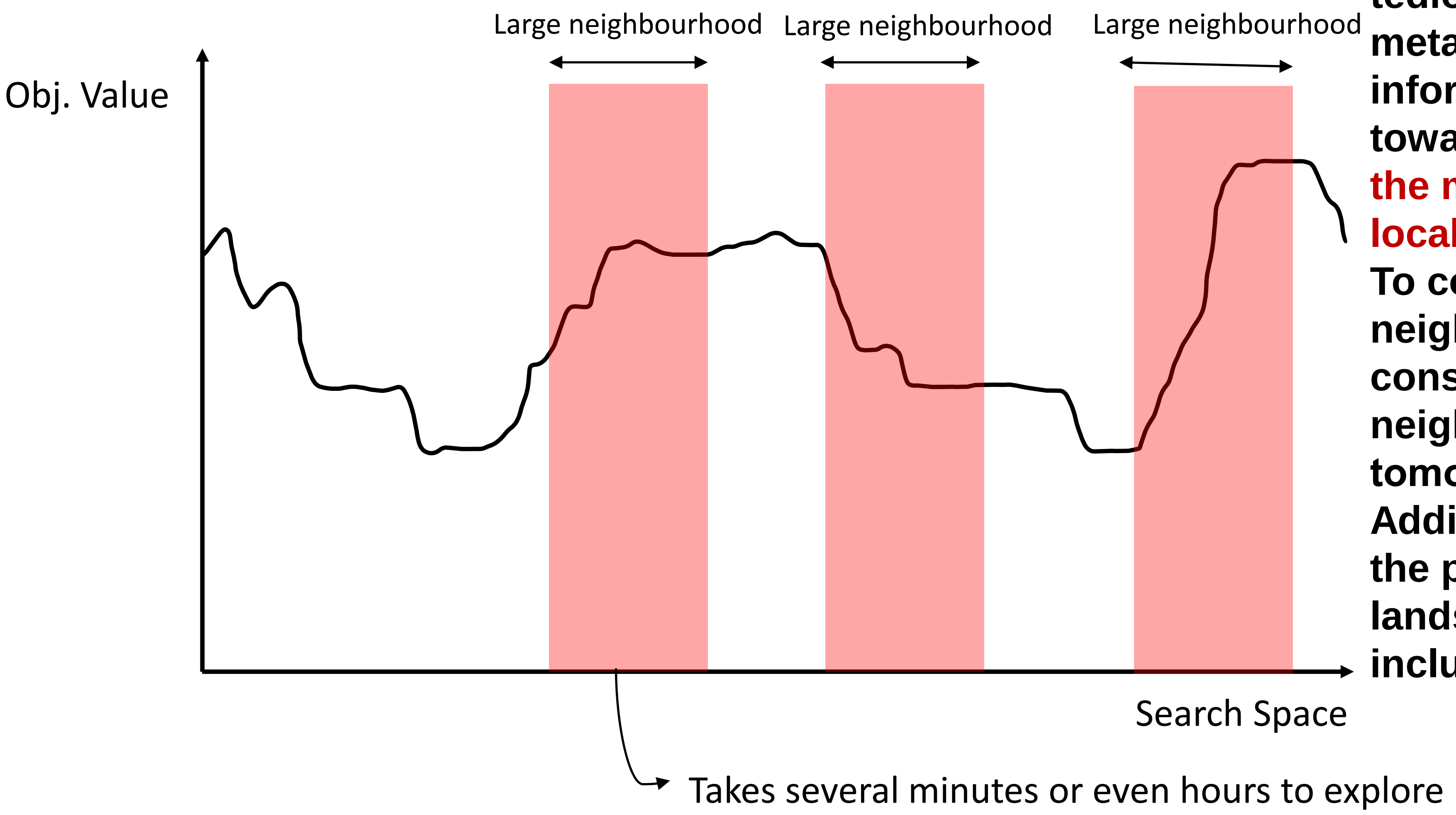
Hard to solve: Plateaus are tediously crossed by metaheuristics. Indeed, no information will guide the search toward better regions. Eventually the metaheuristic will converge to local optima solutions

To cope with these cases, larger neighbourhoods may be considered (very large neighbourhood search – tomorrow's class).

Adding a secondary objective to the problem will help to shape the landscape and different solutions included in the plateau

Difficulty to escape local optima

- Multimodal with **plateaus**



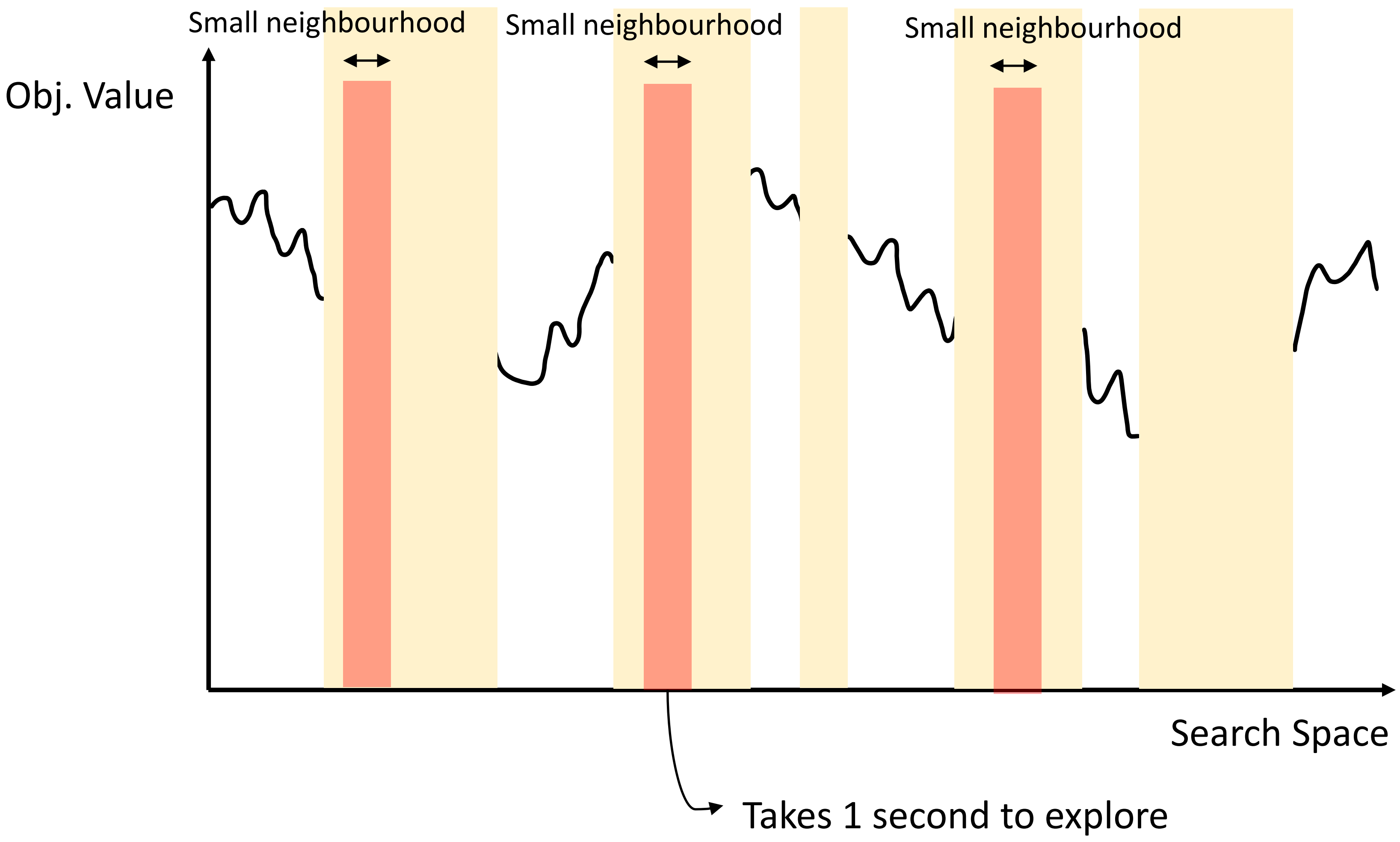
Hard to solve: Plateaus are tediously crossed by metaheuristics. Indeed, no information will guide the search toward better regions. Eventually the metaheuristic will converge to local optima solutions

To cope with these cases, larger neighbourhoods may be considered (very large neighbourhood search – tomorrow's class).

Adding a secondary objective to the problem will help to shape the landscape and different solutions included in the plateau

Highly constrained landscapes

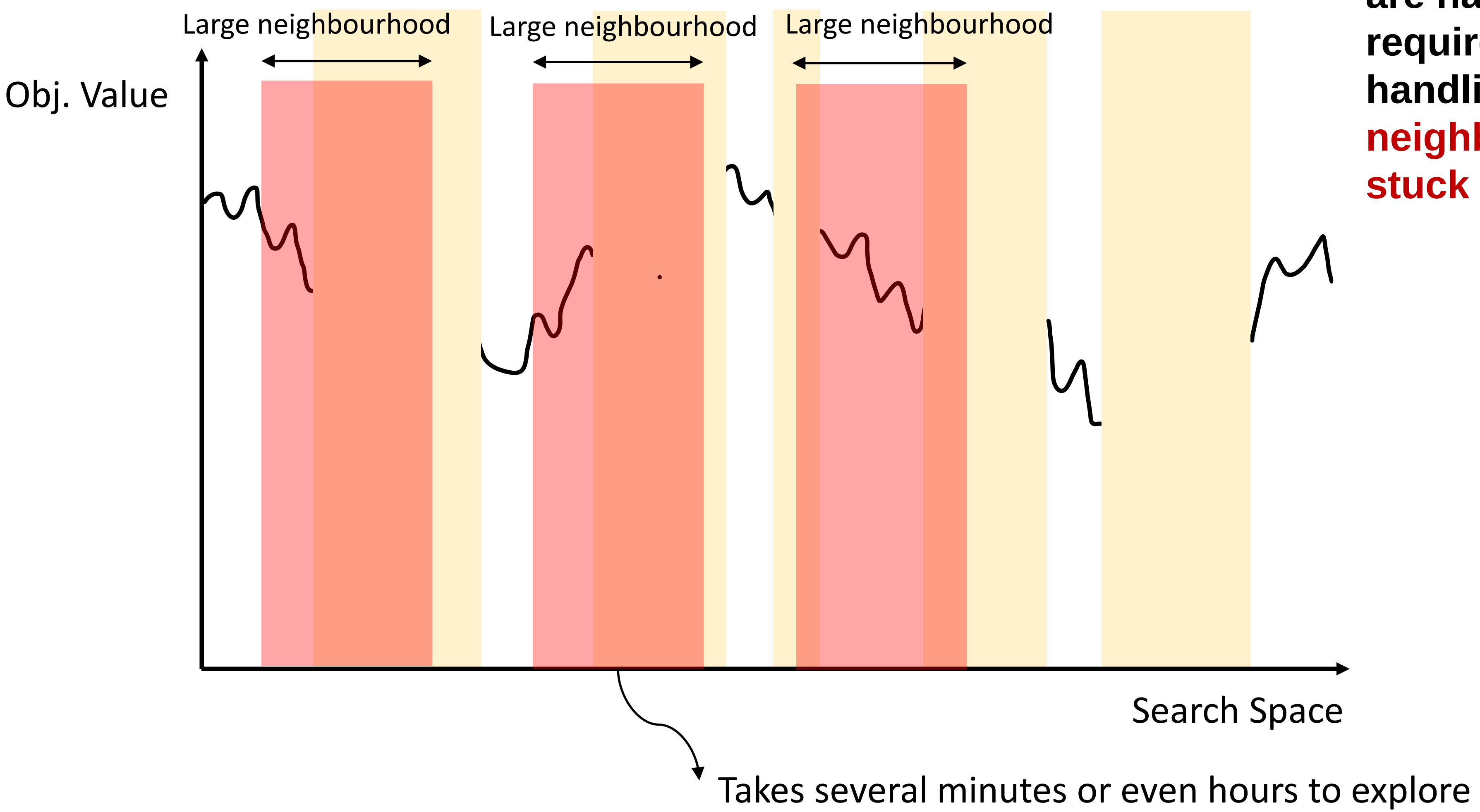
- Constrained Landscape



Highly constrained landscapes are harder to solve, they may require the use of constraint handling techniques, and/or larger neighbourhoods to avoid getting stuck in a constraint region

