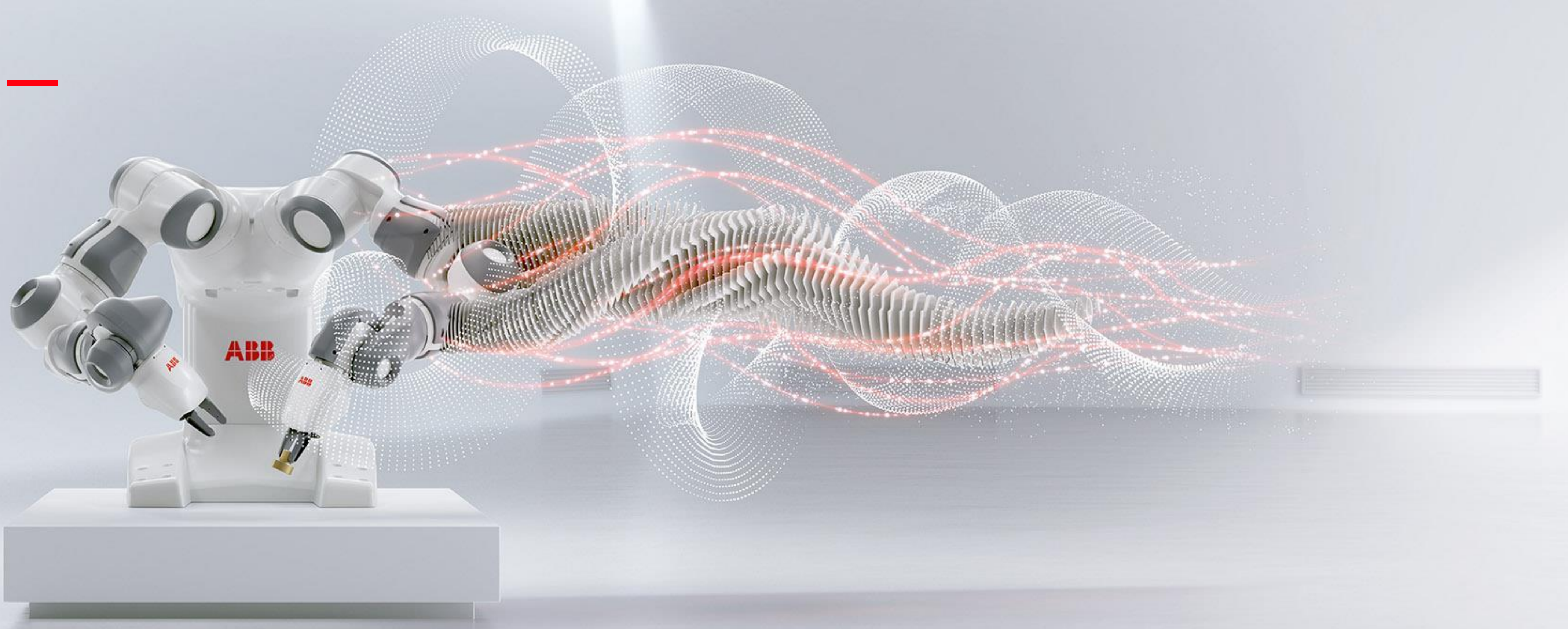


EPFL, 27.03.2018

Optimization challenges in ABB

Alessandro Zanarini, PhD, Principal Scientist

Agenda

Introduction to ABB

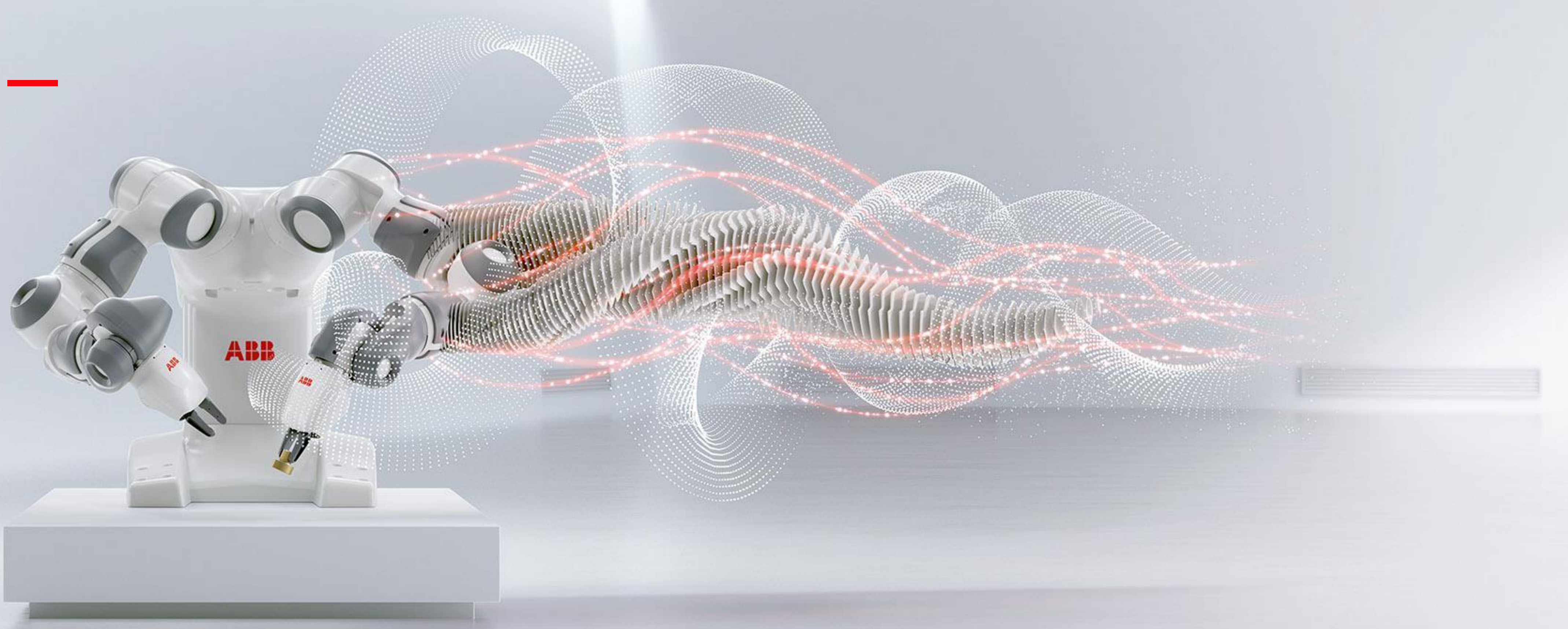
(My very own experience on) Optimization Role and Goals

Optimization challenges in ABB

Case study

Constraint Programming in a Nutshell

Conclusions



Introduction to ABB

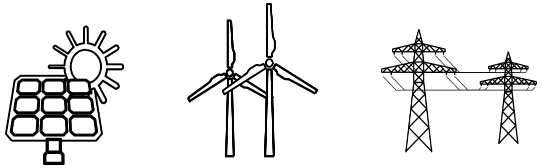
ABB: the pioneering technology leader

What (Offering)	**Pioneering technology**		
	Products 58%	Systems 24%	Services & software 18%
For whom (Customers)	**Utilities**	**Industry**	**Transport & Infrastructure**
	~35% of revenue	~40% of revenue	~25% of revenue
Where (Geographies)	**Globally**		
	Asia, Middle East, Africa 38%	Americas 29%	Europe 33%

~$34 bn revenue	~100 countries	~132,000 employees

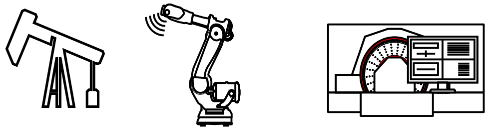
About ABB

Utilities



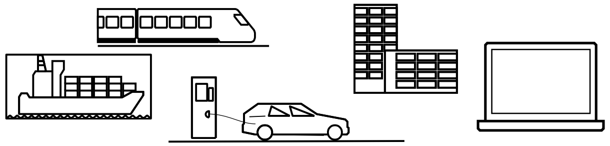
Renewables
Grid automation
Digitalization
Microgrids
Electrification penetration
Energy storage

Industry



Productivity
Energy efficiency
Automation penetration
Internet of Things
Power quality / reliability
Emerging markets

Transportation & Infrastructure



Smart Cities
Data management
Electric transport
Energy efficiency
Power quality / reliability
Decentralized power generation