Highly constrained landscapes

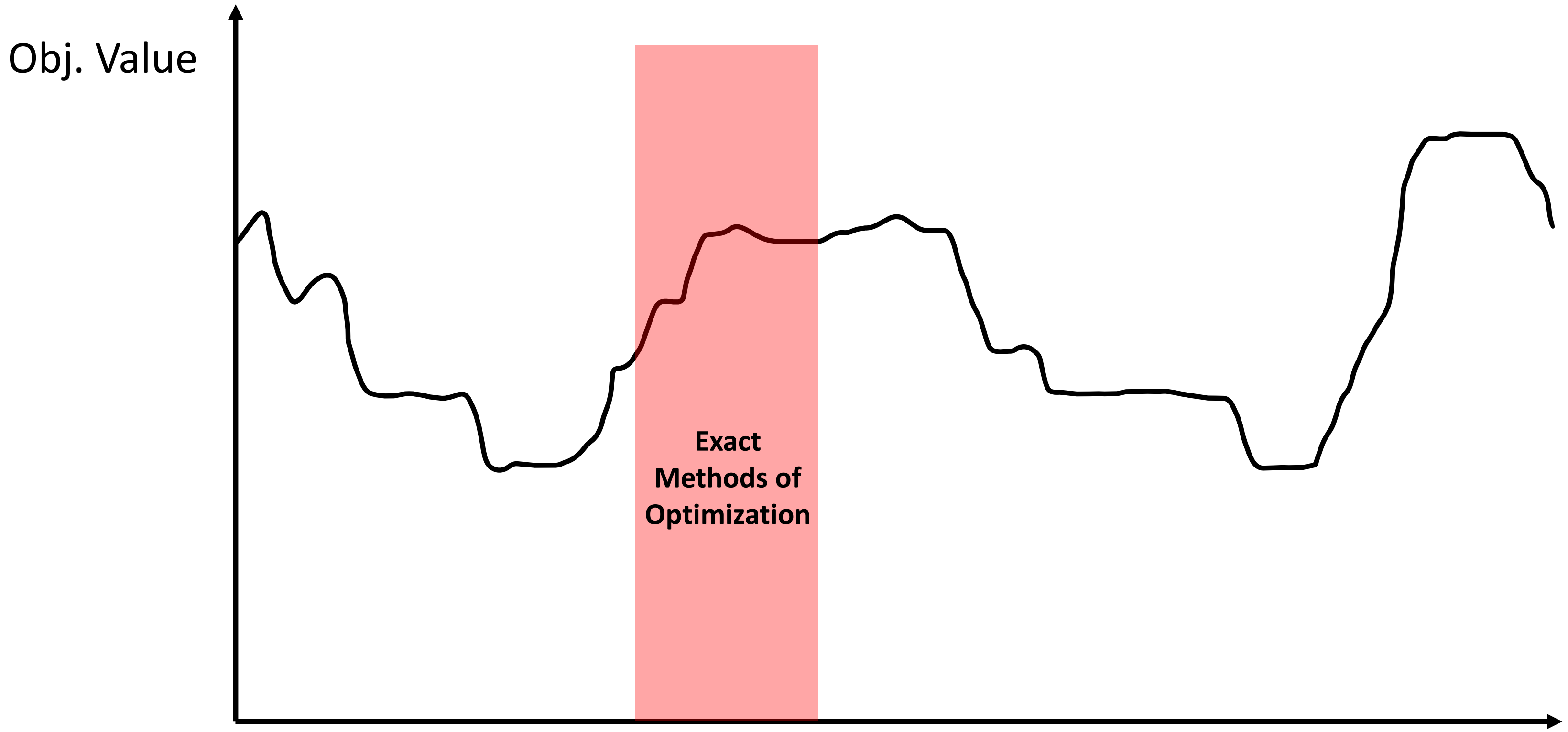
- Constrained Landscape



Highly constrained landscapes are harder to solve, they may require the use of constraint handling techniques, and/or larger neighbourhoods to avoid getting stuck in a constraint region

Exhaustive Search vs Exact Methods

- Exhaustively search large neighbourhoods is time consuming. We can rely on exact methods o optimization to explore these neighbourhoods



VLNS in Airport Scheduling

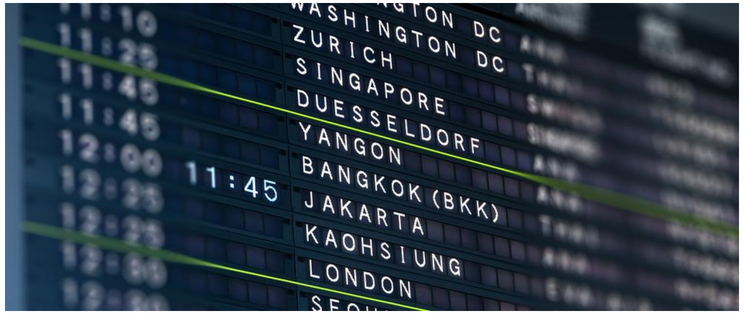
Nuno Antunes Ribeiro

Assistant Professor

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN | Engineering Systems and Design

Airport Congestion Mitigation

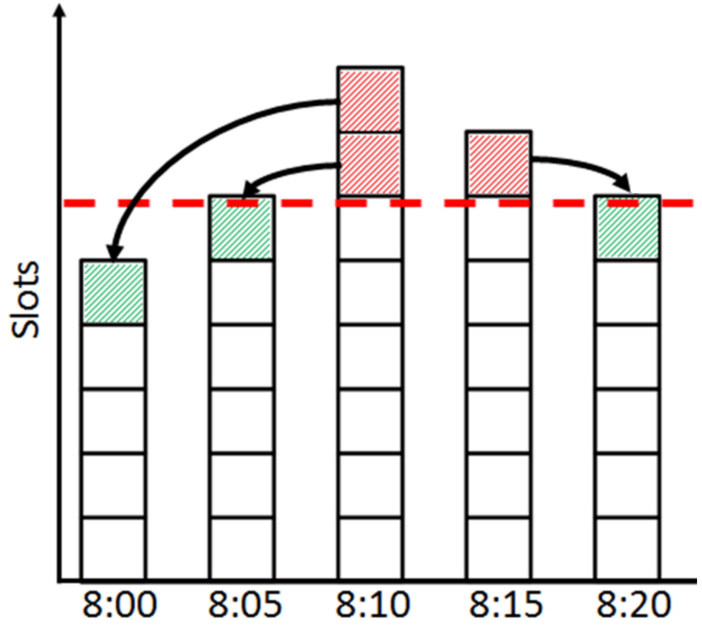
- **strategic interventions (several months in advance)**, aimed at limiting the demand for airport access through slot control mechanisms;
- **tactical interventions (hours before operations)**, aimed at efficiently delaying flights on the ground before take-off to reduce chances of airborne delays at periods of expected airport capacity shortage;
- **operational interventions (real time)**, aimed at controlling real-time air traffic operations through flow management solutions that ensure that aircraft fly safely and efficiently throughout the airspace.



IATA Slot Allocation Process

1 Year	5 months	5 months	4 months	up to day
Declared Capacity	Slot Requests	**Initial Slot Allocation**	Slot Conference	Changes

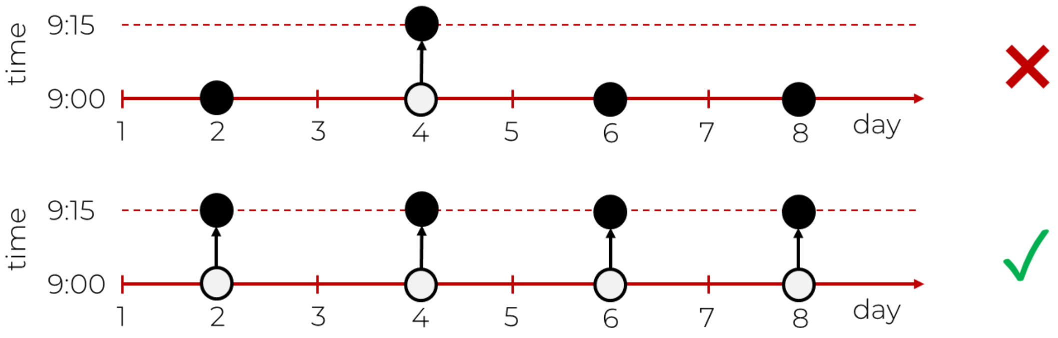
- **IATA Guidelines**
- **Primary Criteria**
  - Slot regularity: Allocate all slots within the same request at the same time of day, throughout the season
  - Connectivity: Maintain appropriate aircraft connecting times
- **Slot priorities**
  - Historic slots
  - Change-to-historic slots
  - New entrants
  - Other slots

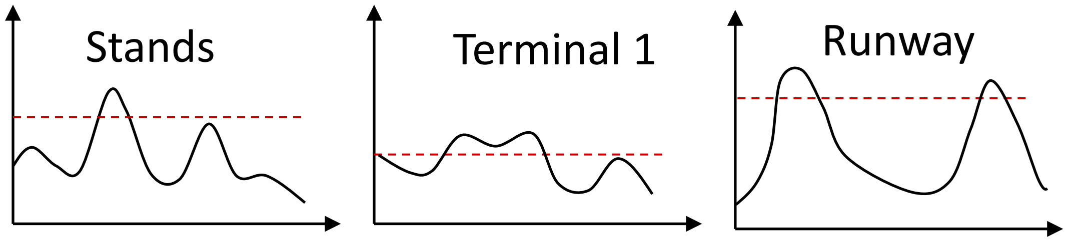


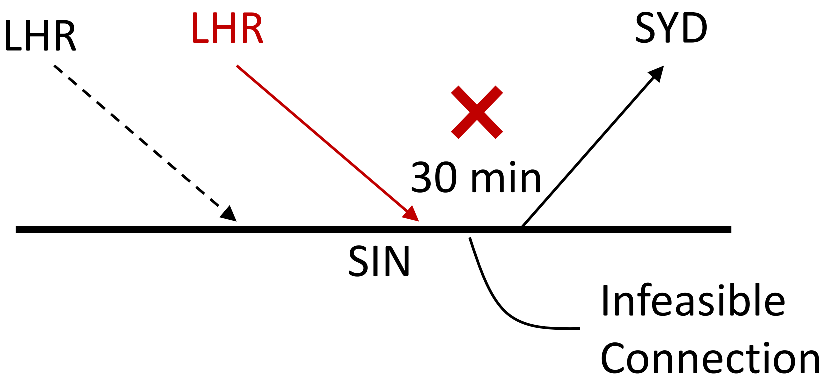
Rules and requirements result in complex slot allocation problems

Slot Allocation Rules

- **Series of flights** – flight operation requested to take place during several days of the season – these flights should be allocated to the same slot time.
- **Different types of capacities** - capacities for the runway, terminal, stands ; capacities per hour , per 15 min, etc.
- **Flight connections** – we should ensure that passenger connections remain feasible







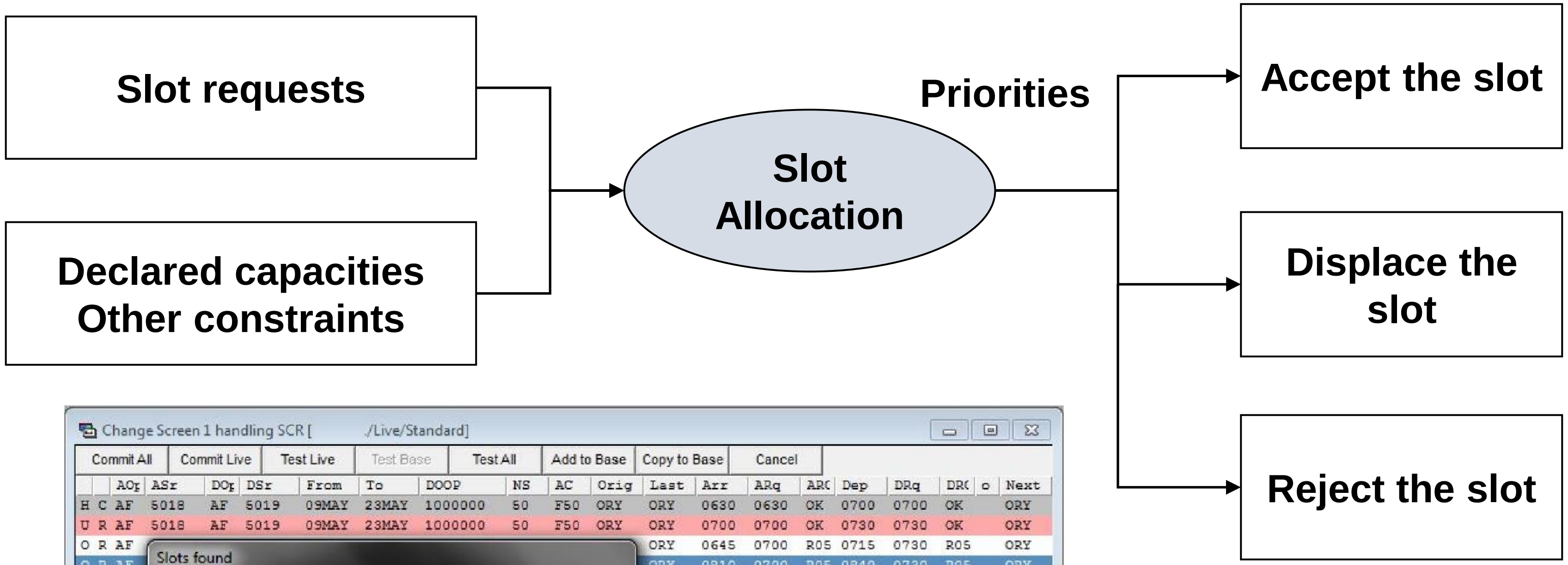
Slot Conference

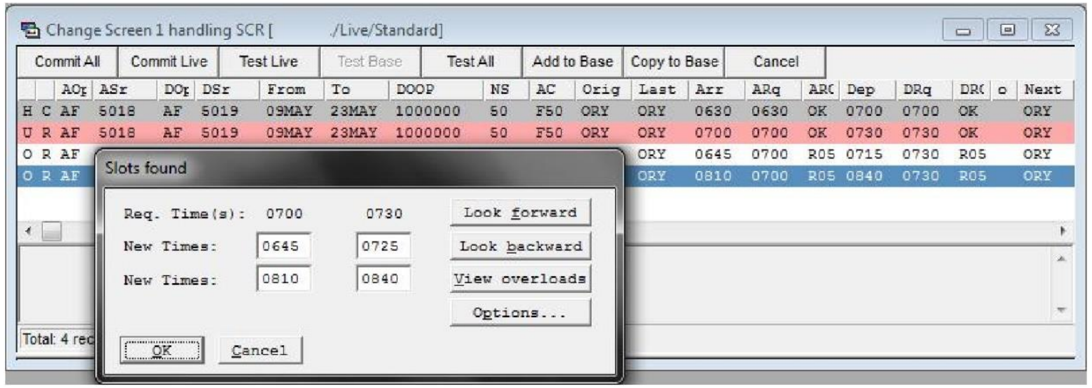
Airlines' room

138th Slot Coference – Hamburg, 2016

Slot Coordinators' room

Current Allocation Process





Treatment of slots, one by one, provides limited visibility into full set of requests and interdependencies between decisions

Example

*Capacity: 1 flight every 5 minutes

Slot Request	Day1	Day 2	Day3	Day 4	Day 5
1	9:00	9:00	9:00		9:00
2		9:00	9:00		9:00
3			9:00	9:00	
4	9:00			9:00	

Example

*Capacity: 1 flight every 5 minutes

Slot Request	Day1	Day 2	Day3	Day 4	Day 5	Total Displacement
1	9:00	9:00	9:00		9:00	0x4=0
2		9:00	9:00		9:00	
3			9:00	9:00		
4	9:00			9:00		

Example

*Capacity: 1 flight every 5 minutes

Slot Request	Day1	Day 2	Day3	Day 4	Day 5	Total Displacement
1	9:00	9:00	9:00		9:00	0x4=0
2		9:05	9:05		9:05	5x3=15
3			9:00	9:00		
4	9:00			9:00		

Example

*Capacity: 1 flight every 5 minutes

Slot Request	Day1	Day 2	Day3	Day 4	Day 5	Total Displacement
1	9:00	9:00	9:00		9:00	0x4=0
2		9:05	9:05		9:05	5x3=15
3			8:55	8:55		5x2=10
4	9:00			9:00		

Example

*Capacity: 1 flight every 5 minutes

Slot Request	Day1	Day 2	Day3	Day 4	Day 5	Total Displacement
1	9:00	9:00	9:00		9:00	0x4=0
2		9:05	9:05		9:05	5x3=15
3			8:55	8:55		5x2=10
4	9:05			9:05		5x2=10
						35 min

Example

*Capacity: 1 flight every 5 minutes

Slot Request	Day1	Day 2	Day3	Day 4	Day 5	Total Displacement
1	9:00	9:00	9:00		9:00	0x4=0
2		9:05	9:05		9:05	5x3=15
3			8:55	8:55		5x2=10
4	9:05			9:05		5x2=10
						35 min

Optimal Solution

Slot Request	Day1	Day 2	Day3	Day 4	Day 5	Total Displacement
1	9:05	9:05	9:05		9:05	5x4=20
2		9:00	9:00		9:00	0x3=0
3			8:55	8:55		5x2=10
4	9:00			9:00		0x2=0
						30 min

The Optimization Model

- minimize $$w_1 \sum_{i\in \boldsymbol{S}} \sum_{d\in \boldsymbol{D}} B_{id} Z_i + w_2 \max_{i\in \boldsymbol{S}} |X_i| + w_3 \sum_{i\in \boldsymbol{S}} \sum_{d\in \boldsymbol{D}} B_{id} |X_i| + \sum_{i\in \boldsymbol{S}} \sum_{d\in \boldsymbol{D}} B_{id} |W_i|$$
- subject to:

$$Y_{iT} = Z_i \quad ,\forall i \in \boldsymbol{S}$$

$$\sum_{t\in \boldsymbol{T}} (Y_{it} - A_{it}) = |X_i| + \sum_{t\in \boldsymbol{T}} (1 - A_{it}) \times Z_i \quad ,\forall i \in \boldsymbol{S}$$

$$|W_i| \geq Y_{it} - A_{it} + Z_i \quad \forall i \in \boldsymbol{S}, t \in \boldsymbol{T}$$

$$|W_i| \geq -Y_{it} + A_{it} \quad \forall i \in \boldsymbol{S}$$

$$\sum_{i\in \boldsymbol{S}_{arr}} \sum_{t\in \boldsymbol{T}_c^s} (Y_{it} - Y_{i,t+1}) B_{id} \leq C_{sdc}^{arr} \quad ,\forall s \in \boldsymbol{T}_c, d \in \boldsymbol{D}, c \in \boldsymbol{C}$$

$$\sum_{i\in \boldsymbol{S}_{dep}} \sum_{t\in \boldsymbol{T}_c^s} (Y_{it} - Y_{i,t+1}) B_{id} \leq C_{sdc}^{dep} \quad ,\forall s \in \boldsymbol{T}_c, d \in \boldsymbol{D}, c \in \boldsymbol{C}$$

$$\sum_{i\in \boldsymbol{S}} \sum_{t\in \boldsymbol{T}_c^s} (Y_{it} - Y_{i,t+1}) B_{id} \leq C_{sdc}^{T} \quad ,\forall s \in \boldsymbol{T}_c, d \in \boldsymbol{D}, c \in \boldsymbol{C}$$

$$\sum_{t\in \boldsymbol{T}} (Y_{jt} - Y_{it}) - \sum_{t\in \boldsymbol{T}} (A_{jt} - A_{it}) \geq T^{\min} - T(Z_i + Z_j) \quad ,\forall i, j \in \boldsymbol{P}$$

$$\sum_{t\in \boldsymbol{T}} (Y_{jt} - Y_{it}) - \sum_{t\in \boldsymbol{T}} (A_{jt} - A_{it}) \leq T^{\max} - T(Z_i + Z_j) \quad ,\forall i, j \in \boldsymbol{P}$$

$$\sum_{i\in \boldsymbol{S}_{arr}} \sum_{t\in \boldsymbol{T}_c^s} \sum_{k\in \boldsymbol{K}} (Y_{it} - Y_{i,t+1}) B_{id} I_{ik} S_i LF_i \leq P_{sdck}^{arr} \quad ,\forall s \in \boldsymbol{T}_c, d \in \boldsymbol{D}, c \in \boldsymbol{C}$$

$$\sum_{i\in \boldsymbol{S}_{dep}} \sum_{t\in \boldsymbol{T}_c^s} \sum_{k\in \boldsymbol{K}} (Y_{it} - Y_{i,t+1}) B_{id} I_{ik} S_i LF_i \leq P_{sdck}^{dep} \quad ,\forall s \in \boldsymbol{T}_c, d \in \boldsymbol{D}, c \in \boldsymbol{C}$$

$$\sum_{i\in \boldsymbol{S}} \sum_{t\in \boldsymbol{T}_c^s} \sum_{k\in \boldsymbol{K}} (Y_{it} - Y_{i,t+1}) B_{id} I_{ik} S_i LF_i \leq P_{sdck}^{T} \quad ,\forall s \in \boldsymbol{T}_c, d \in \boldsymbol{D}, c \in \boldsymbol{C}$$

$$\sum_{i\in \boldsymbol{S}_{arr}} (Y_{it} - Y_{i,t+1}) B_{id} - \sum_{i\in \boldsymbol{S}_{dep}} (Y_{it}^{dep} - Y_{i,t+1}^{dep}) B_{id} + Q_{t-1,\mathrm{d}} = Q_{td} \quad ,\forall t \in \boldsymbol{T}, d \in \boldsymbol{D}$$

$$\sum_{i\in \boldsymbol{S}_{arr}} (Y_{i1} - Y_{i2}) B_{id} - \sum_{i\in \boldsymbol{S}_{dep}} (Y_{i1} - Y_{i2}) B_{id} + Q_{\mathrm{T,d}-1} = Q_{1d} \quad ,\forall d \in \boldsymbol{D}$$

$$Q_{td} \leq C_{td}^{apron} \quad ,\forall t \in \boldsymbol{T}, d \in \boldsymbol{D}$$

$$\sum_{t\in \boldsymbol{T}} (Y_{jt} - Y_{it} + D_{ij}) \geq T_k^{\max} \quad ,\forall i, j \in \boldsymbol{P}, k \in \boldsymbol{K}$$

$$\sum_{i\in \boldsymbol{S}} \sum_{t\in \boldsymbol{T}_c^s} \sum_{k\in \boldsymbol{K}} (Y_{it} - Y_{i,t+1}) B_{id} I_{ik} S_i LF_i \leq P_{sdck}^{T} \quad ,\forall s \in \boldsymbol{T}_c, d \in \boldsymbol{D}, c \in \boldsymbol{C}$$

Ribeiro, N. A., Jacquillat, A., & Antunes, A. P. (2019). A large-scale neighborhood search approach to airport slot allocation. *Transportation Science*, *53*(6), 1772-1797.

Exact Methods of Optimization

Airport	Madeira (FNC)	Porto (OPO)	Lisbon (LIS)	São Paulo (GRU)	Singapore (SIN)	Paris (CDG)
Season	S2014	S2014	S2015	W2018	**S2019**	S2018
Capacity per hour	14	20	38	53	**70**	110
No. slots	13696	41547	114176	161469	**315226**	348977
CPU Time	1 min	5 min	**>7 days**	**Memory Error**	**Memory Error**	**Memory Error**

Results

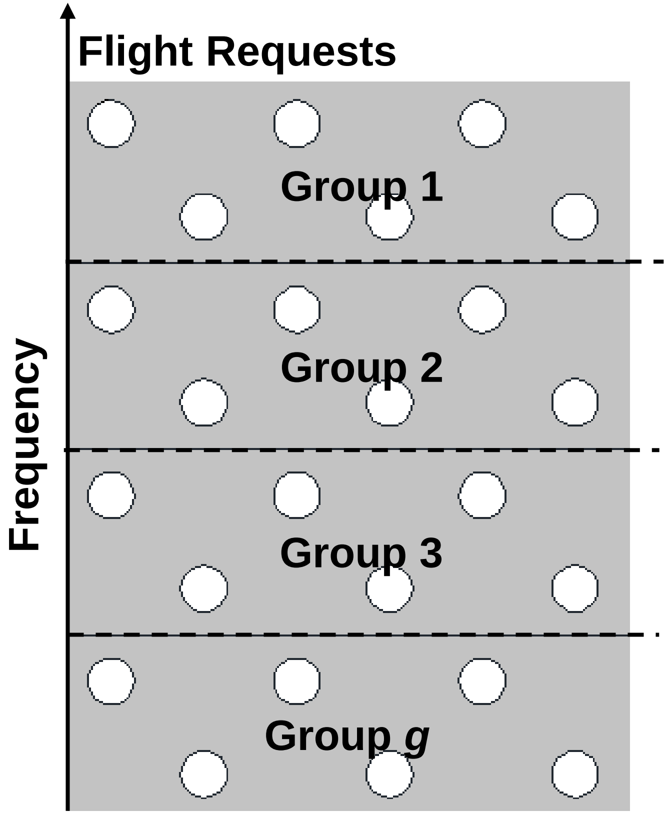
Main Indicators	Porto Airport		São Paulo Airport	
	Slot. Coord.	Opt. Model	Slot. Coord.	Opt. Model
Max Displacement (min)	80	55 **-31%**	390	295 **-24%**
Total Displacement (min)	53.2k	38,6k **-27%**	3270k	2,682k **-18%**
No. Slots Displaced	2.5k	2,3k **-7%**	63,8k	32,275 **-49%**

VLNS – Slot Allocation Optimization

- Basic premise: There is a “limit” on the size and complexity of the slot allocation problem that can be solved with exact methods
- Decomposition of the problem to allocate a subset of all slot requests at each iteration
  - Small enough subsets to ensure tractability of the model
  - Large enough subsets to capture interdependencies across slot requests
- Algorithm using large-scale neighbourhood search principles, based on “destroy and repair” approach
  - Constructive Greedy Algorithm: Generates a feasible solution quickly
  - Local Search Algorithm : Finds iteratively local improvements within a large neighbourhood, while maintaining global feasibility