Shaping the world through innovation

+$1.5 bn

Investment annually

~ 8,500

Scientists and engineers

~ 70

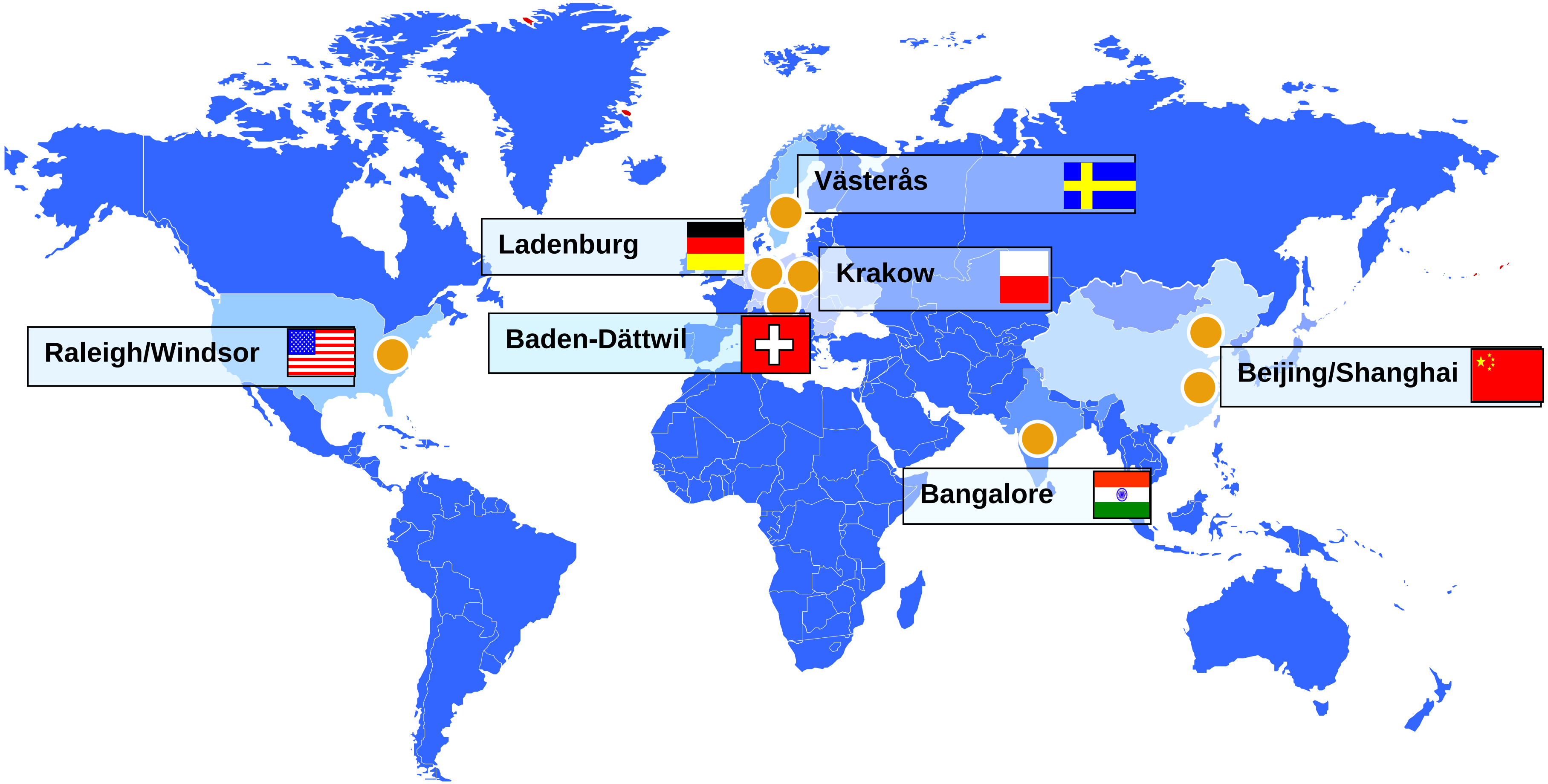
University collaborations

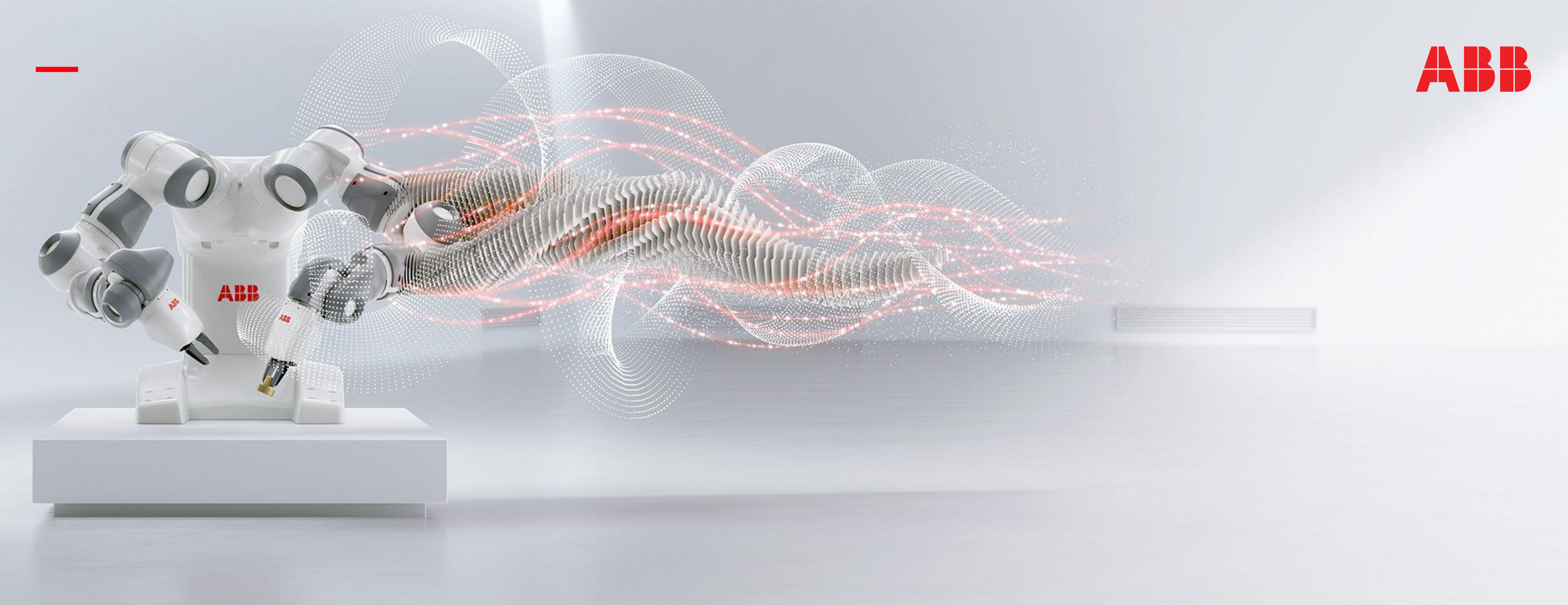
7

Corporate research labs linked by a global research organization

Innovation is ingrained in the DNA of ABB

7 Research centers





(My very own experience on) Optimization Role and Goals

Automated tool vs Optimization

- Shift from “manual” to “automated tool” is seen as the holy grail – underlying problem can be tough
- Optimization seen as cherry on the cake… but the cake is needed first ☺
- Optimization expert needs to educate the customer about “optimization potential/capabilities”
- Customer does not (always) know what he/she wants to optimize
- Optimization can unleash considerable potential savings

- Optimization may threaten jobs. No-optimization may threaten entire companies

Optimization development phases

1. Discovery
   – Understanding the problem, its constraints, its objective function(s)
2. **Designing and implementing an optimization model/algorithm**
   – **All models are wrong but some are useful (cit. George Box)**
   → understand necessary assumptions/approximations
3. Integrating with existing IT system / workflow
   – Fetching and preparing input to optimization model/algorithm
   – Feeding back the (sub) optimal solution
4. Testing
   – Verifying constraint satisfaction, hypothesis, etc…

40%

15%

25%

20%

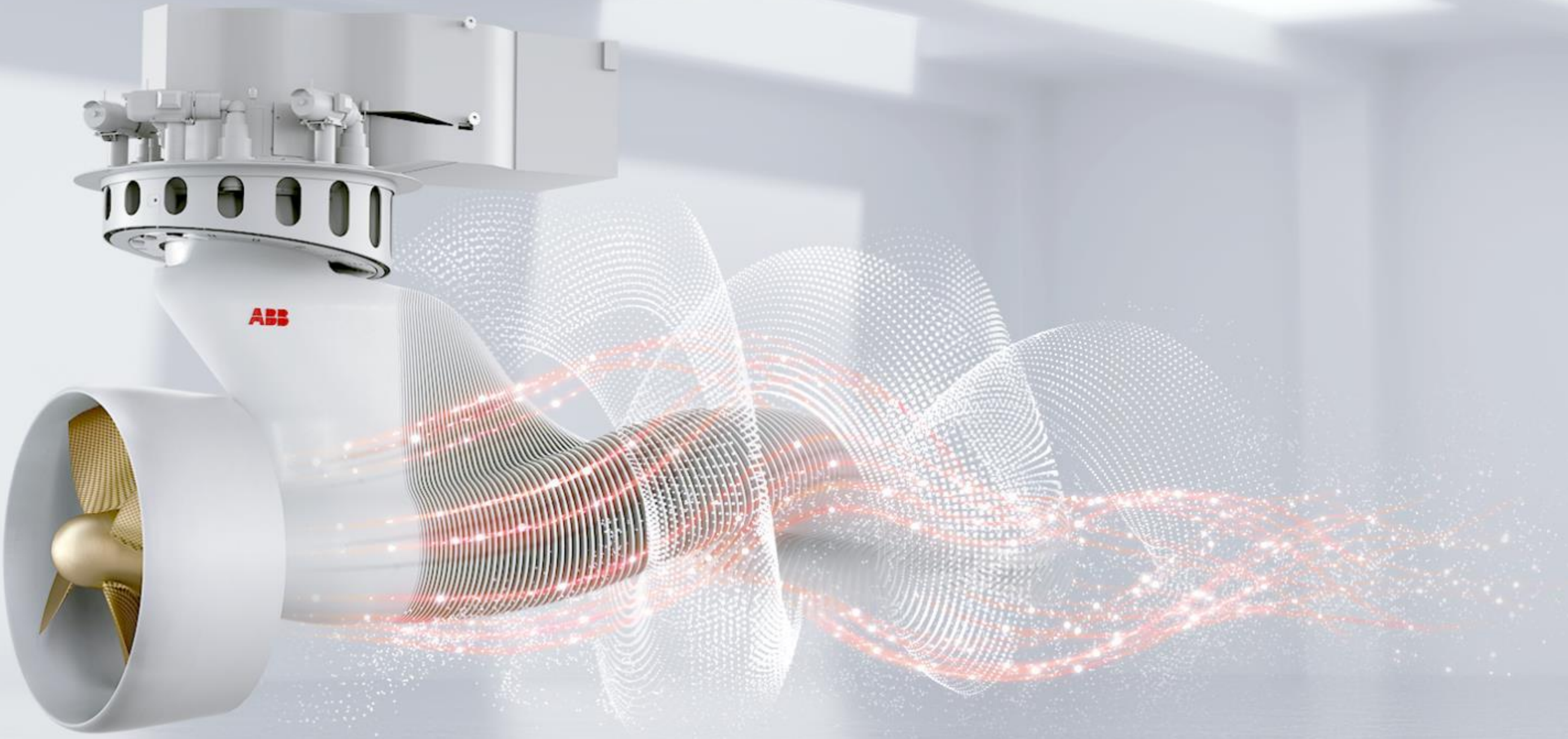
Business case/model needs to be defined!!!

Optimization & Data science technologies

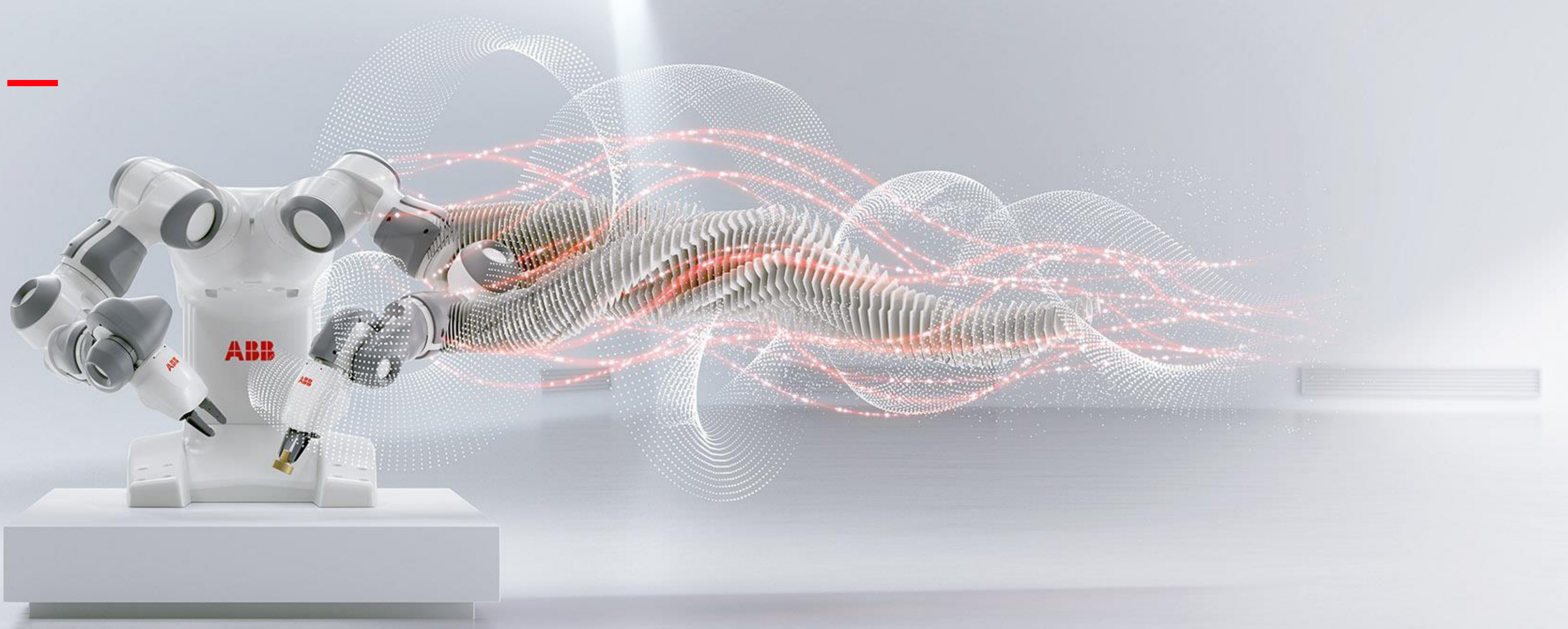
An incomplete list for discrete optimization

- Mathematical Programming
- Greedy algorithm
- Constraint Programming
- Graph algorithms
- Metaheuristics
- Swarm intelligence
- Genetic Algorithms
- Machine Learning