Constructive Greedy Heuristic

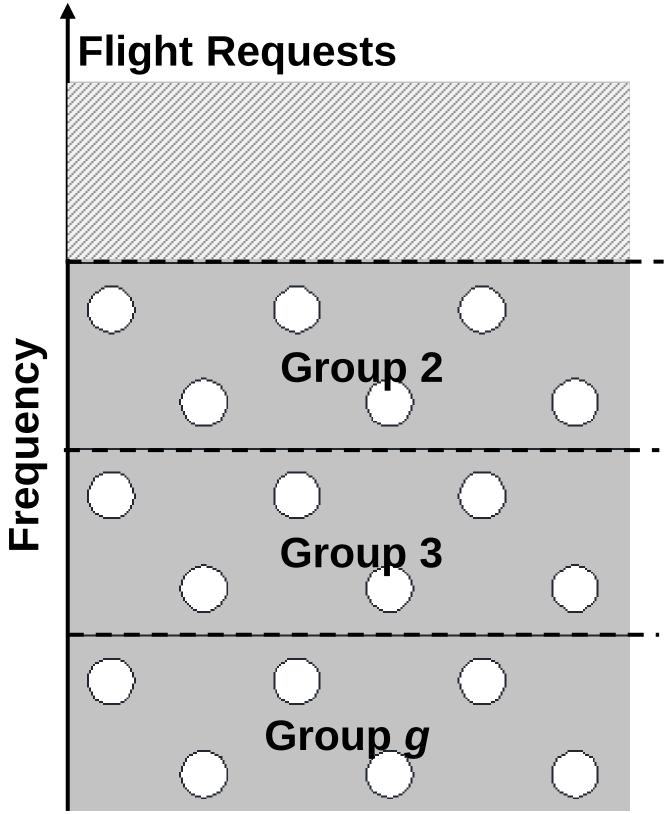
- **Premise:** Flights with higher frequency (i.e., taking place on more days during a season) are “harder” to allocate
  - Allocation of flights by decreasing order of frequency



1. **Arrange flights into groups of decreasing frequency**
2. **Allocate flights to each group sequentially**
   Solve model for Group 1
   Solve model for Group 2
   Etc.
3. **Other adjustments to maintain global feasibility**
   Update capacity values
   Ensure feasibility of connections
   Ensure priorities among slot classes
   Etc.

Constructive Greedy Heuristic

- **Premise:** Flights with higher frequency (i.e., taking place on more days during a season) are “harder” to allocate
  - Allocation of flights by decreasing order of frequency



1. **Arrange flights into groups of decreasing frequency**
2. **Allocate flights to each group sequentially**

   Solve model for Group 1

   Solve model for Group 2

   Etc.
3. **Other adjustments to maintain global feasibility**

   Update capacity values

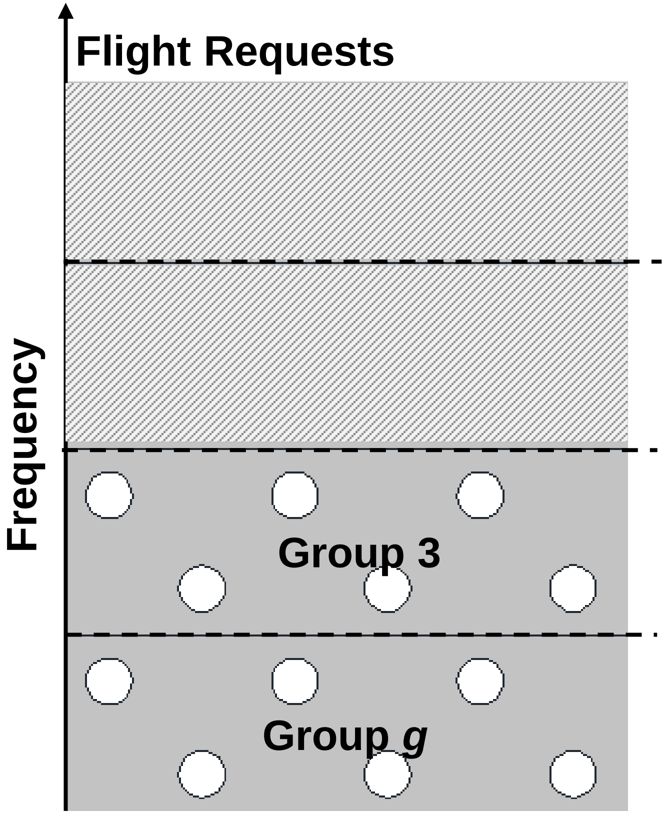
   Ensure feasibility of connections

   Ensure priorities among slot classes

   Etc.

Constructive Greedy Heuristic

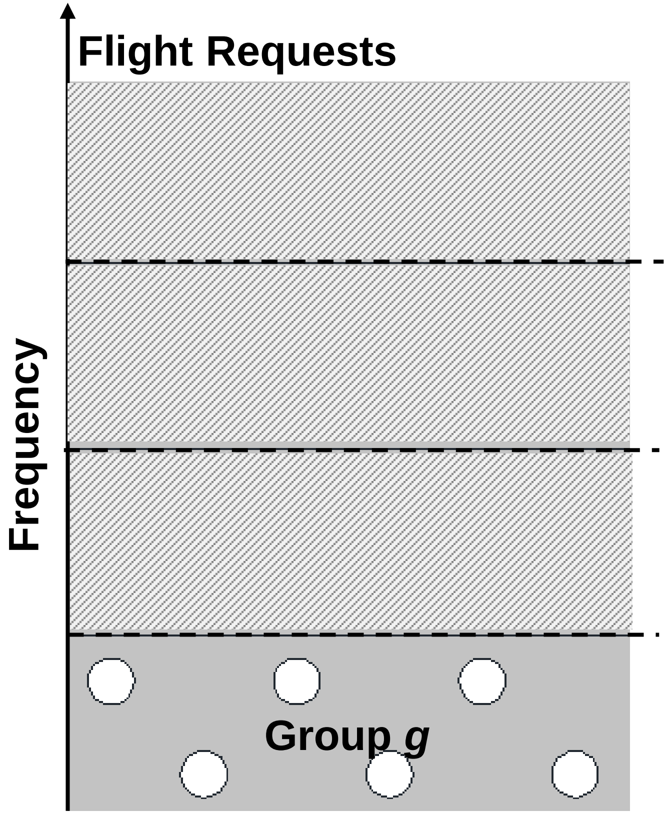
- **Premise:** Flights with higher frequency (i.e., taking place on more days during a season) are "harder" to allocate
  - Allocation of flights by decreasing order of frequency



1. **Arrange flights into groups of decreasing frequency**
2. **Allocate flights to each group sequentially**
   Solve model for Group 1
   Solve model for Group 2
   Etc.
3. **Other adjustments to maintain global feasibility**
   Update capacity values
   Ensure feasibility of connections
   Ensure priorities among slot classes
   Etc.

Constructive Greedy Heuristic

- **Premise:** Flights with higher frequency (i.e., taking place on more days during a season) are “harder” to allocate
  - Allocation of flights by decreasing order of frequency



1. **Arrange flights into groups of decreasing frequency**
2. **Allocate flights to each group sequentially**

   Solve model for Group 1

   Solve model for Group 2

   Etc.
3. **Other adjustments to maintain global feasibility**

   Update capacity values

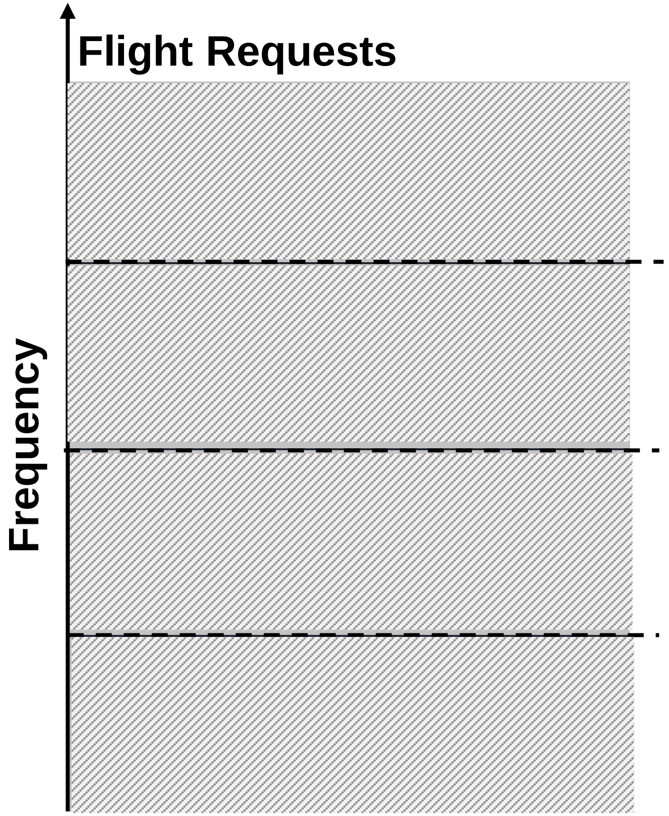
   Ensure feasibility of connections

   Ensure priorities among slot classes

   Etc.

Constructive Greedy Heuristic

- **Premise:** Flights with higher frequency (i.e., taking place on more days during a season) are “harder” to allocate
  - Allocation of flights by decreasing order of frequency



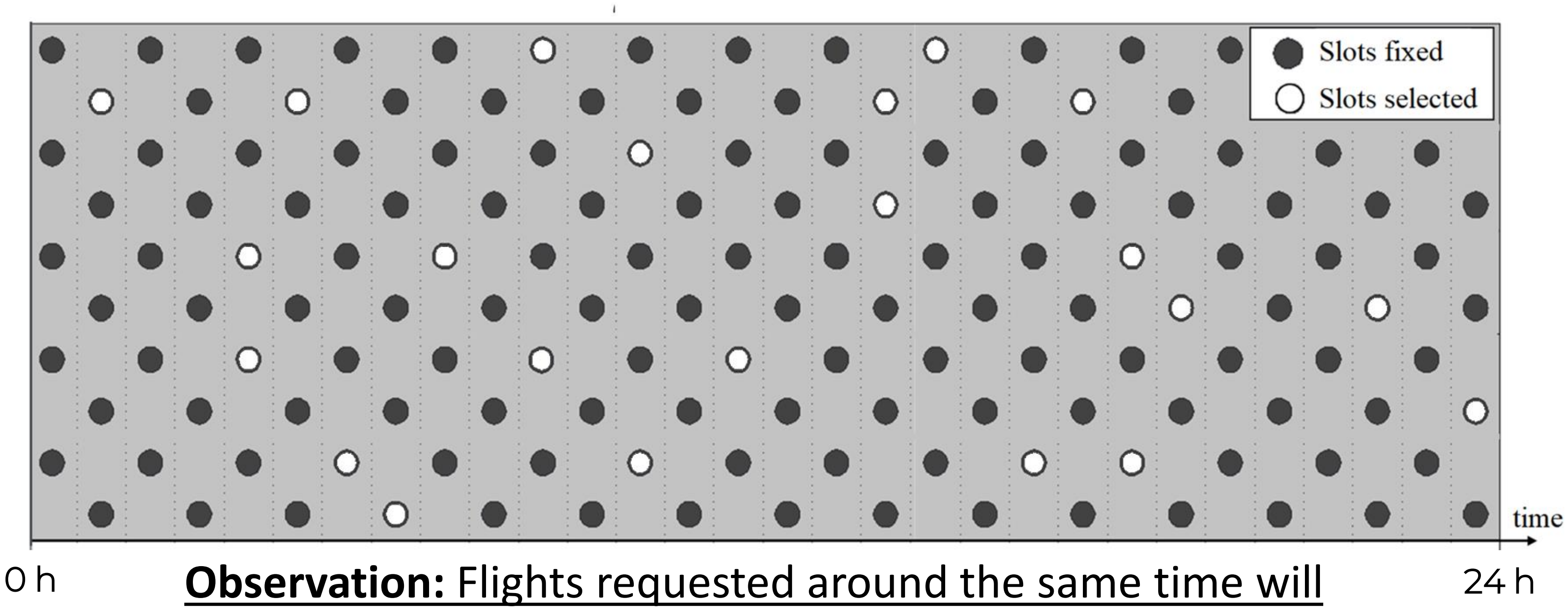
1. **Arrange flights into groups of decreasing frequency**
2. **Allocate flights to each group sequentially**
   Solve model for Group 1
   Solve model for Group 2
   Etc.
3. **Other adjustments to maintain global feasibility**
   Update capacity values
   Ensure feasibility of connections
   Ensure priorities among slot classes
   Etc.