Master the technologies and understand pros and cons

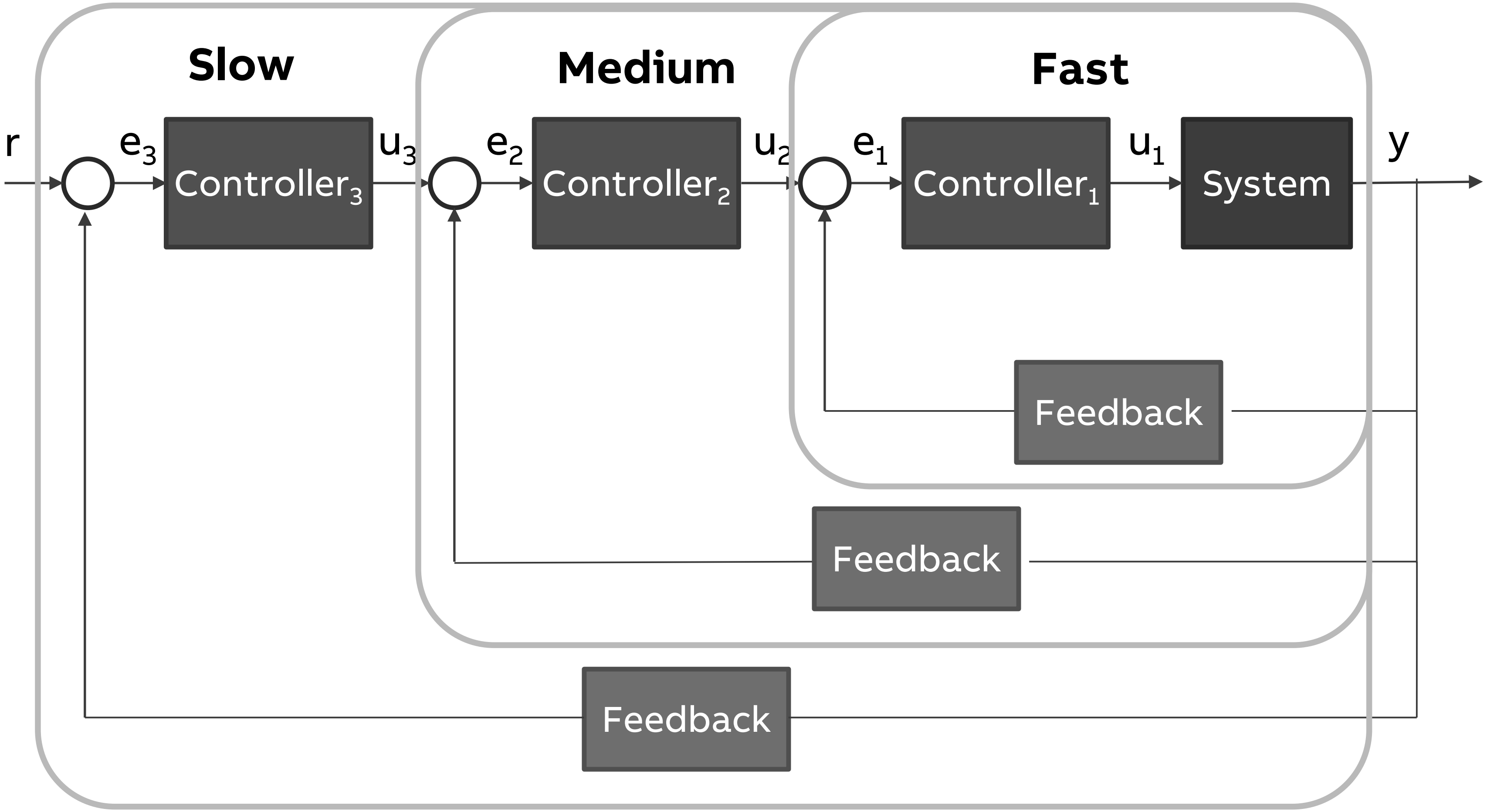


Optimization challenges in ABB



Optimal deployment of control solutions

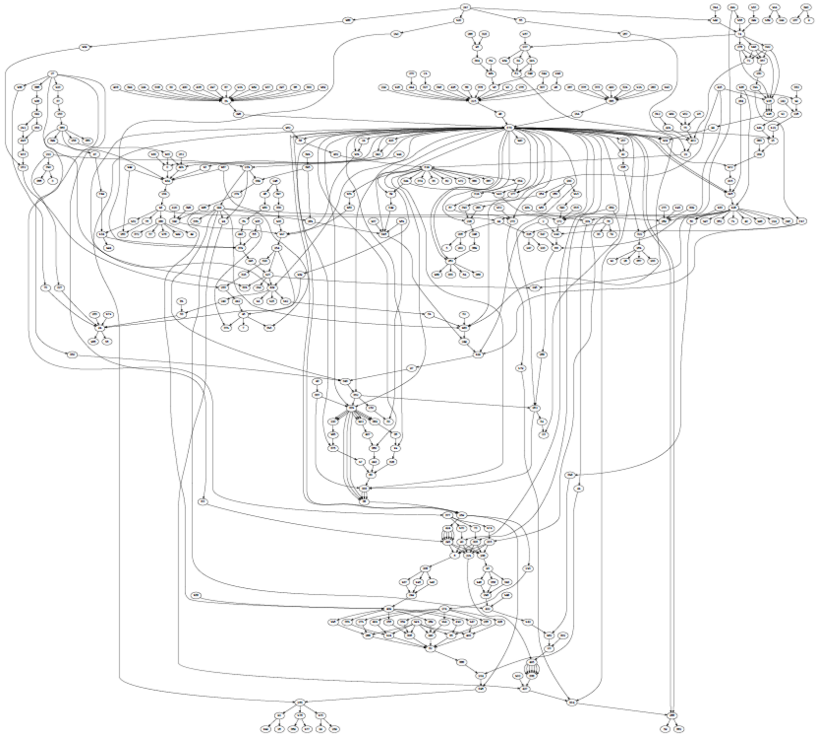
Multirate control systems



Alessio Bonfietti, Alessandro Zanarini, Michele Lombardi, Michela Milano
https://link.springer.com/chapter/10.1007/978-3-319-44953-1_8

Context

Software



Hardware

- Heteregeneous parallel computational resources
- SoC (2 cores + FPGA)

Problem Definition

- *Set of homogeneous resources R*
- Set of *cyclic applications* $A = \{a_0, \dots, a_{n-1}\}$
  - with fixed priority $prio(a_0) > \dots > prio(a_{n-1})$
  - *with different periods* $\lambda_{i+1} = \eta_i \lambda_i \quad (\lambda_{max} = \lambda_{n-1})$
- Apps composed from *activities* $V_i = \{x_j^i\}$
  - with fixed duration $d(x_j^i)$
  - and precedences $x_j^i \prec x_k^i$

Exploit periodicity and modularity to decrease # variables, computation time and memory usage

Objective function

Minimize makespan of a_0 then a_1 then …

$$\min lexico(makespan(a_0), \dots, makespan(a_{n-1}))$$

Alessio Bonfietti, Alessandro Zanarini, Michele Lombardi, Michela Milano
https://link.springer.com/chapter/10.1007/978-3-319-44953-1_8

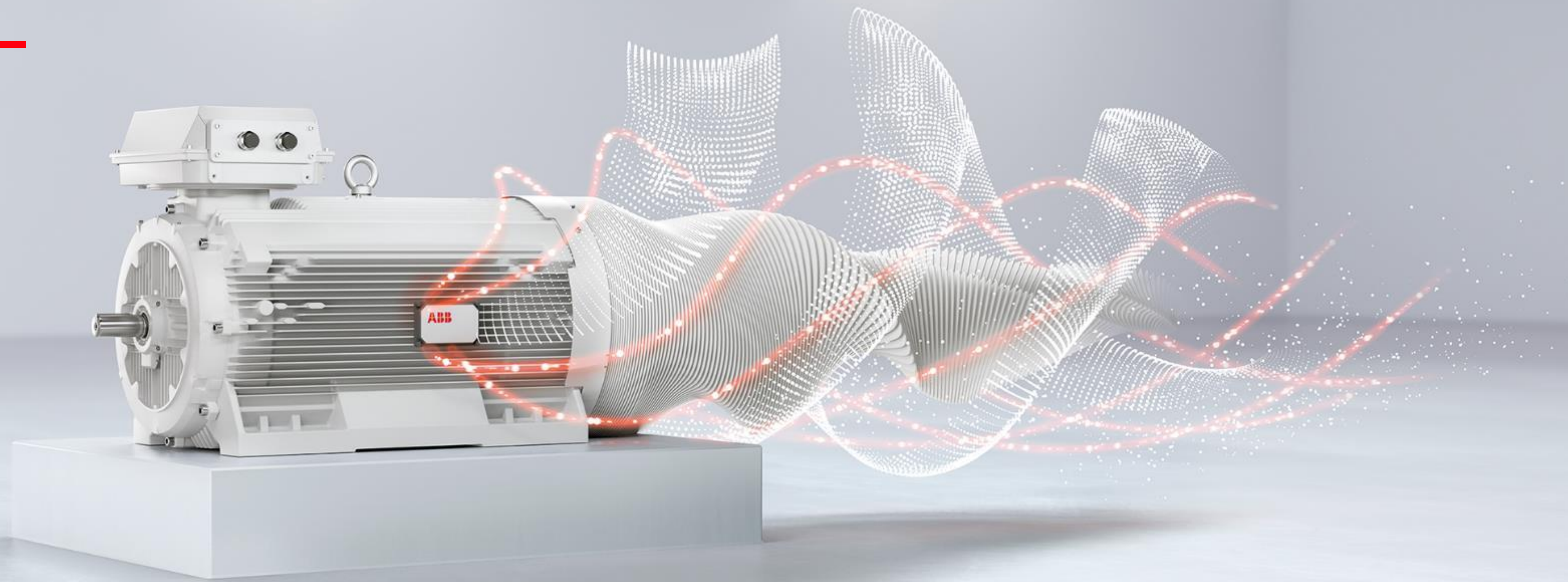
Experimental evaluation

	Avg #act	MRC	T&E	DJ
Real 1 (η_{tot} = 36)	2353	5	521	496
Real 2 (η_{tot} = 2000)	177646	159	1827187	2468504

Solution time (ms)

		MRC	T&E	DJ
Real 1 (η_{tot} = 36)		14.9	27.4	29.25
Real 2 (η_{tot} = 2000)		34.4	1258.3	1253.8

Memory Consumption (MB)

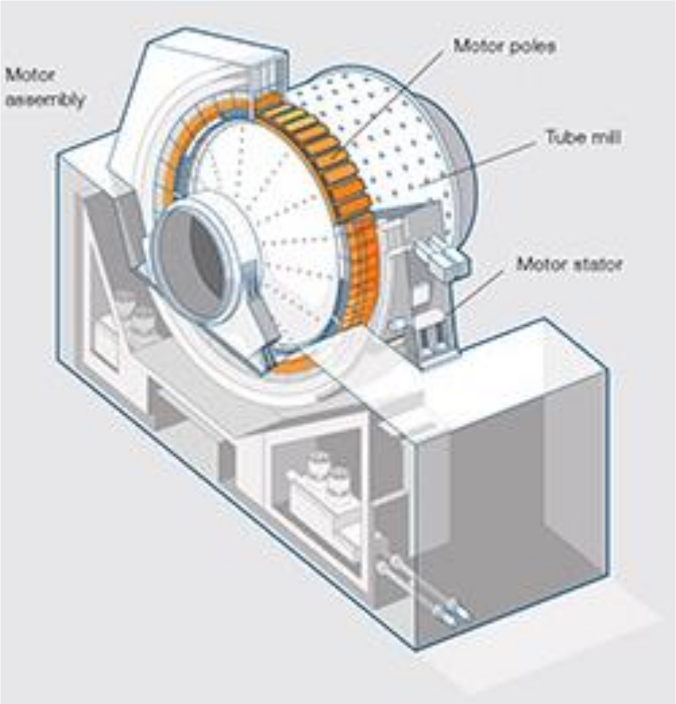


Stator Winding Design Optimization

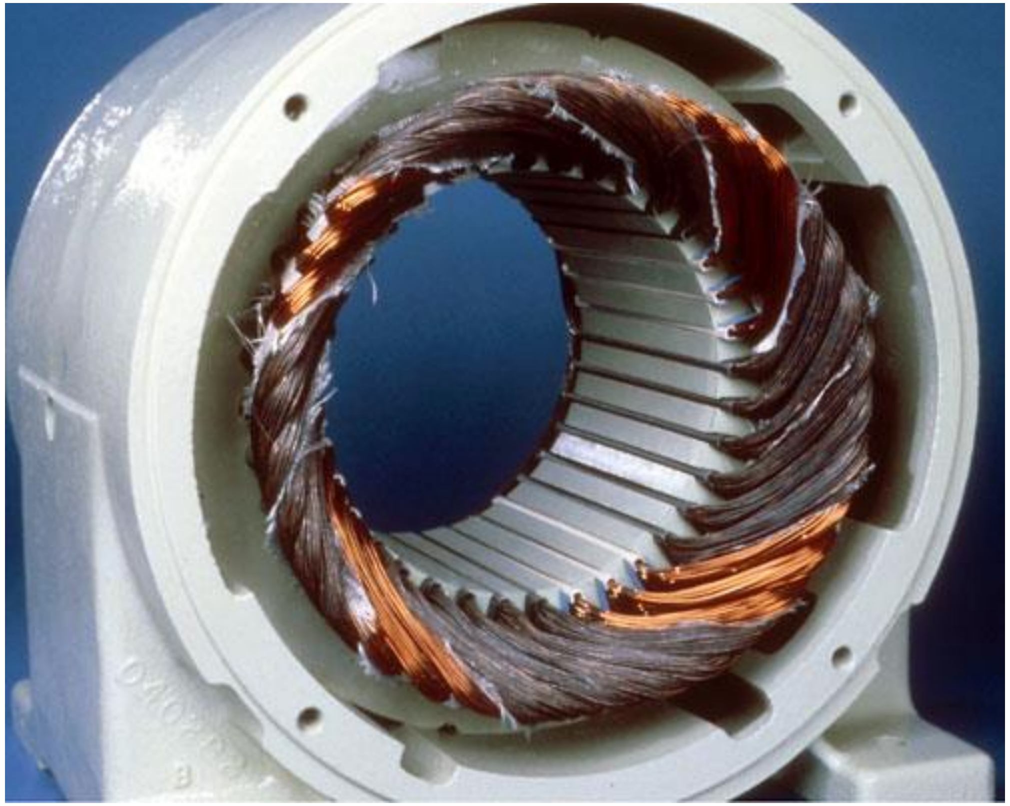
Gearless Mill Drives

Pioneered by ABB in 1969
low rpm, high torque, diameter up to 12m, up to ~30MW

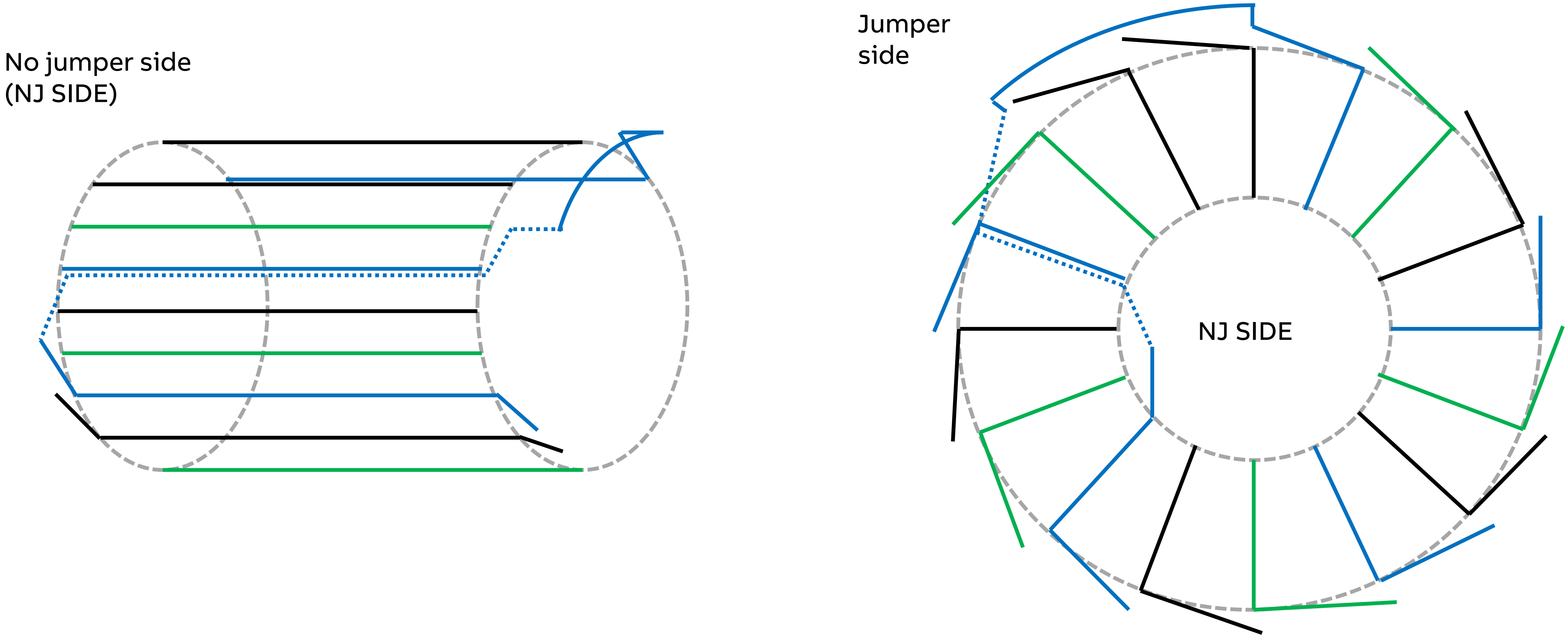
Stator



Stator Winding



Definitions – phases and connectors



Black lines → phase U, Blue lines → phase V, Green lines → phase W

Problem description

Parameters definition

1. Physical dimensions
2. Number of slots
3. Number of poles
4. Coil pitch

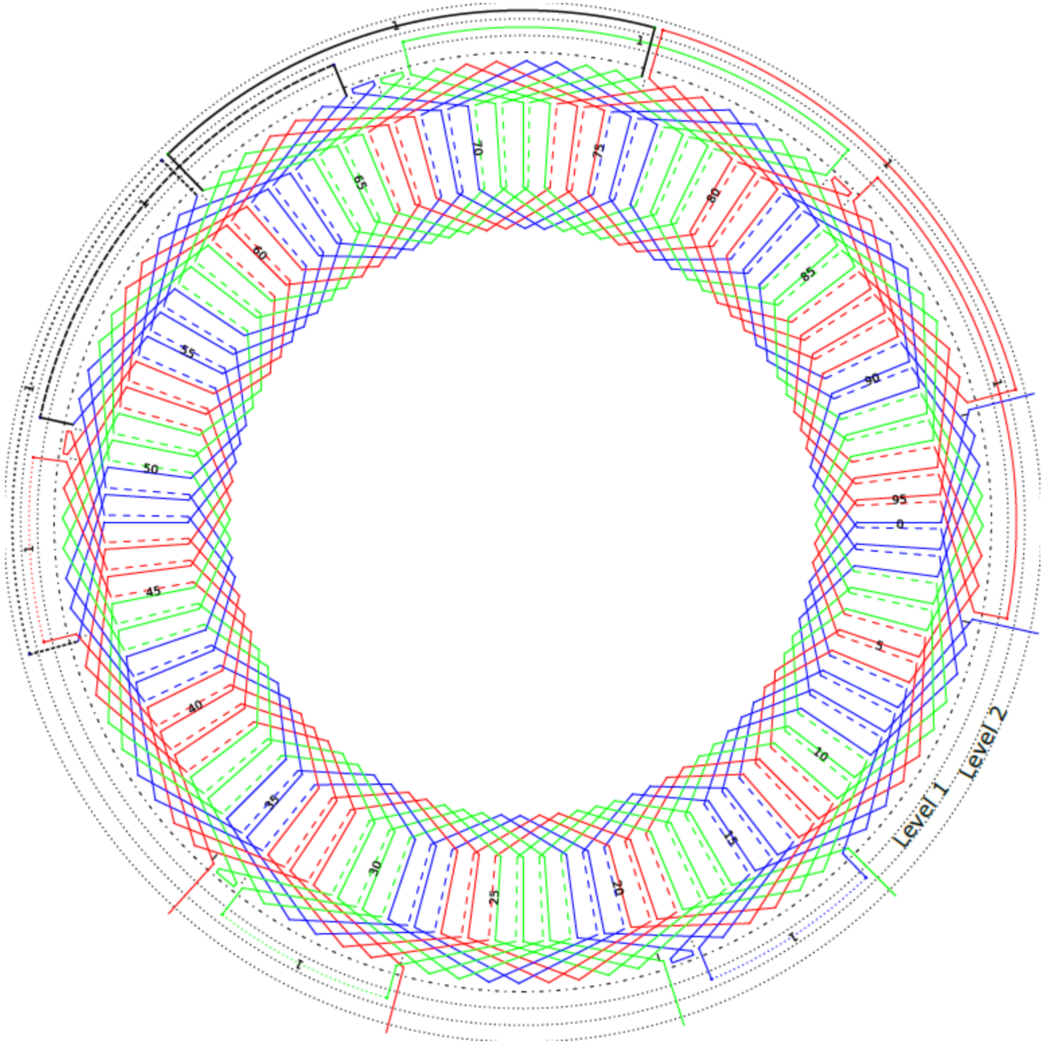
Design Optimization

1. Bar to phase assignment
2. Routing of phases
3. Jumper placement

Validation

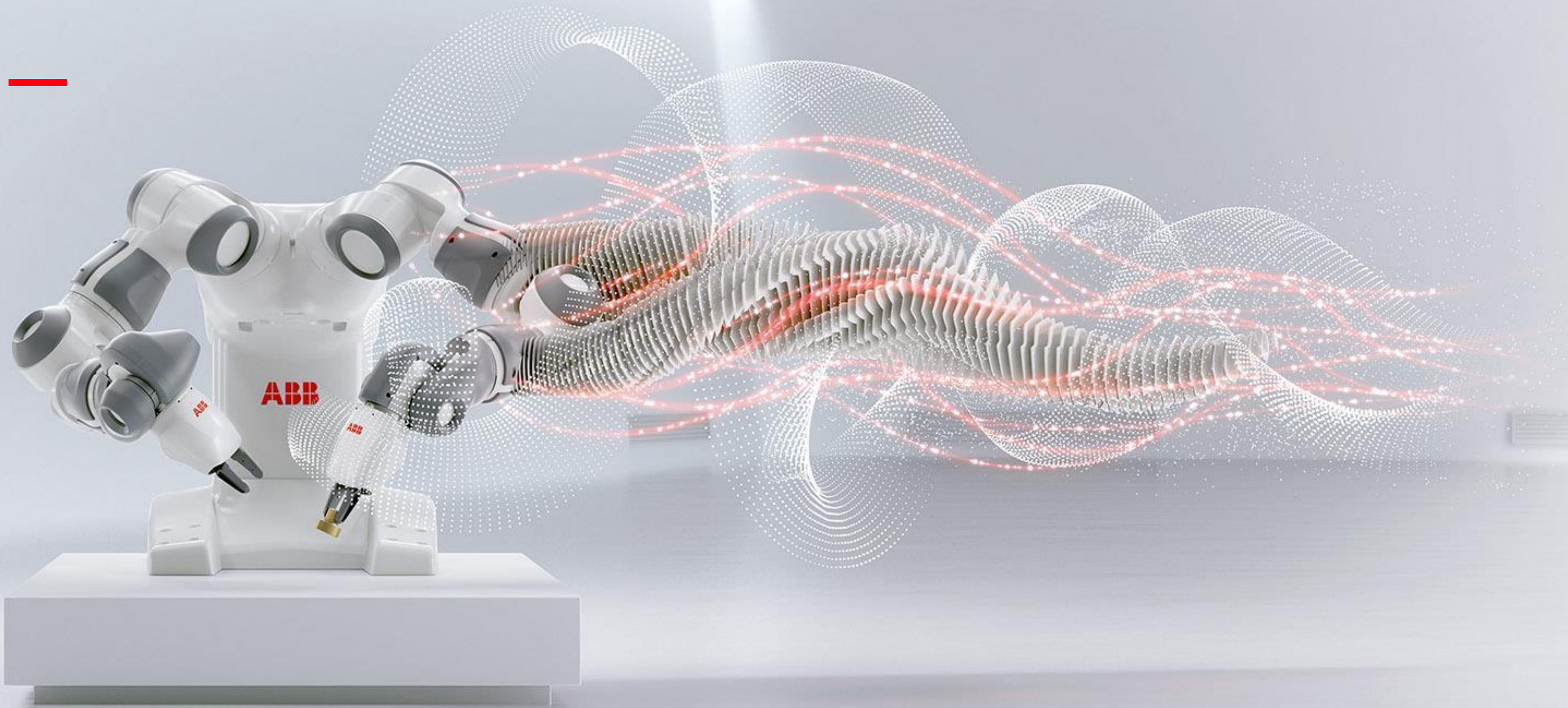
1. Comparison of different bar assignments
2. Verification of harmonics

Jumper Placement



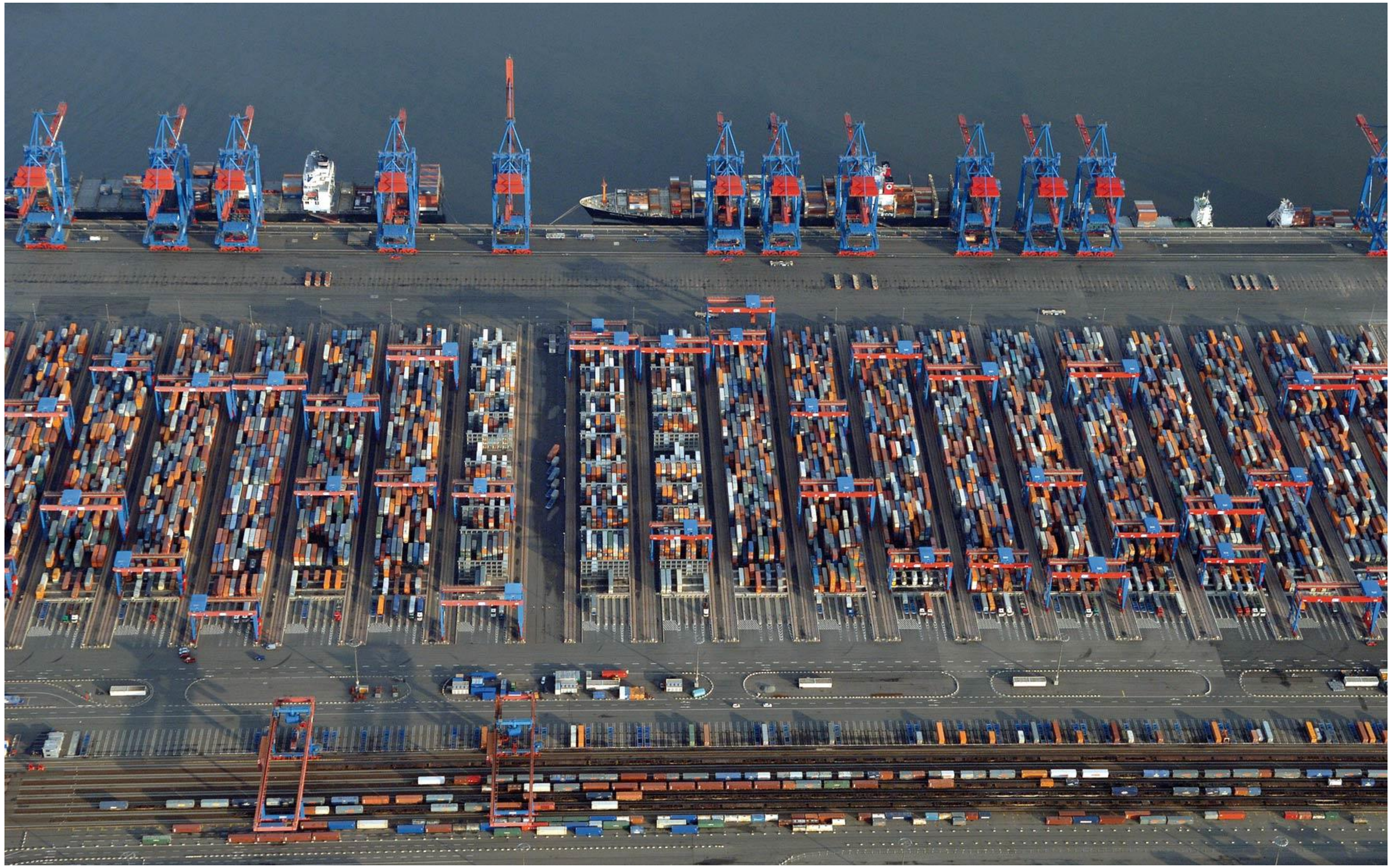
Results Routing + Jumper Placement

	Decomposed MIP+CP				Decomposed MIP				MIP			
n_s	t (μ)	t (σ)	Obj_{CP}	%Sol	t (μ)	t (σ)	$\frac{Obj}{Obj_{CP}}$	%Sol	t (μ)	t (σ)	$\frac{Obj}{Obj_{CP}}$	%Sol
102	4.4	1.0	12.18	100%	2.4	1.2	100.0%	100%	177.6	112.2	98.2%	90%
264	28.6	28.7	23.57	100%	26.0	28.9	100.0%	95%	340.7	2.0	101.7%	5%
384	23.2	19.5	25.39	100%	19.4	19.4	99.9%	95%	342.1	3.2	-	0%
480	42.0	35.6	32.34	100%	38.8	34.8	100.1%	100%	339.8	2.2	-	0%
576	65.0	33.4	43.56	70%	60.4	32.7	99.8%	30%	341.2	2.4	-	0%