Constructive Greedy Heuristic Results

	Lisbon 2015, with terminal and apron		
Slot allocation strategy	Displacement (min)	Gap (%)	CPU time
SimRand	497,424	33.0	10 hr
SimRandFreqs	399,310	6.8	10 hr
10 groups	394,040	5.4	41 min
8 groups	390,450	4.4	37 min
7 groups	392,290	4.9	30 min
6 groups	390,335	4.4	31 min
5 groups	393,335	5.2	25 min
4 groups	387,125	3.5	22 min
3 groups	381,365	2.0	25 min
2 groups	378,715	1.3	6 hr 15 min
Direct CPLEX	374,005[a]	0.0	>7 days

VLNS Algorithm

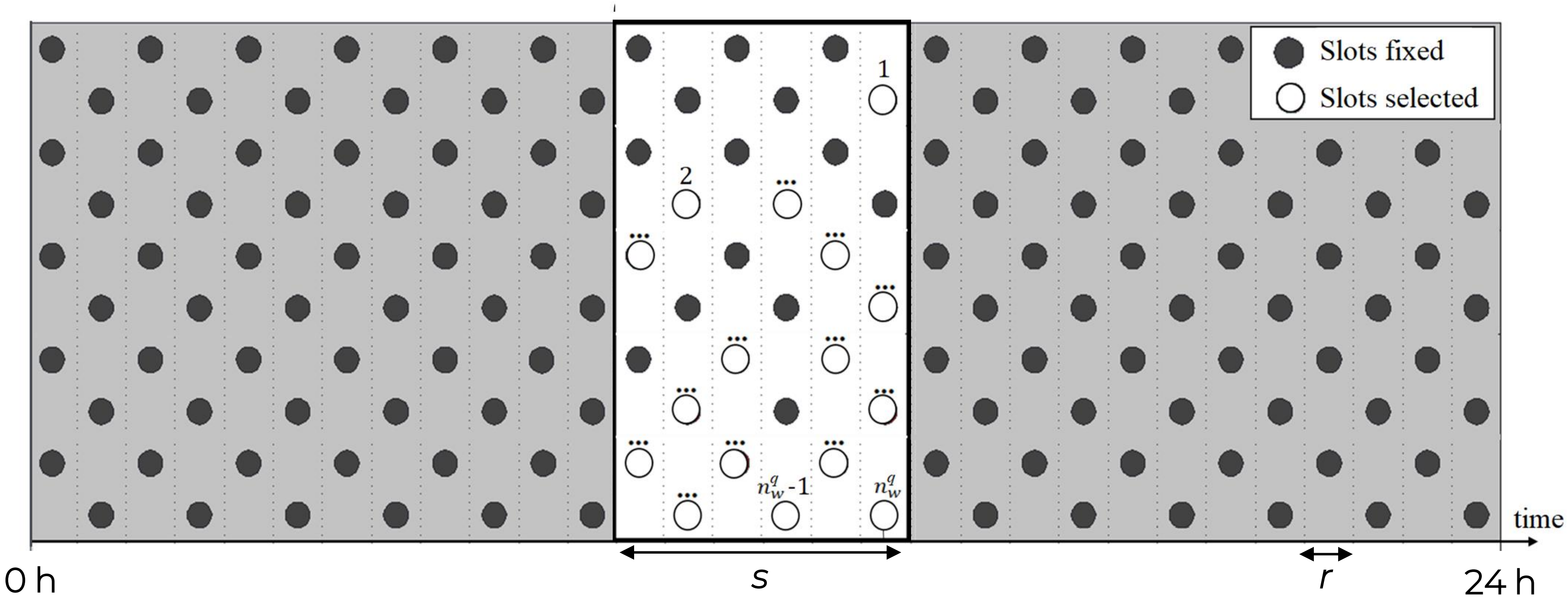
- **Decomposition** of the problem to allocate a subset of all slot requests at each iteration
- Algorithm using large-scale neighborhood search principles, based on “destroy and repair” approach



Observation: Flights requested around the same time will compete with each other for the same slots

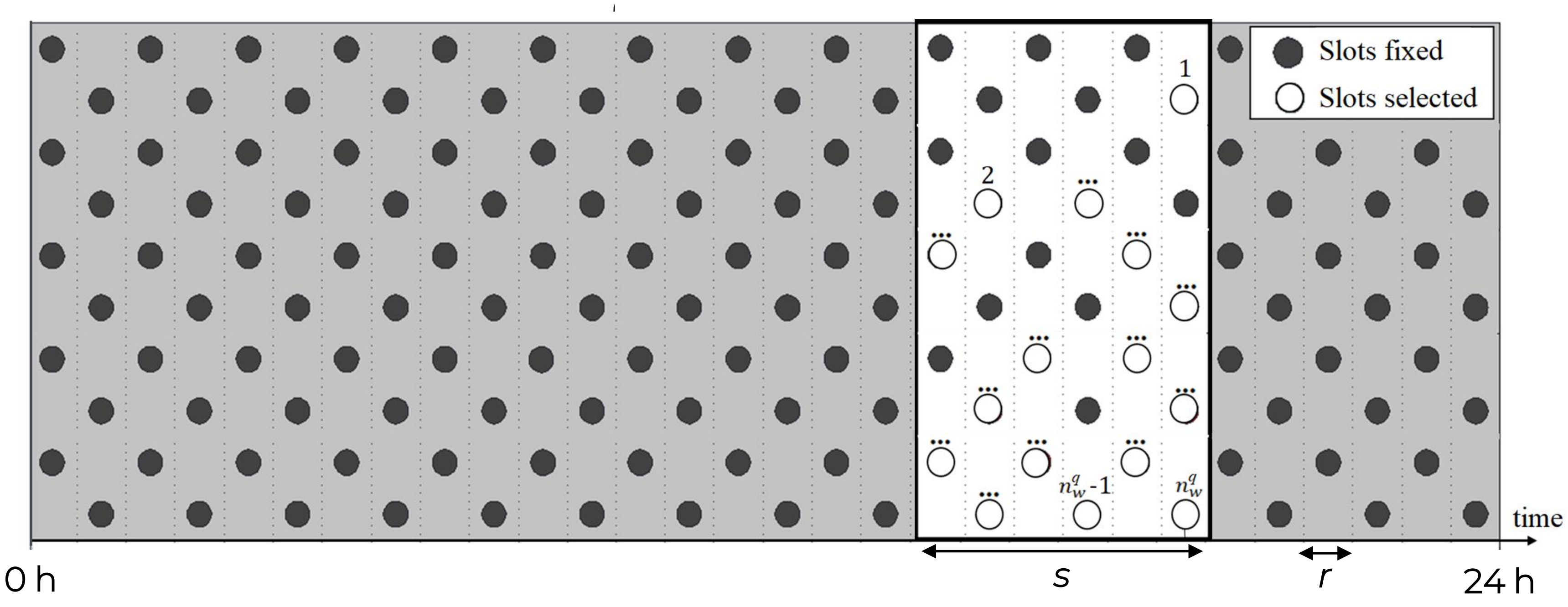
VLNS Algorithm

- **Decomposition over time**:
  - Coupling constraints apply across consecutive time periods of the day (e.g., airport capacity, aircraft connections, schedule regularity)
  - Creation of time-based neighbourhood to capture interdependencies



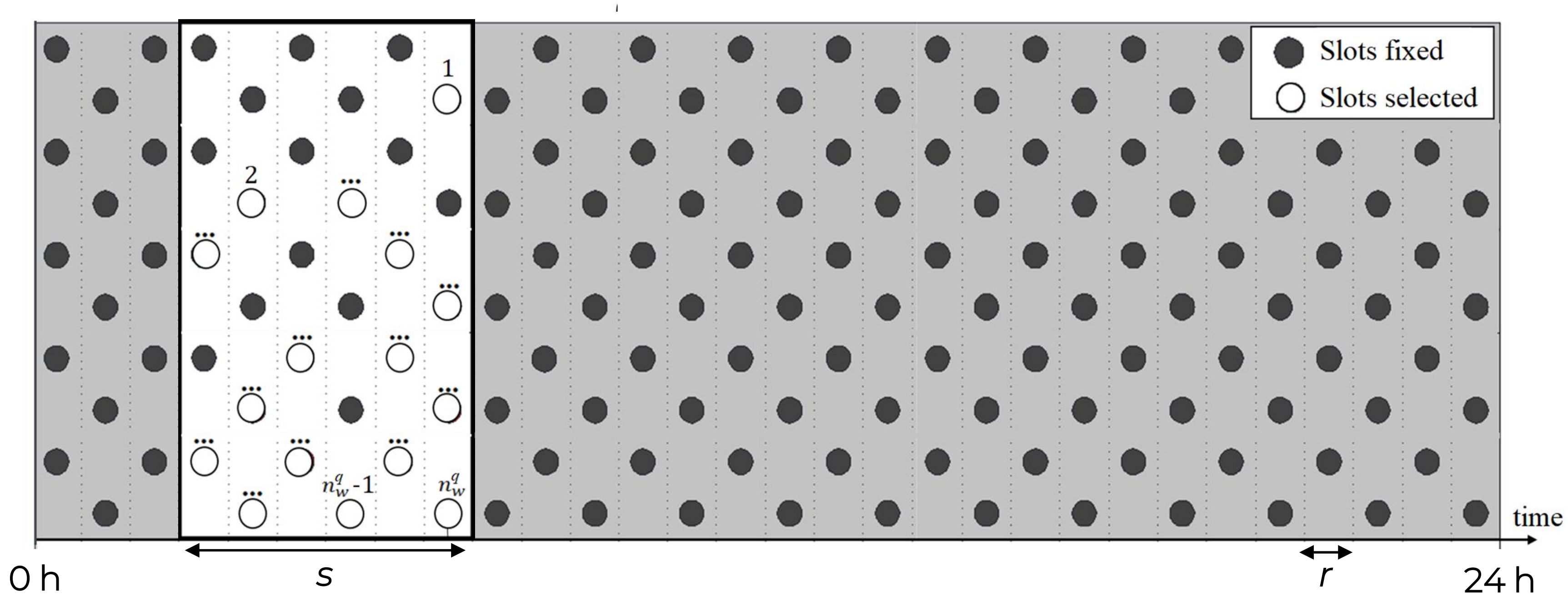
VLNS Algorithm

- **Decomposition over time**:
  - Coupling constraints apply across consecutive time periods of the day (e.g., airport capacity, aircraft connections, schedule regularity)
  - Creation of time-based neighbourhood to capture interdependencies



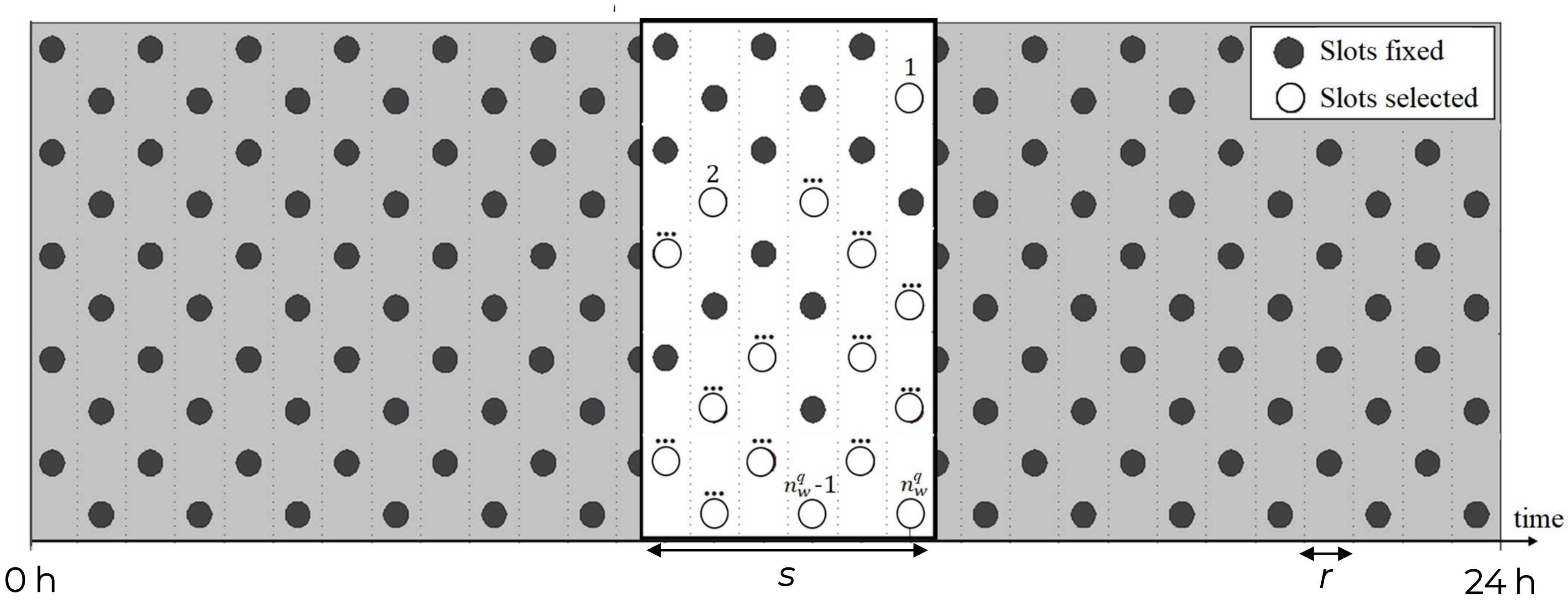
VLNS Algorithm

- **Decomposition over time**:
  - Coupling constraints apply across consecutive time periods of the day (e.g., airport capacity, aircraft connections, schedule regularity)
  - Creation of time-based neighbourhood to capture interdependencies