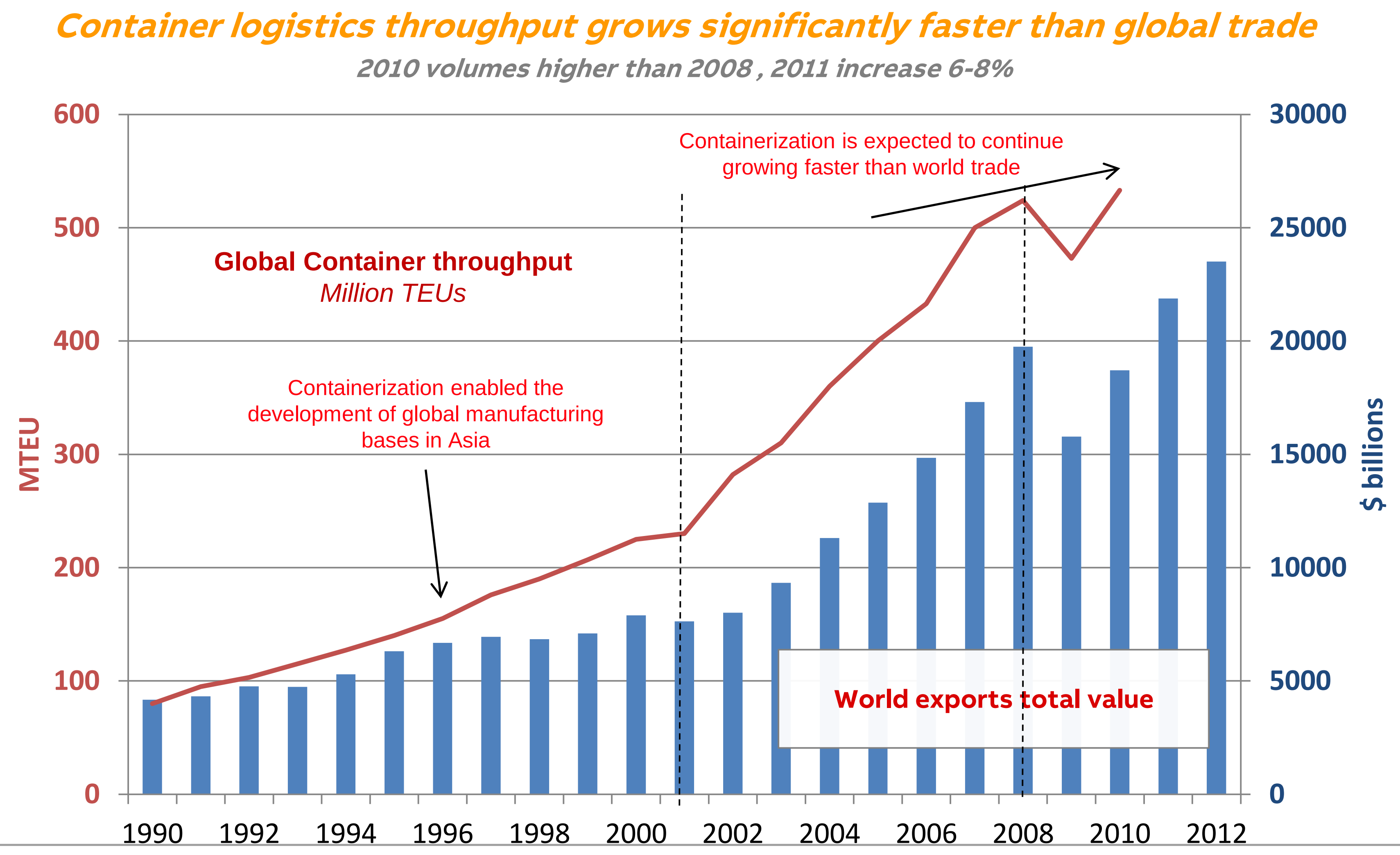


Container Terminal Optimization

Container terminals



Container trade growth



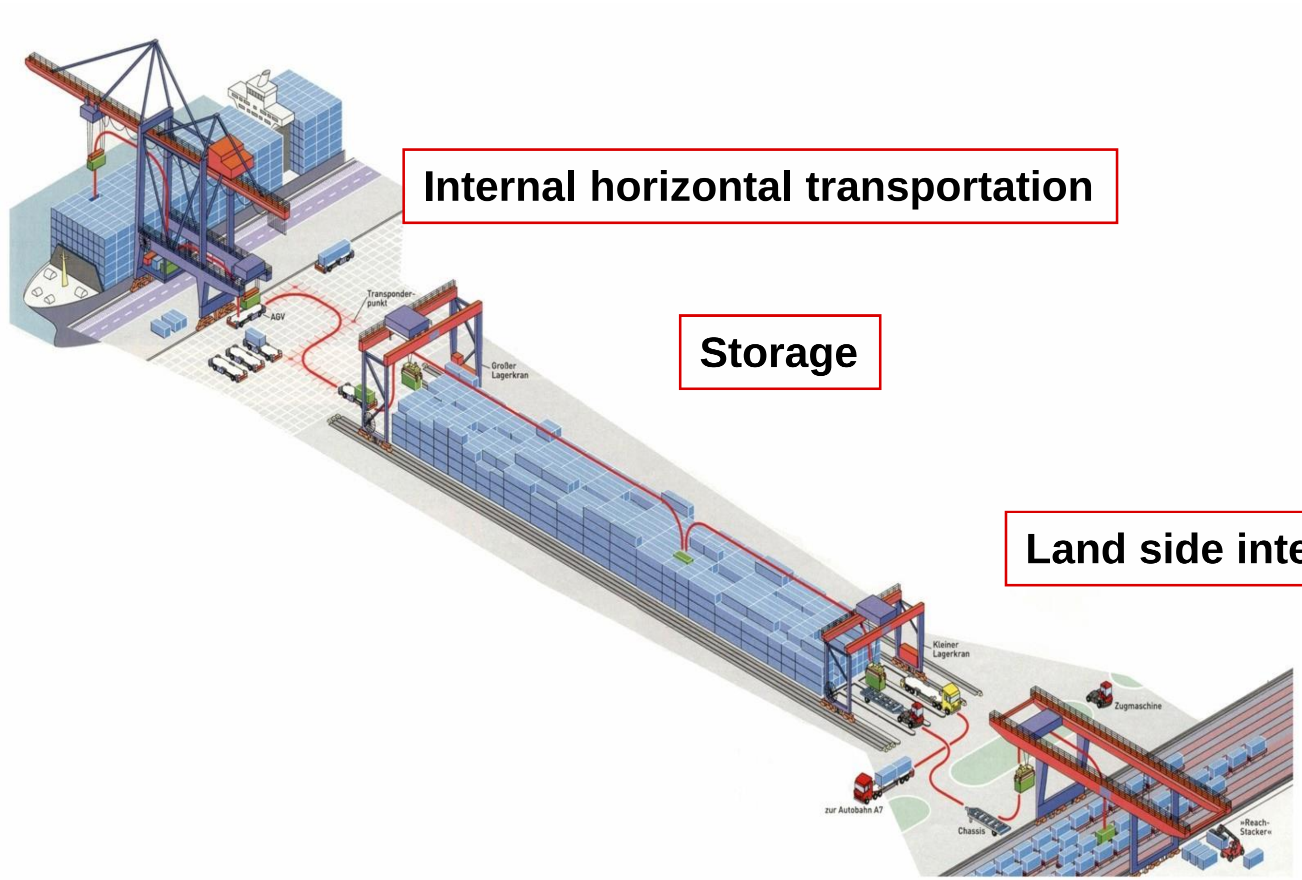
Zooming in

Off-load/load ship

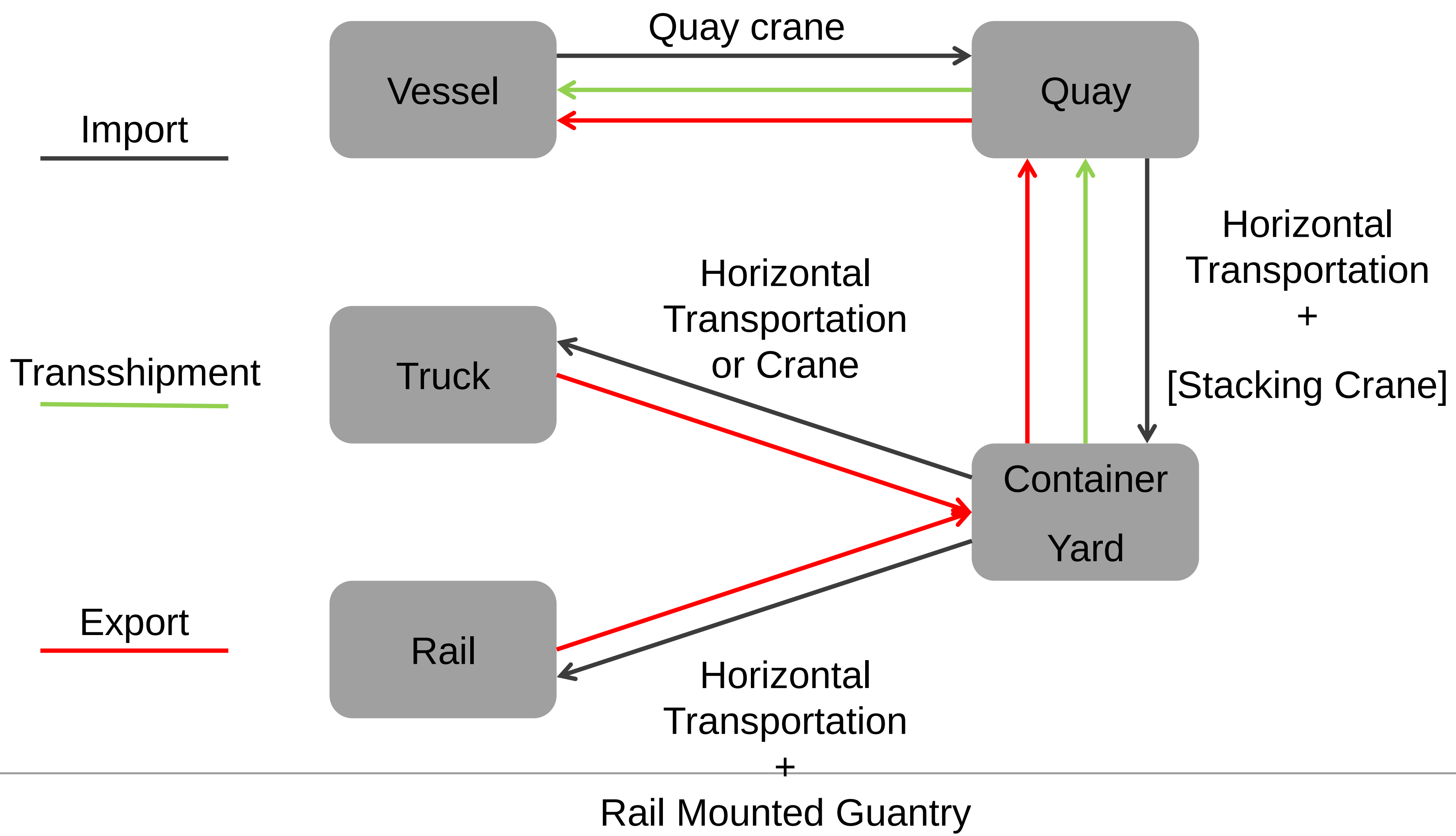
Internal horizontal transportation

Storage

Land side interface

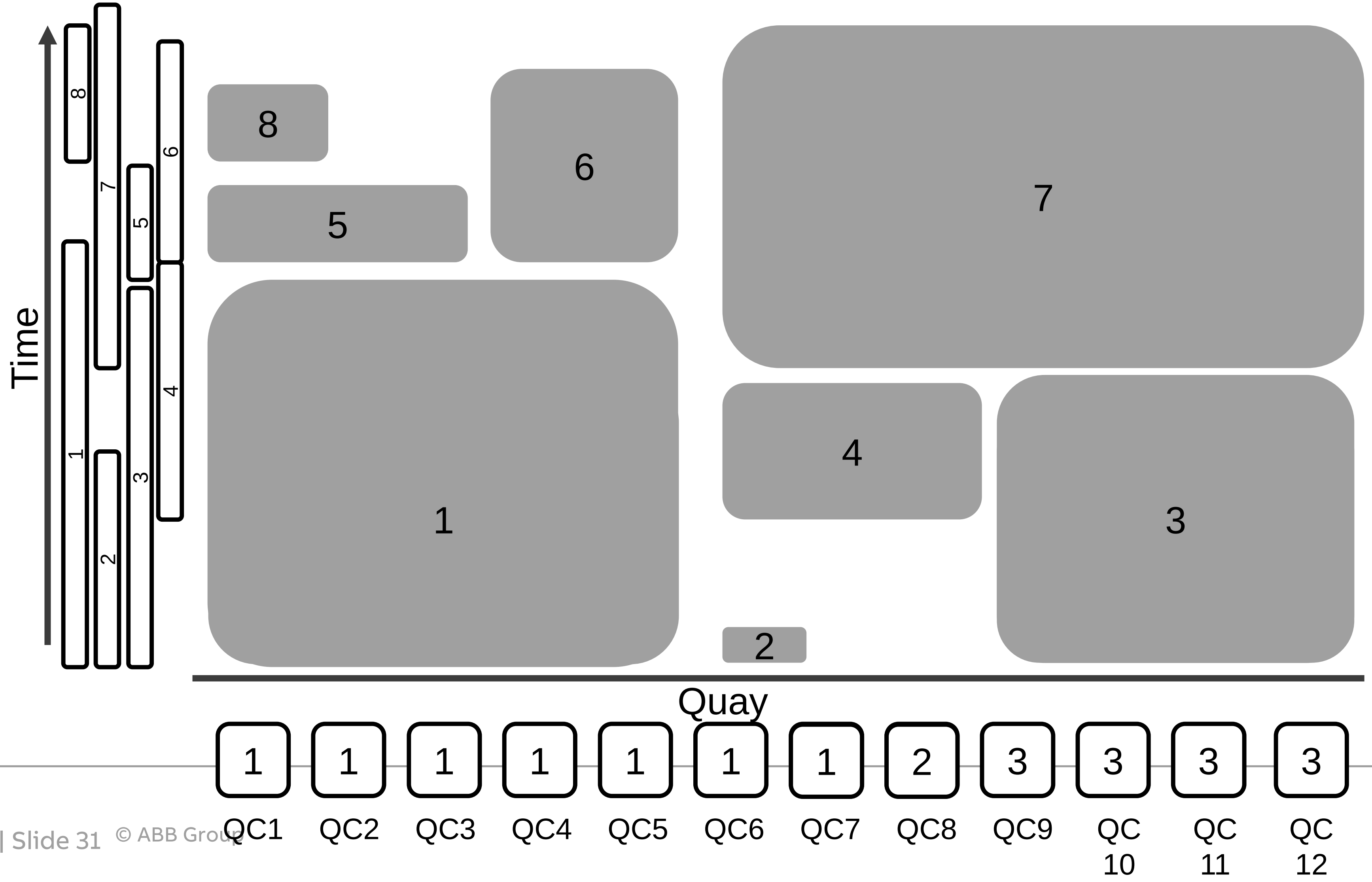


The life of a container in a terminal



Berth Allocation

Rich 2D packing problem



Berth Allocation

High Level Model

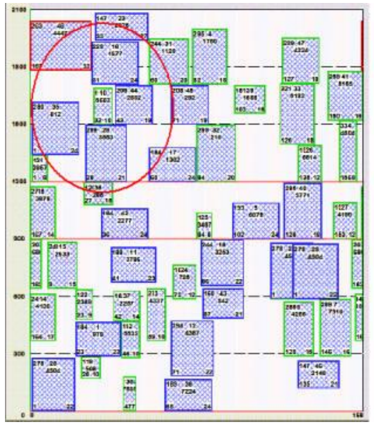
Objective function

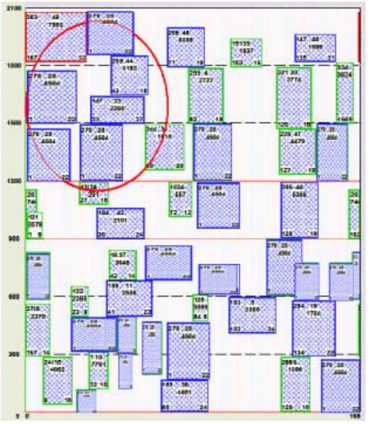
- Maximize Quay Utilization
- Minimize Lateness
- Minimize Number QC Used Per Shift
- Minimize Number QC Night Shifts
- Minimize QC Idleness

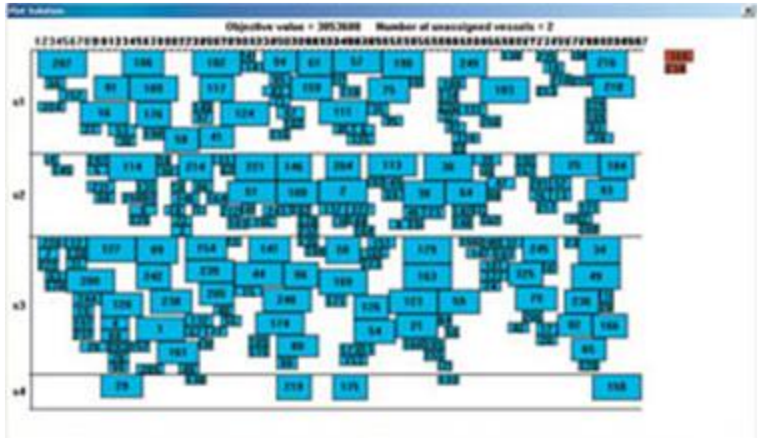
Constraints

- Space and Time Constraints
- Non Passing Cranes
- Crane/Ship Compatibility