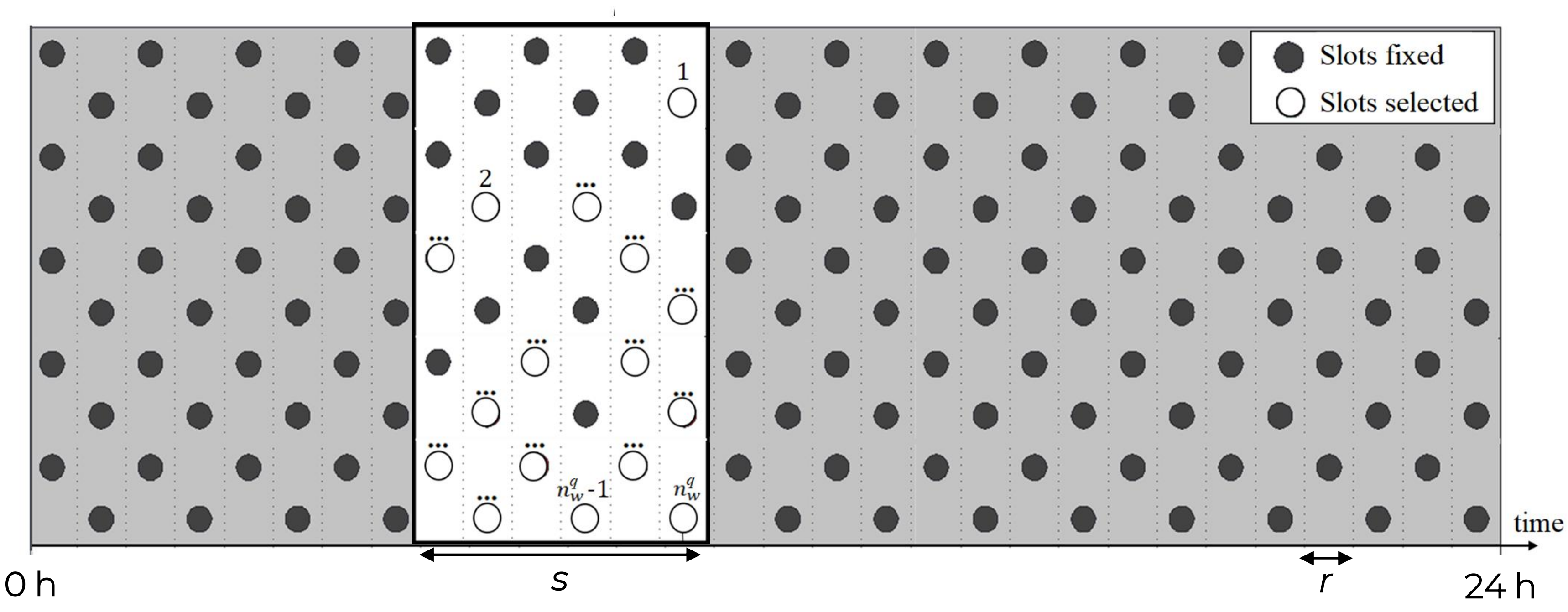
VLNS Algorithm

- **Decomposition over time**:
  - Coupling constraints apply across consecutive time periods of the day (e.g., airport capacity, aircraft connections, schedule regularity)
  - Creation of time-based neighbourhood to capture interdependencies



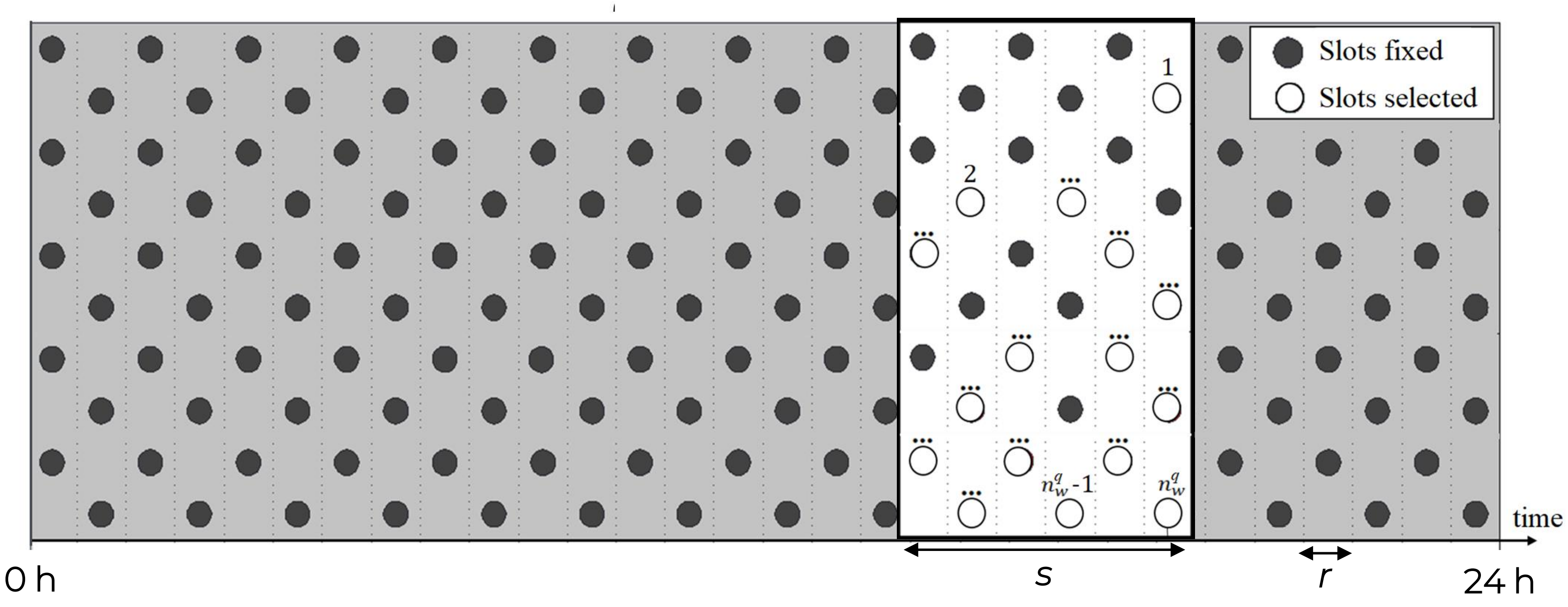
VLNS Algorithm

- **Decomposition over time**:
  - Coupling constraints apply across consecutive time periods of the day (e.g., airport capacity, aircraft connections, schedule regularity)
  - Creation of time-based neighbourhood to capture interdependencies



VLNS Algorithm

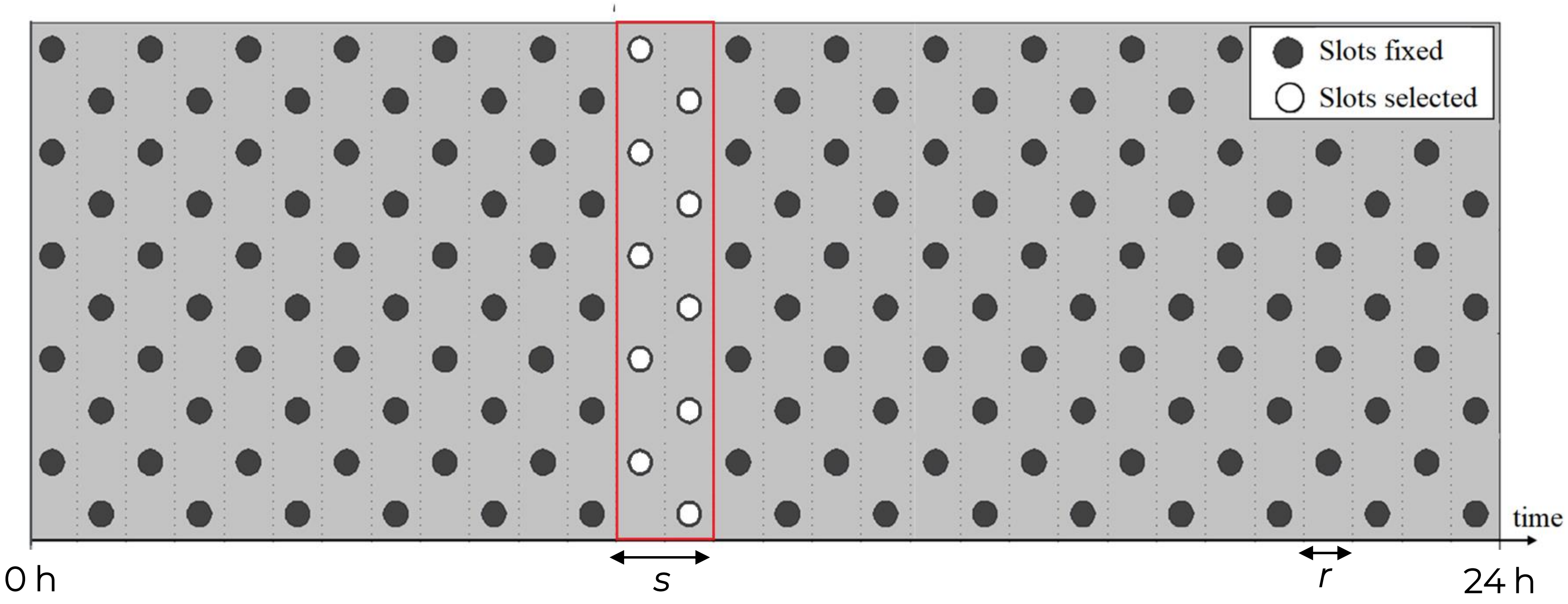
- **Decomposition over time**:
  - Coupling constraints apply across consecutive time periods of the day (e.g., airport capacity, aircraft connections, schedule regularity)
  - Creation of time-based neighbourhood to capture interdependencies



Designing the VLNS Algorithm

Size of the neighbourhood

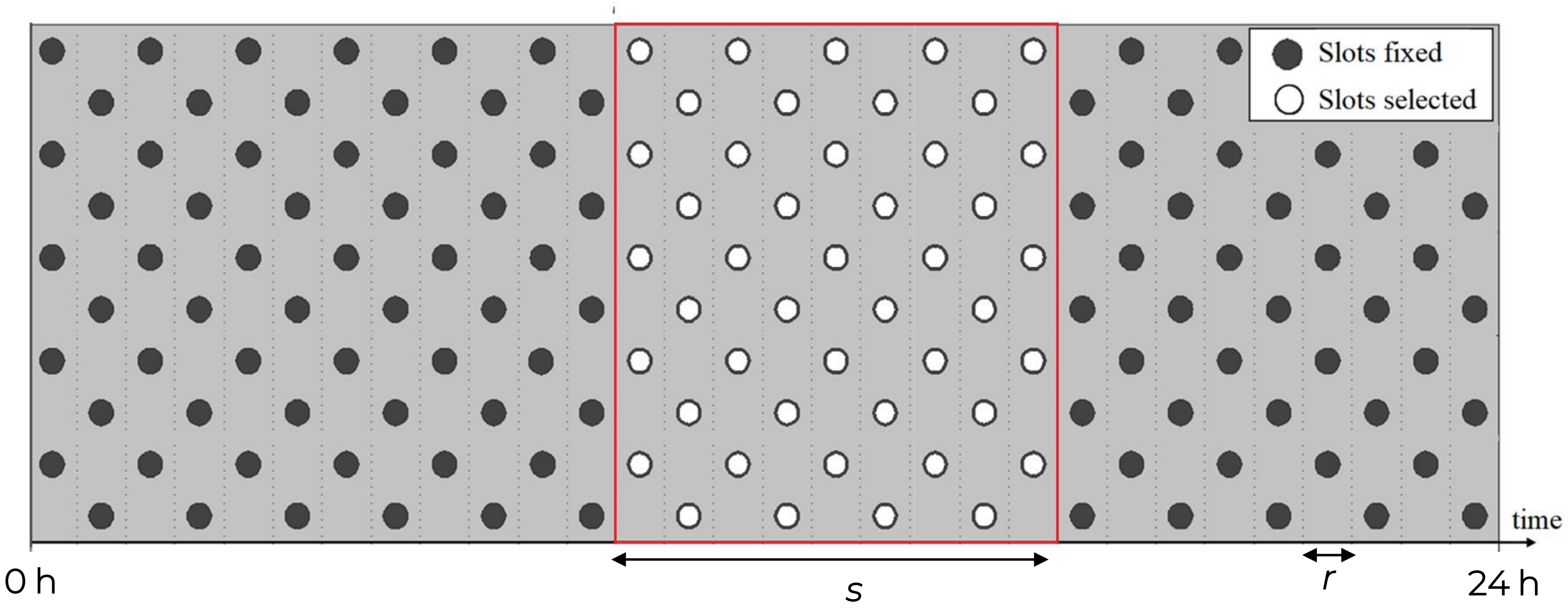
- **Size of the Time Window**
  - Small time windows – small neighbourhoods – local optima
  - Large time windows – larger neighbourhoods – longer runtime



Designing the VLNS Algorithm

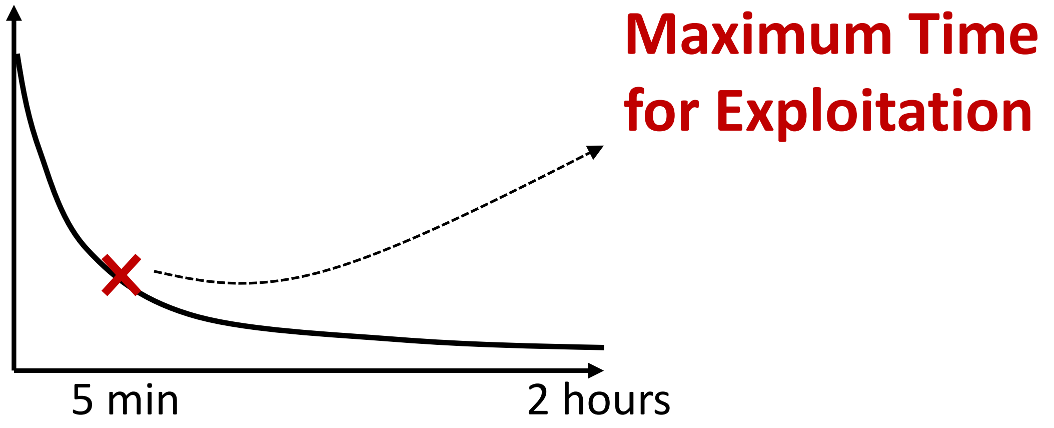
Size of the neighbourhood

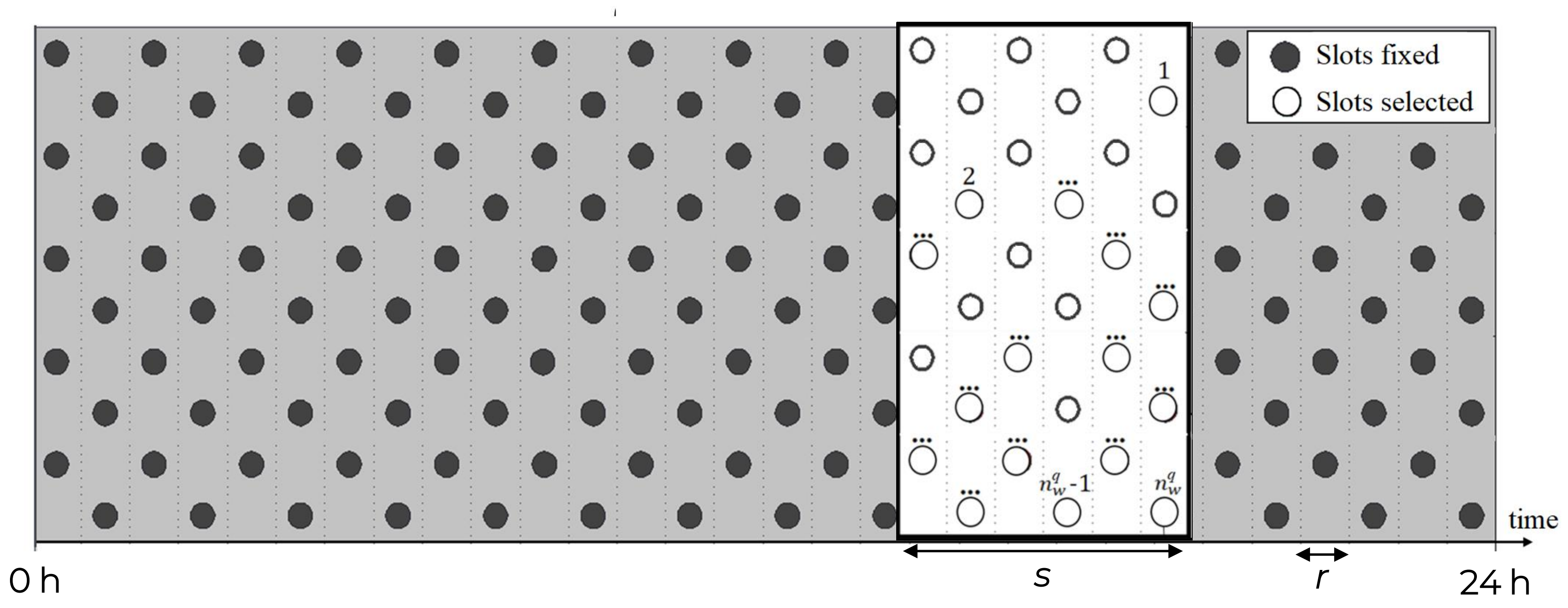
- **Size of the Time Window**
  - Small time windows – small neighbourhoods – local optima
  - Large time windows – larger neighbourhoods – longer runtime



Designing the VLNS Algorithm

- **Time for Optimization**
  - t_o: maximum optimization runtime at each iteration (the most congested time windows take several hours to solve – we aim to avoid getting stuck in a given neighbourhood)

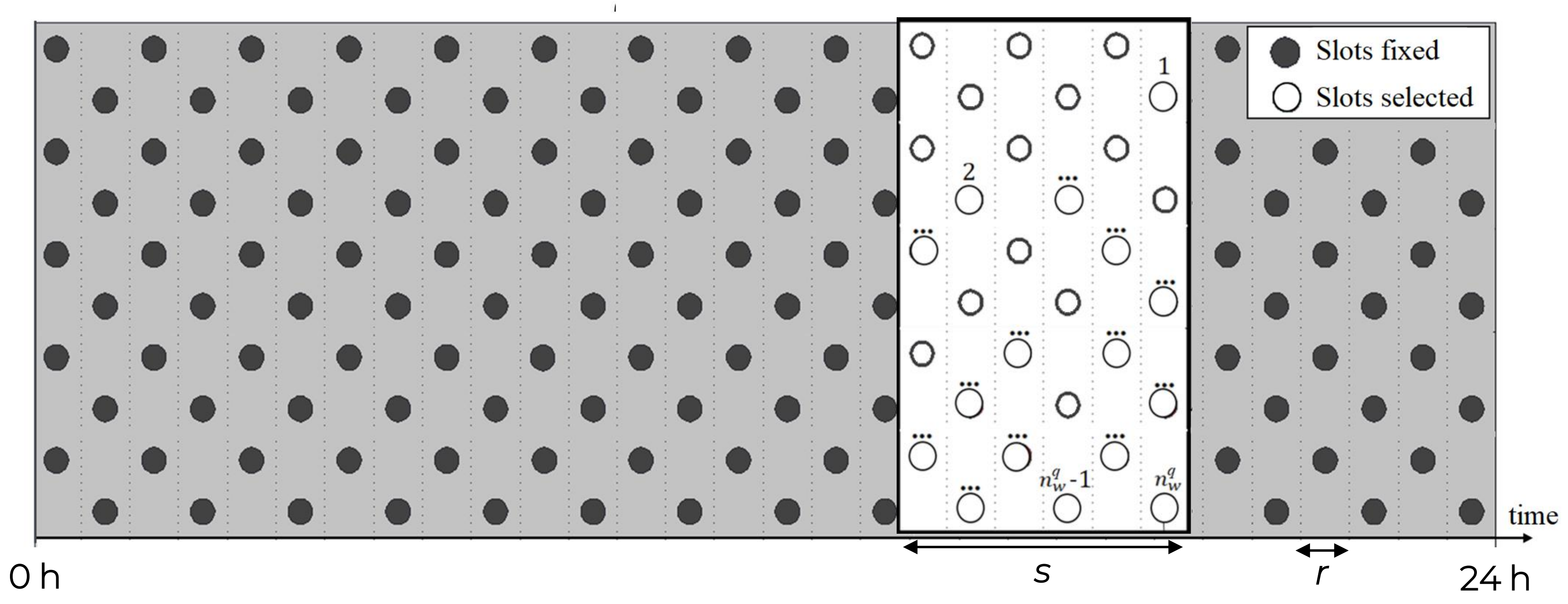




Designing the VLNS Algorithm

- Time window characteristics
  - Probabilistic approach to **slot selection within time window**
  - Initially, all slots are selected in each time window
  - If optimization does not terminate within t_o, reduction of the number of slots selected in next iteration by a factor ρ
  - Priority is given to slots that were requested fewer times

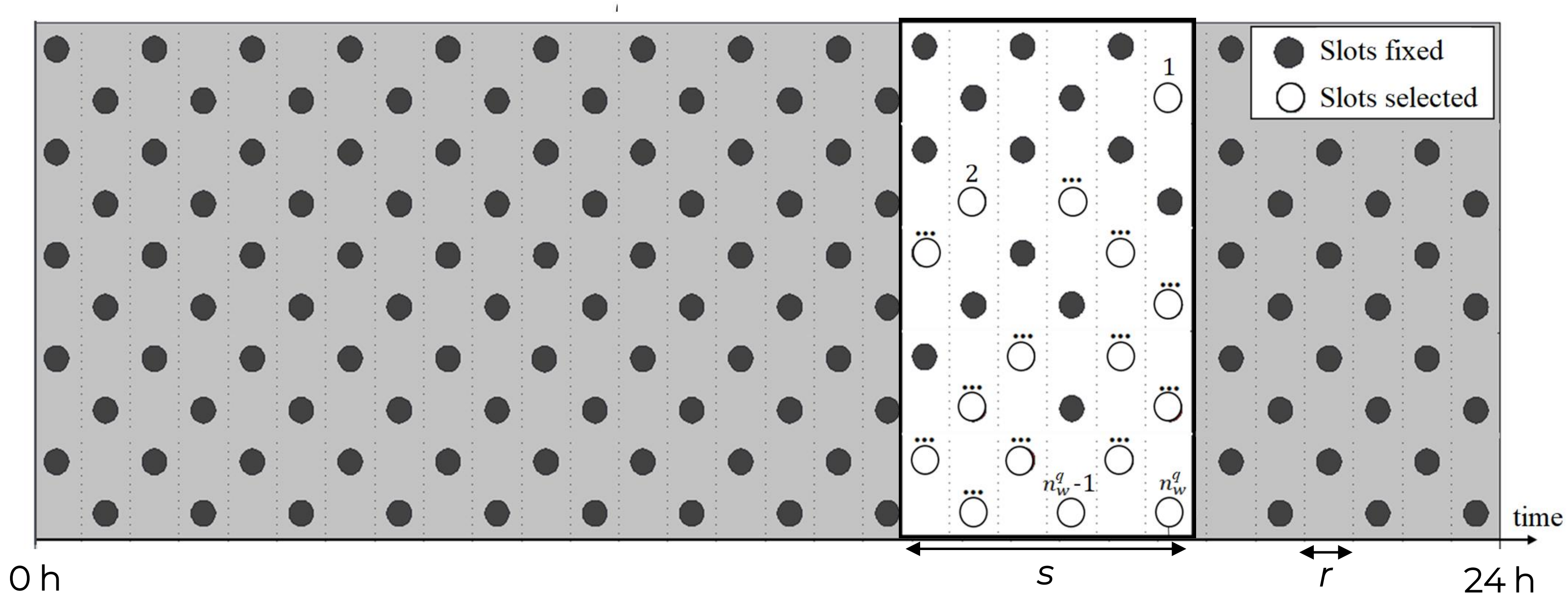
Long Term Memory
Continuous Diversification



Designing the VLNS Algorithm

- Probabilistic approach to **time window selection**
  - If improvements found at an iteration, selection probabilities unchanged
  - If no improvement found in time window w, selection probability of time window w reduced – governed by a parameter δ

Adaptive Neighbourhood Selection



Adaptive VLNS

iteration	Window selected	Optimal solution?	Solution improved?	Proportion of slots selected (R_w^q)			Probability of time window selection (PT_w^q)		
				1	2	3	1	2	3
0	-	-	-	1	1	1	25.0%	50.0%	25.0%
1	2	No	Yes	1	0.8	1	25.0%	50.0%	25.0%
2	2	No	No	1	0.64	1	27.8%	44.4%	27.8%
3	1	Yes	No	1	0.64	1	0.3%	61.4%	38.4%
4	3	No	Yes	1	0.64	0.8	0.3%	61.4%	38.4%
5	2	Yes	No	1	0.64	0.8	0.3%	56.0%	43.7%
6	3	Yes	Yes	1	0.64	0.8	0.3%	56.0%	43.7%
7	2	Yes	No	1	0.64	0.8	0.3%	50.4%	49.2%
8	2	Yes	No	1	0.64	0.8	0.4%	44.9%	54.8%
9	3	Yes	Yes	1	0.64	0.8	0.4%	44.9%	54.8%
10	2	Yes	Yes	1	0.64	0.8	0.4%	44.9%	54.8%