- Maximum Number Cranes per Ship

Features: offline problem

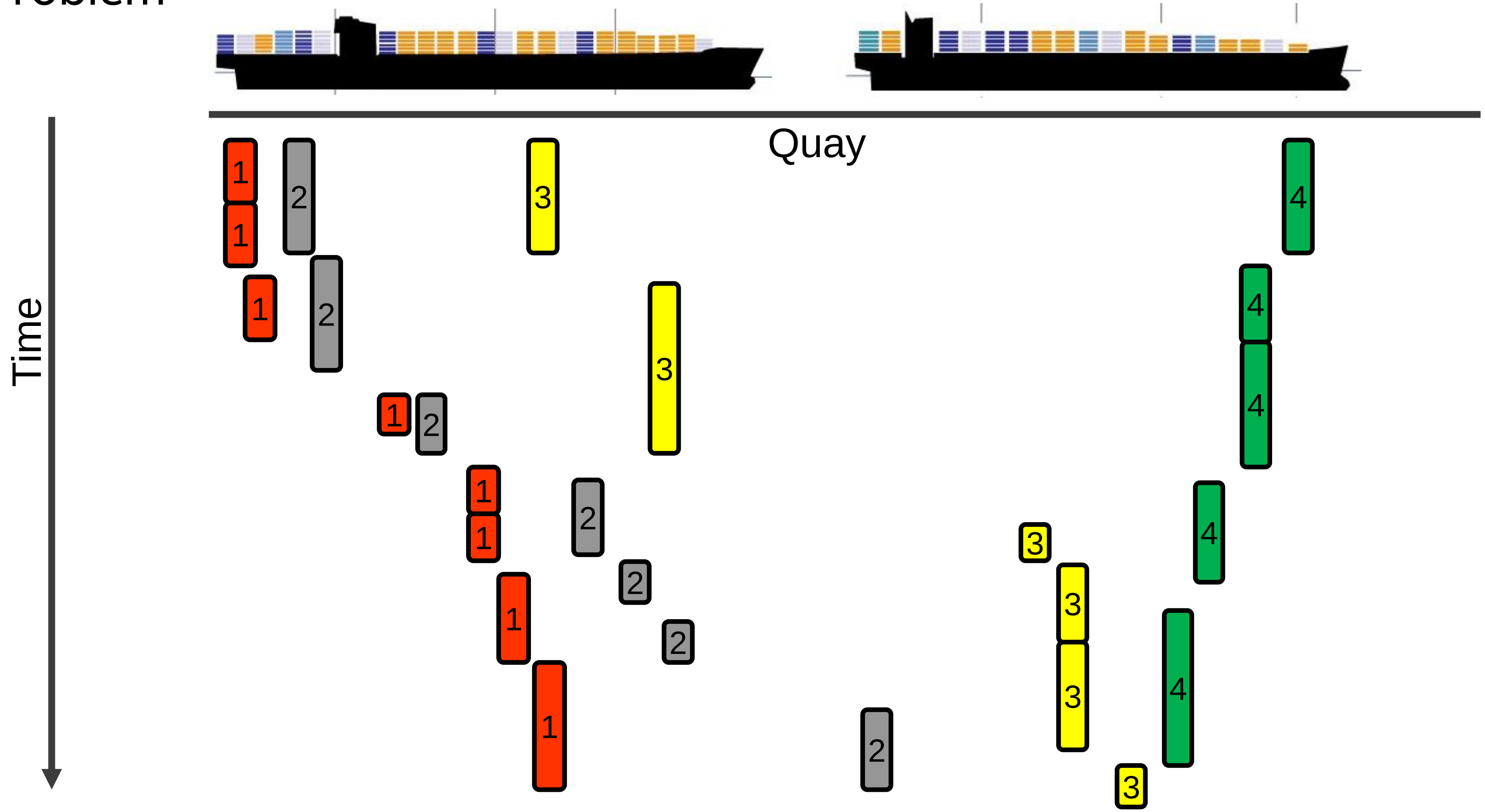






Quay Crane Allocation and Scheduling

Scheduling Problem



Quay Crane Allocation and Scheduling

High Level Model

Objective Function

- Maximize Throughput
- Minimize Interference
- Minimize QC Idleness
- Maximize Dual Cycling (single crane / multiple crane)

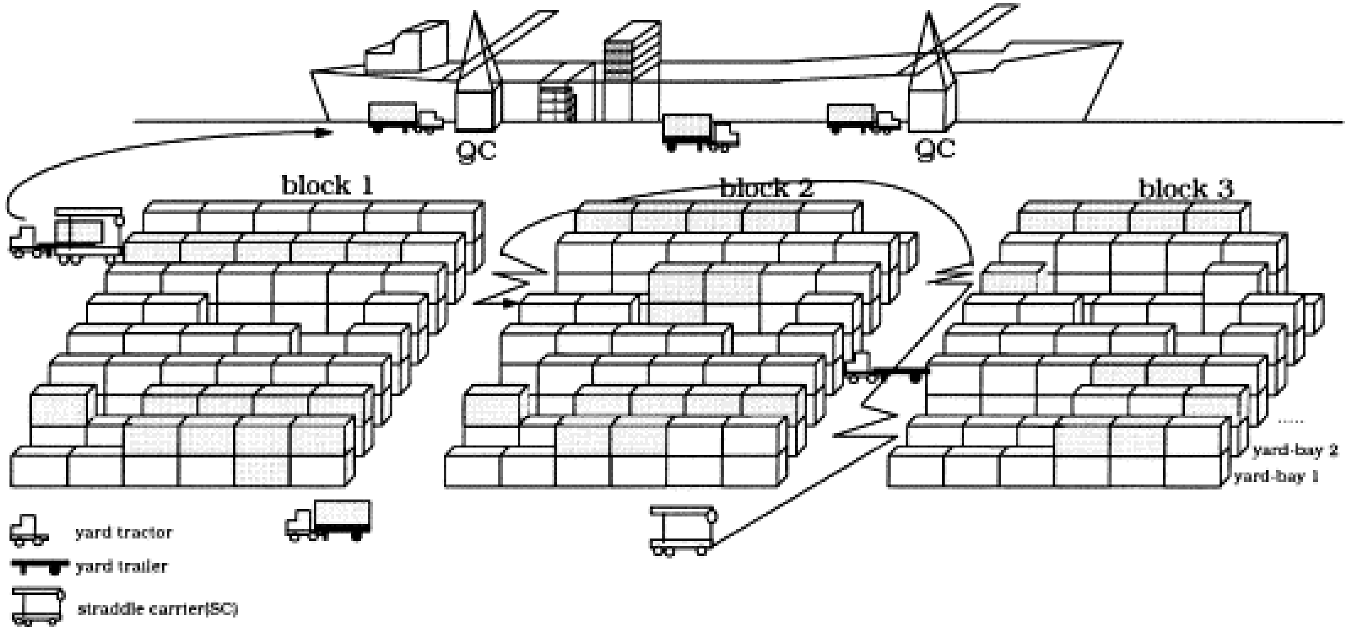
Constraints

- Safety Distance
- Non Passing Cranes
- Precedence between Working Queues
- Setup Time between Working Queues
- Boom-up / boom-down
- Crane/Ship Compatibility

Features: online and stochastic (working queue timing and QC failures)

Horizontal Transportation

Routing Problem



Horizontal Transportation

High Level Model

Objective Function

- Minimize QC/ASC Waiting Time
- Maximize Throughput (moves/hour)
- Minimize Empty Travelling Distance

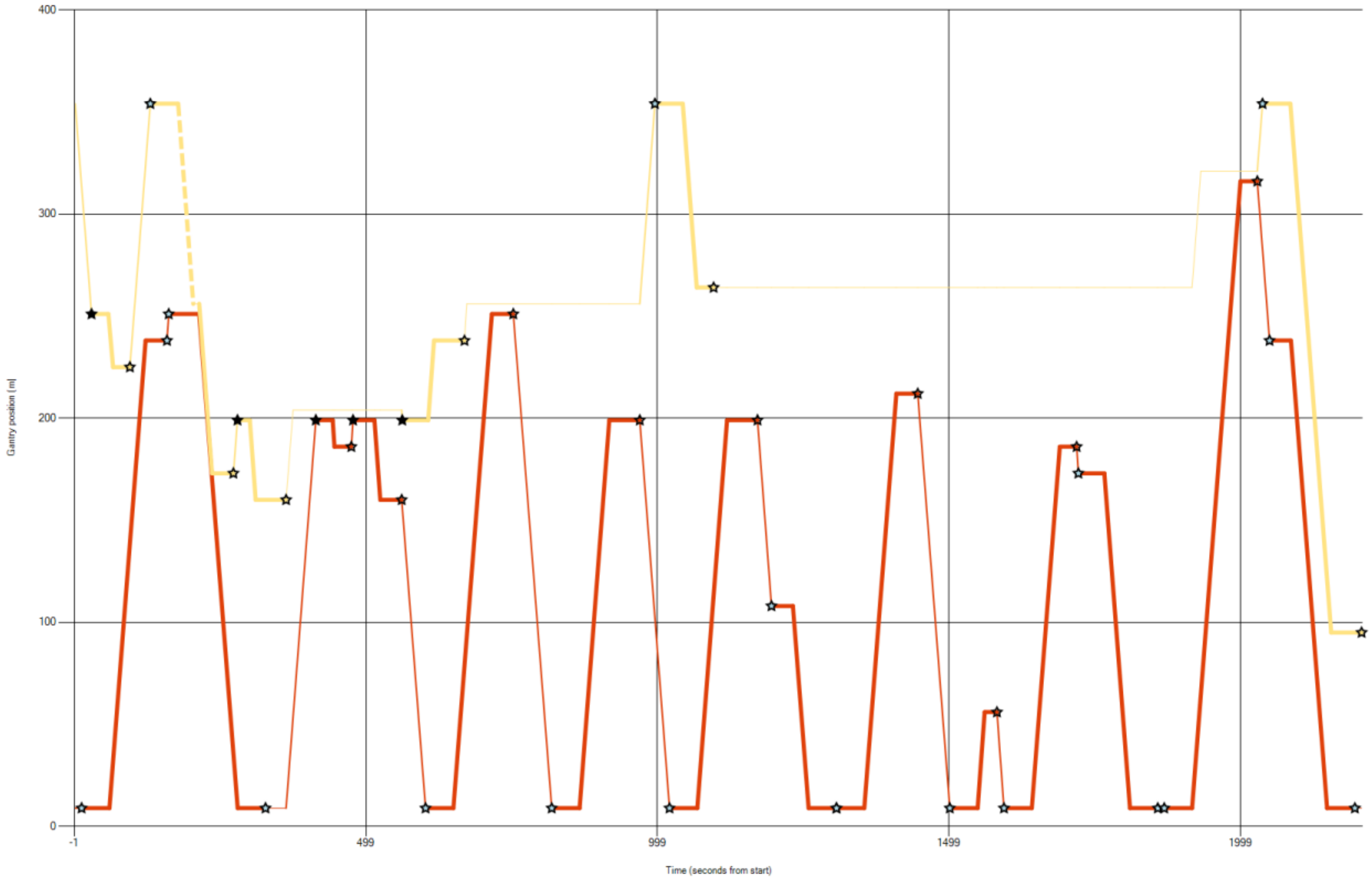
Constraints

- Precedence between Job Orders
- Job Order Time Windows (release and due dates)
- Maximum Waiting Time for Trucks [Straddle Carriers]
- Global Pooling vs Local Pooling
- Union Regulations [Manned Vehicles]

Features: online, highly stochastic (timing and job orders), data flow

Automatic Stacking Crane Scheduling [Columbus]

Scheduling Problem



Automatic Stacking Crane

High Level Model

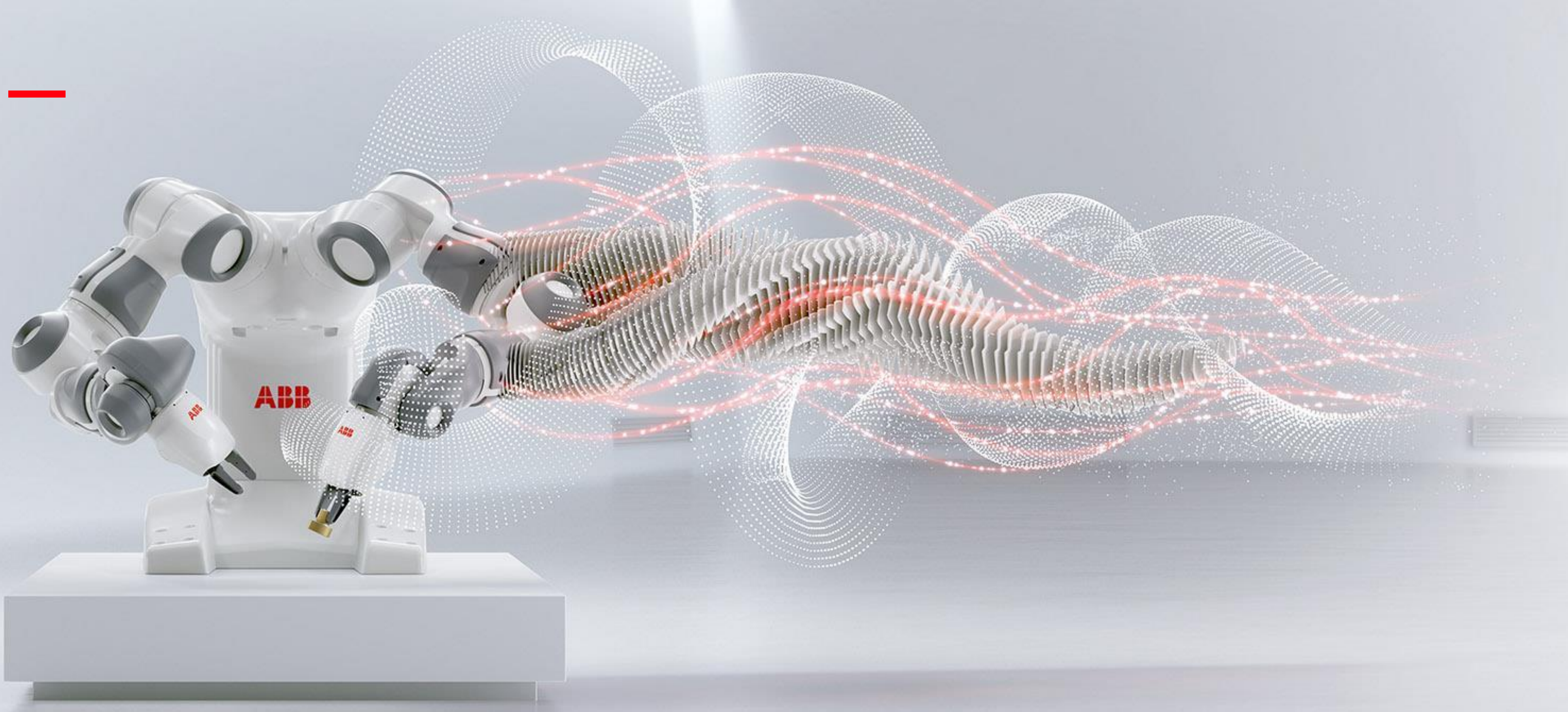
Objective Function

- Maximize ASC Throughput
- Minimize Empty Travelling Distance
- Minimize AGV/Trucks Waiting Time

Constraints

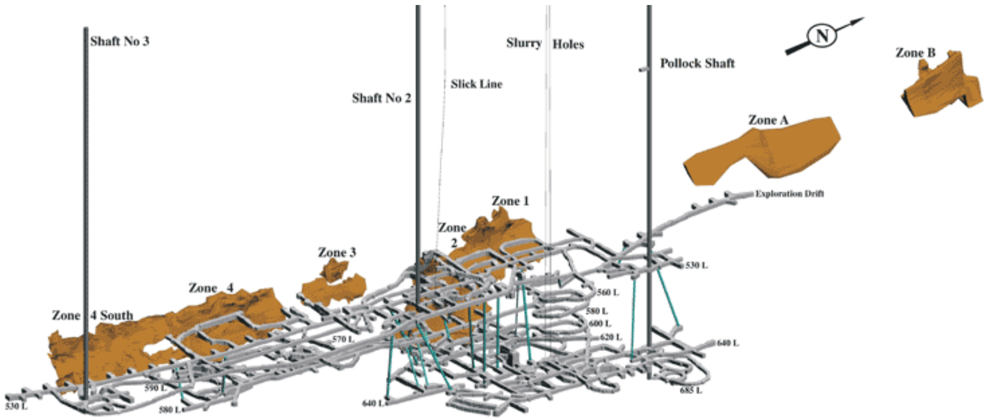
- Non Passing Cranes
- Precedence between Job Orders
- Job Order Time Windows (release and due dates)
- Coupled vs Decoupled Transfer Zone