Initial contribution to the objective value

Adaptive VLNS

q	w	Optimal solution?	Solution improved?	Proportion of slots selected (R_w^q)			Probability of time window selection (PT_w^q)		
				1	2	3	1	2	3
0	-	-	-	1	1	1	25.0%	50.0%	25.0%
1	2	No	Yes	1	0.8	1	25.0%	50.0%	25.0%
2	2	No	No	1	0.64	1	27.8%	44.4%	27.8%
3	1	Yes	No	1	0.64	1	0.3%	61.4%	38.4%
4	3	No	Yes	1	0.64	0.8	0.3%	61.4%	38.4%
5	2	Yes	No	1	0.64	0.8	0.3%	56.0%	43.7%
6	3	Yes	Yes	1	0.64	0.8	0.3%	56.0%	43.7%
7	2	Yes	No	1	0.64	0.8	0.3%	50.4%	49.2%
8	2	Yes	No	1	0.64	0.8	0.4%	44.9%	54.8%
9	3	Yes	Yes	1	0.64	0.8	0.4%	44.9%	54.8%
10	2	Yes	Yes	1	0.64	0.8	0.4%	44.9%	54.8%

$$n_w^{(q+1)} = \rho \cdot n_w^{(q)}$$

$$R_w^{(q)} = \frac{n_w^{(q)}}{|S_W|}.$$

Adaptive VLNS

q	w	Optimal solution?	Solution improved?	Proportion of slots selected (R_w^q)			Probability of time window selection (PT_w^q)		
				1	2	3	1	2	3
0	-	-	-	1	1	1	25.0%	50.0%	25.0%
1	2	No	Yes	1	0.8	1	25.0%	50.0%	25.0%
2	2	No	No	1	0.64	1	27.8%	44.4%	27.8%
3	1	Yes	No	1	0.64	1	0.3%	61.4%	38.4%
4	3	No	Yes	1	0.64	0.8	0.3%	61.4%	38.4%
5	2	Yes	No	1	0.64	0.8	0.3%	56.0%	43.7%
6	3	Yes	Yes	1	0.64	0.8	0.3%	56.0%	43.7%
7	2	Yes	No	1	0.64	0.8	0.3%	50.4%	49.2%
8	2	Yes	No	1	0.64	0.8	0.4%	44.9%	54.8%
9	3	Yes	Yes	1	0.64	0.8	0.4%	44.9%	54.8%
10	2	Yes	Yes	1	0.64	0.8	0.4%	44.9%	54.8%

λ is a calibration parameter, and s_i^q indicates the number of times slot request i was selected up to iteration q. The larger the λ, the more the algorithm favors slot requests that were selected fewer times before. If $\lambda = 0$, slot requests are selected completely at random; if $\lambda = \infty$ we select slot requests exclusively among those that were explored the least numbers of times in previous iterations.

$$PS_i^{(q)} = \frac{e^{-\lambda s_i^{(q)}}}{\sum_{j \in S} e^{-\lambda s_j^{(q)}}}, \forall i \in S_W$$

Long Term Memory

Continuous Diversification

Adaptive VLNS

q	w	Optimal solution?	Solution improved?	Proportion of slots selected ($\boldsymbol{R_w^q}$)			Probability of time window selection ($\boldsymbol{PT_w^q}$)		
				1	2	3	1	2	3
0	-	-	-	1	1	1	25.0%	50.0%	25.0%
1	2	No	Yes	1	0.8	1	25.0%	50.0%	25.0%
2	2	No	No	1	0.64	1	27.8%	44.4%	27.8%
3	1	Yes	No	1	0.64	1	0.3%	61.4%	38.4%
4	3	No	Yes	1	0.64	0.8	0.3%	61.4%	38.4%
5	2	Yes	No	1	0.64	0.8	0.3%	56.0%	43.7%
6	3	Yes	Yes	1	0.64	0.8	0.3%	56.0%	43.7%
7	2	Yes	No	1	0.64	0.8	0.3%	50.4%	49.2%
8	2	Yes	No	1	0.64	0.8	0.4%	44.9%	54.8%
9	3	Yes	Yes	1	0.64	0.8	0.4%	44.9%	54.8%
10	2	Yes	Yes	1	0.64	0.8	0.4%	44.9%	54.8%

If the solution improved, then probabilities PT_w^q are not updated. Otherwise, they are updated. β and γ_q are calibration parameters, and R_w^q denotes the ratio of the number of slots selected at iteration q to the total number of slots requested in time window w

$$PT_w^{(q)'} = PT_w^{(q-1)} \times e^{-\beta\left(R_w^{(q)}\right)^{\gamma_q}};$$

$$PT_{w'}^{(q)} = \frac{PT_w^{(q)'}}{\sum_{u \epsilon W} PT_u^{(q)'}}, \forall w' \epsilon \boldsymbol{W};$$

$$PT_2^{(3)'} = 0.5 \times e^{-5(0.64)^{6.97}} = 0.400,$$

$$PT_1^{(3)} = PT_3^{(3)} = \frac{0.25}{0.25+0.4+0.25} = 0.278,$$

$$PT_2^{(3)} = \frac{0.4}{0.25+0.4+0.25} = 0.444.$$

Adaptive Neighbourhood Selection

Adaptive VLNS

q	w	Optimal solution?	Solution improved?	Proportion of slots selected (R_w^q)			Probability of time window selection (PT_w^q)		
				1	2	3	1	2	3
0	-	-	-	1	1	1	25.0%	50.0%	25.0%
1	2	No	Yes	1	0.8	1	25.0%	50.0%	25.0%
2	2	No	No	1	0.64	1	27.8%	44.4%	27.8%
3	1	Yes	No	1	0.64	1	0.3%	61.4%	38.4%
4	3	No	Yes	1	0.64	0.8	0.3%	61.4%	38.4%
5	2	Yes	No	1	0.64	0.8	0.3%	56.0%	43.7%
6	3	Yes	Yes	1	0.64	0.8	0.3%	56.0%	43.7%
7	2	Yes	No	1	0.64	0.8	0.3%	50.4%	49.2%
8	2	Yes	No	1	0.64	0.8	0.4%	44.9%	54.8%
9	3	Yes	Yes	1	0.64	0.8	0.4%	44.9%	54.8%
10	2	Yes	Yes	1	0.64	0.8	0.4%	44.9%	54.8%

Adaptive VLNS

q	w	Optimal solution?	Solution improved?	Proportion of slots selected (R_w^q)			Probability of time window selection (PT_w^q)		
				1	2	3	1	2	3
0	-	-	-	1	1	1	25.0%	50.0%	25.0%
1	2	No	Yes	1	0.8	1	25.0%	50.0%	25.0%
2	2	No	No	1	0.64	1	27.8%	44.4%	27.8%
3	1	Yes	No	1	0.64	1	0.3%	61.4%	38.4%
4	3	No	Yes	1	0.64	0.8	0.3%	61.4%	38.4%
5	2	Yes	No	1	0.64	0.8	0.3%	56.0%	43.7%
6	3	Yes	Yes	1	0.64	0.8	0.3%	56.0%	43.7%
7	2	Yes	No	1	0.64	0.8	0.3%	50.4%	49.2%
8	2	Yes	No	1	0.64	0.8	0.4%	44.9%	54.8%
9	3	Yes	Yes	1	0.64	0.8	0.4%	44.9%	54.8%
10	2	Yes	Yes	1	0.64	0.8	0.4%	44.9%	54.8%

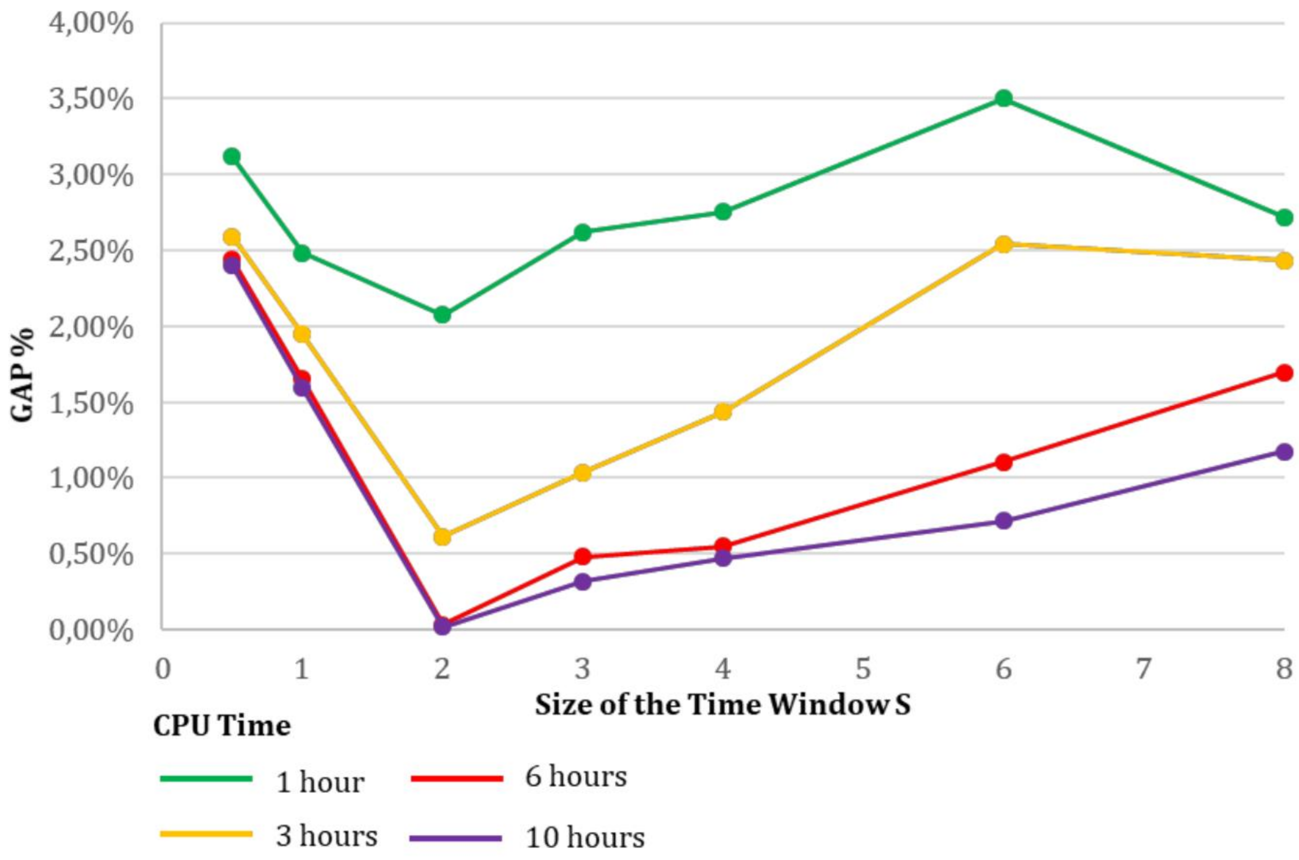
Adaptive VLNS - Results

CPU Time	Exact Methods		Heuristic Approach	
	Total Disp. (min)	Gap %	Total Disp. (min)	Gap %
30 min	N/A		432,500	3.51%
1 hour	N/A		429,995	2.91%
6 hours	N/A		418,165	0.08%
1 day	N/A		417,840	0.00%
2 days	458,620	9.76%	417,840	0.00%
4 days	430,415	3.01%	417,840	0.00%
7 days	419,845	0.48%	417,840	0.00%

- **Observation: The heuristic approach generates, in a few hours, near optimal solutions in instances where exact methods do not find the optimal solution after several days**

Size of the Neighbourhood

- Sweet spot in length of time windows
  - Large neighbourhoods take too much time in each iteration
  - Small neighbourhoods require too many iterations before convergence
- Validates our large-scale neighbourhood search approach



Neighborood Selection

- Completely random time window selection yields poor results
- Sweet spot between exploration and exploitation approaches
  - Validation of probabilistic time window selection that orients the search toward more “promising” neighbourhoods

Value of δ	Initial solution	CPU Time								Avg. no. iterations
		1 hour		3 hours		6 hours		10 hours		
	Avg. gap (%)	Avg. gap (%)	Gap range (%)	Avg. gap (%)	Gap range (%)	Avg. gap (%)	Gap range (%)	Avg. gap (%)	Gap range (%)	
0.3	3.51	2.23	1.2-3.5	0.66	0.1-1.4	0.13	0.0-0.5	0.06	0.0-0.2	94
0.5	3.51	2.43	1.3-3.5	0.44	0.1-1.5	0.11	0.0-0.4	0.06	0.0.-0.3	83
0.7	3.51	2.49	1.6-3.5	0.58	0.0-2.0	0.09	0.0-0.4	0.06	0.0-0.3	88
0.8	3.51	2.07	1.2-3.5	0.61	0.1-2.8	0.03	0.0-0.1	0.02	0.0-0.1	83
0.9	3.51	2.38	1.3-3.5	1.00	0.0-2.5	0.12	0.0-0.4	0.05	0.0-0.3	77
1	3.51	2.33	1.3-3.5	0.92	0.0-1.9	0.31	0.0-0.7	0.04	0.0-0.1	84
No PD_w	3.51	2.12	0.4-3.0	0.76	0.2-1.3	0.45	0.1-1.3	0.26	0.0-0.4	116

Exploration vs Exploitation

- Sweet spot in time for optimization
  - if t_o is set too high, the improvement heuristic spends much time searching - poor exploration
  - if t_o is set too low, improvement heuristic stops before significant improvements are obtained, leading to bad convergence solutions - poor exploitation