Features: online, highly stochastic (timing and job orders), data flow



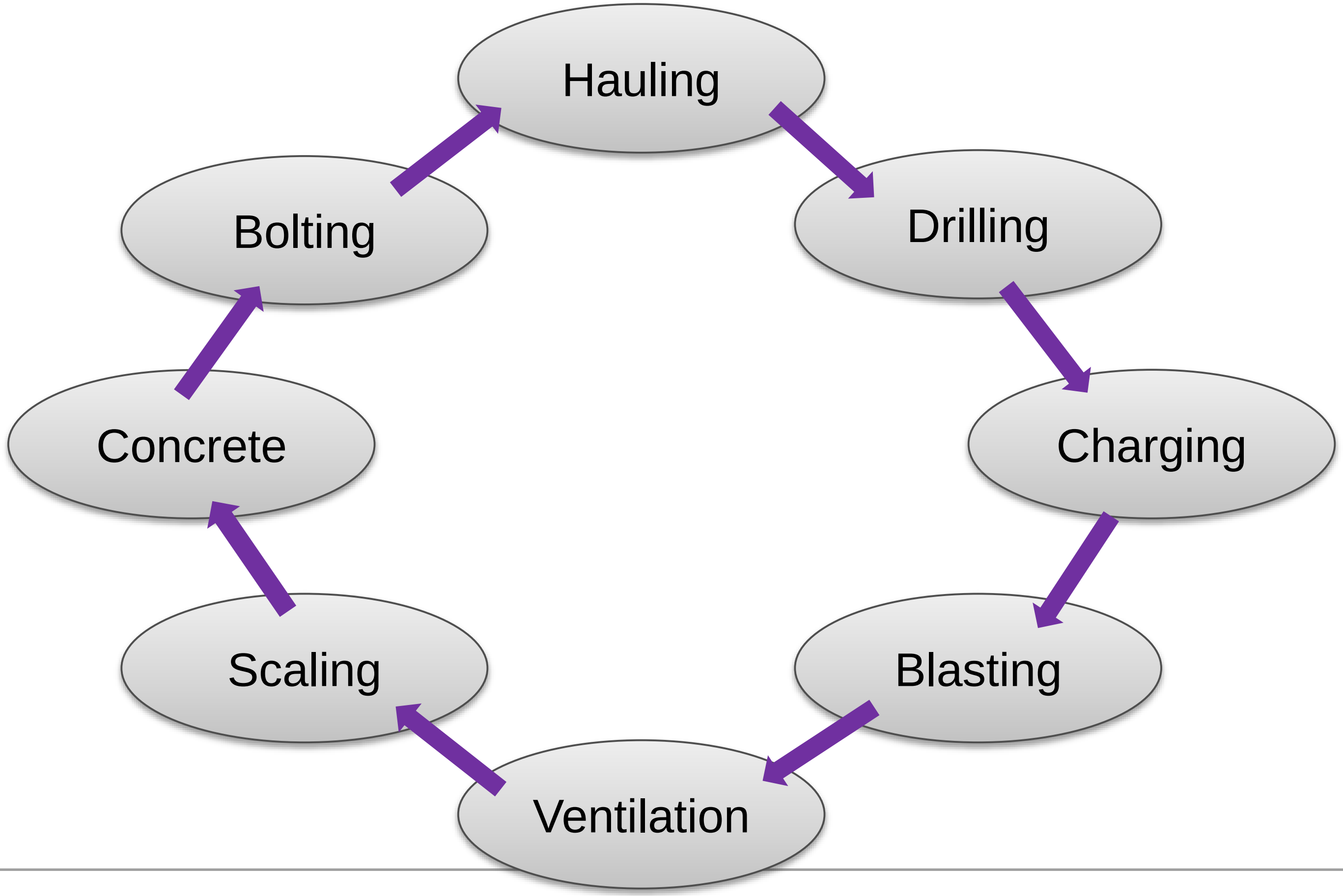
Mining industry

Underground Mine

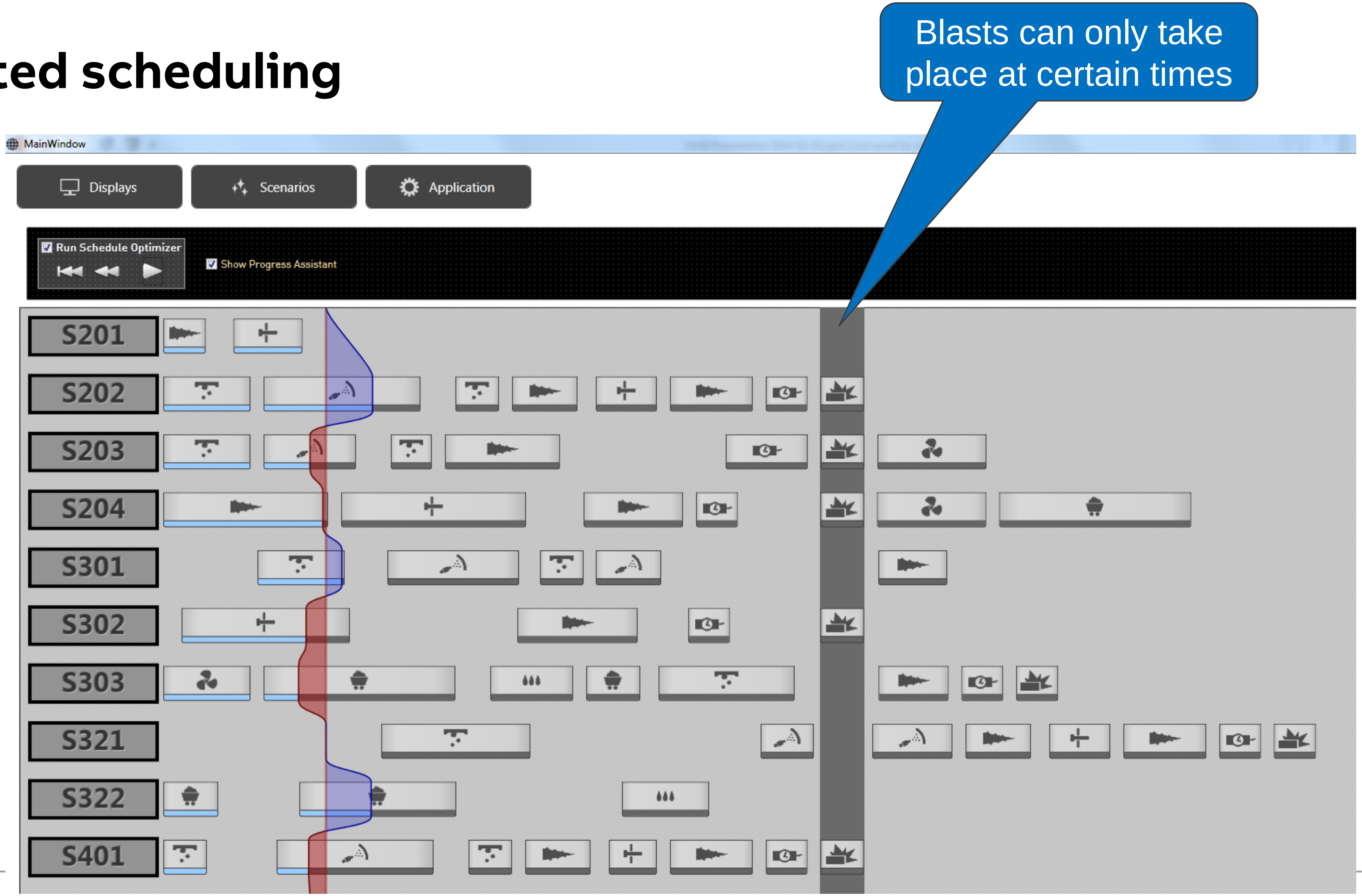


Automated scheduling

Example of drill & blast cycle

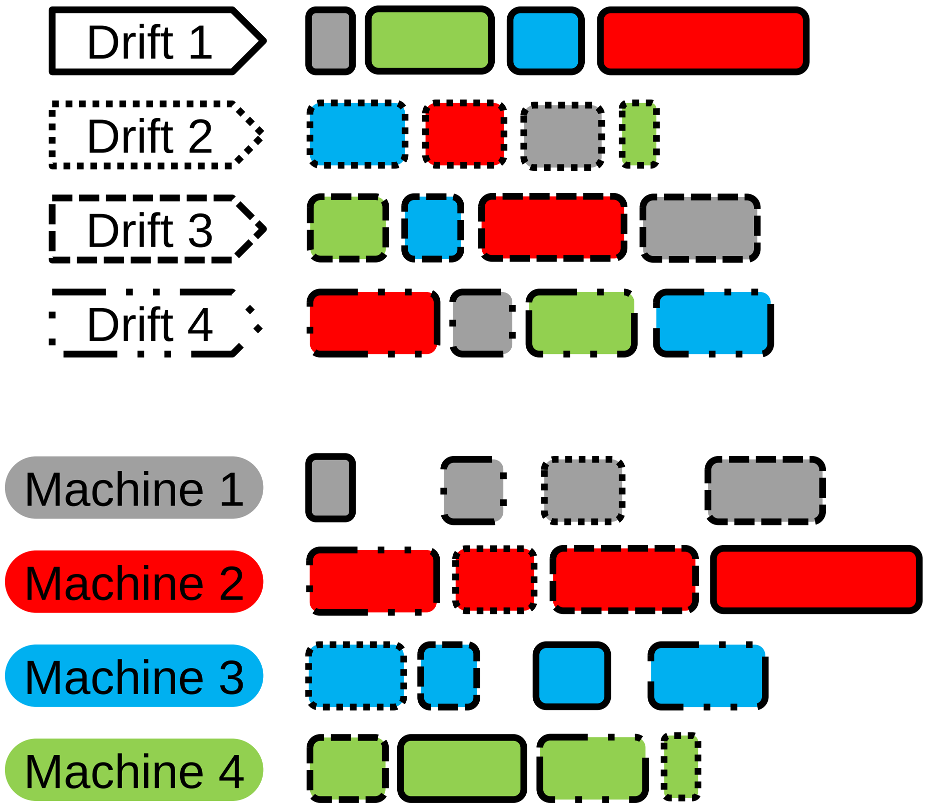


Automated scheduling



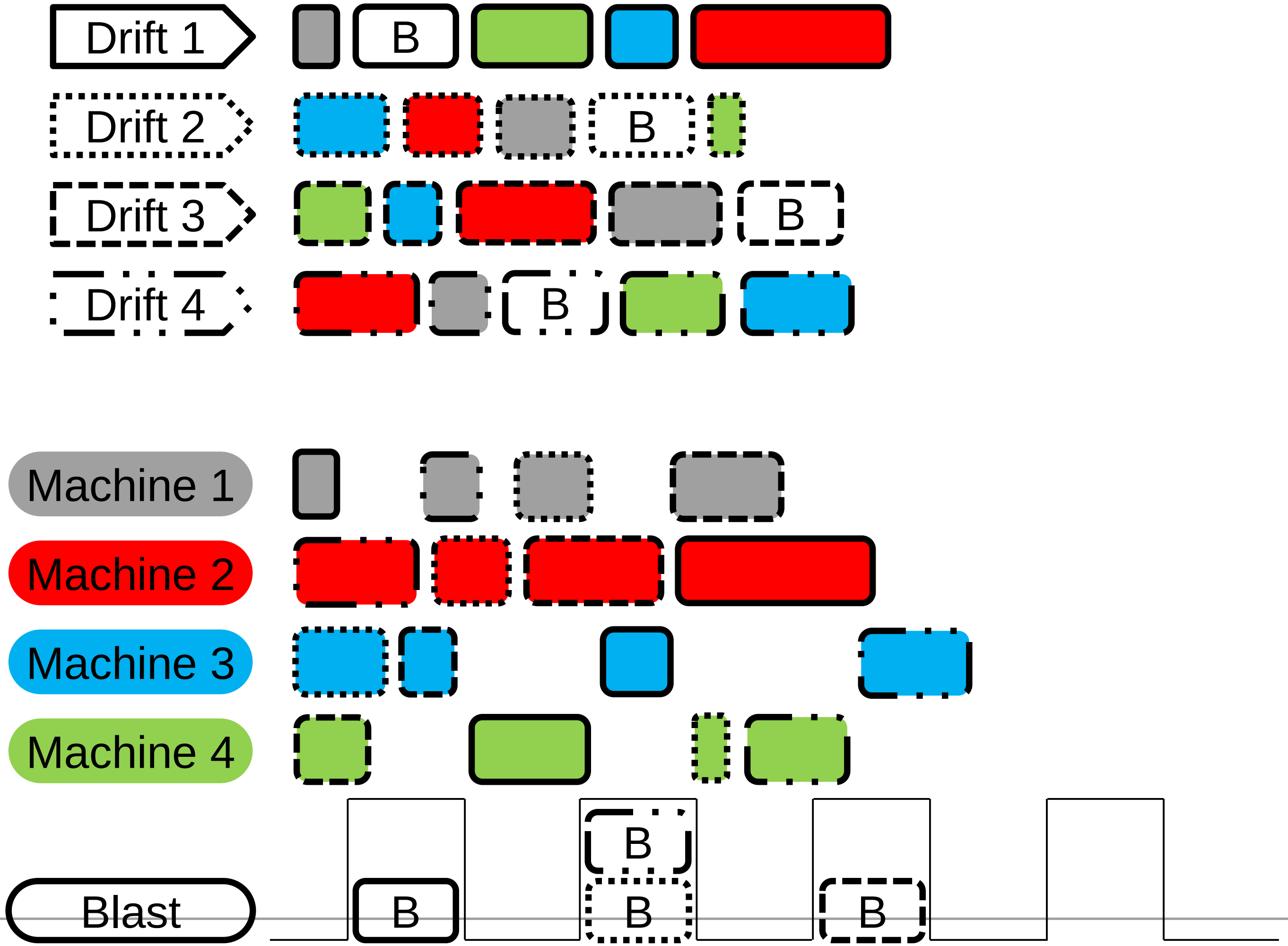
Mine Scheduling as a Rich Job Shop Problem

The pure Job Shop Problem



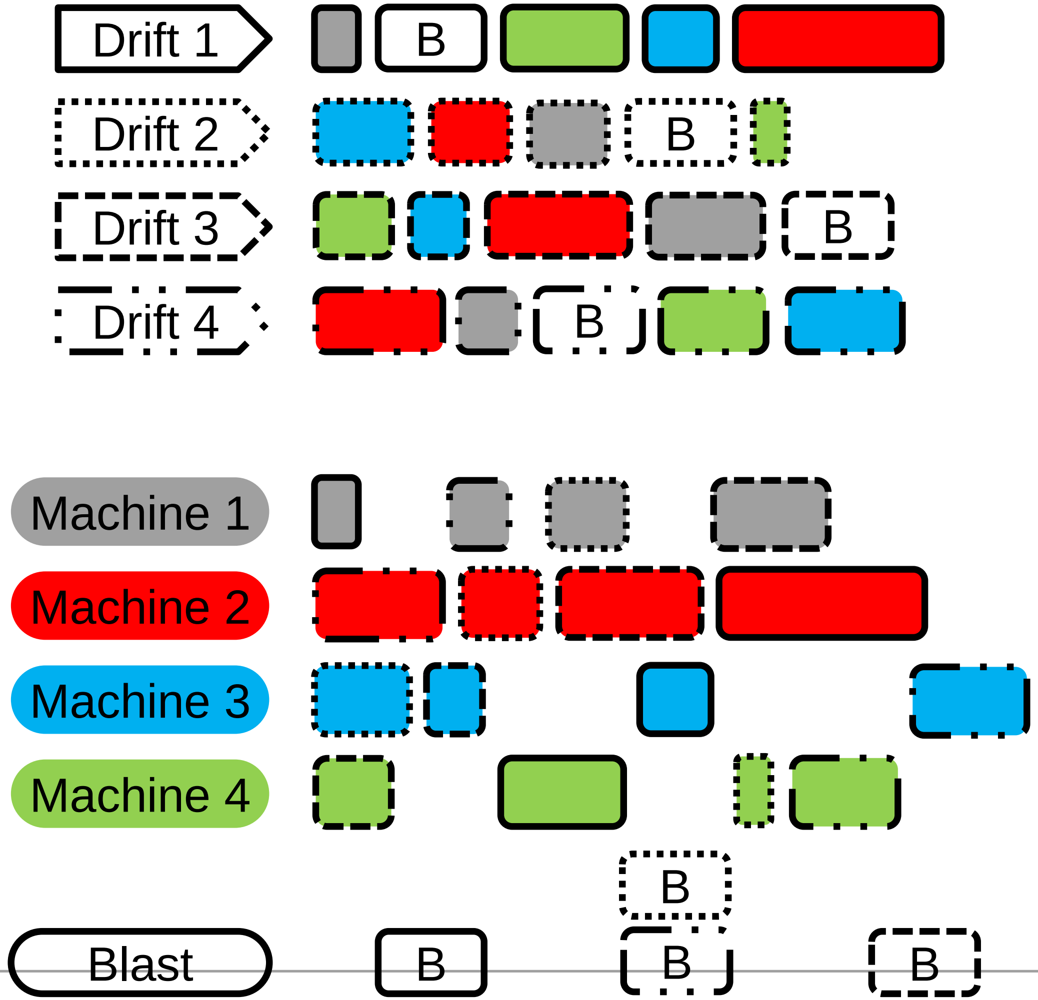
Mine Scheduling as a Rich Job Shop Scheduling

Adding blasts



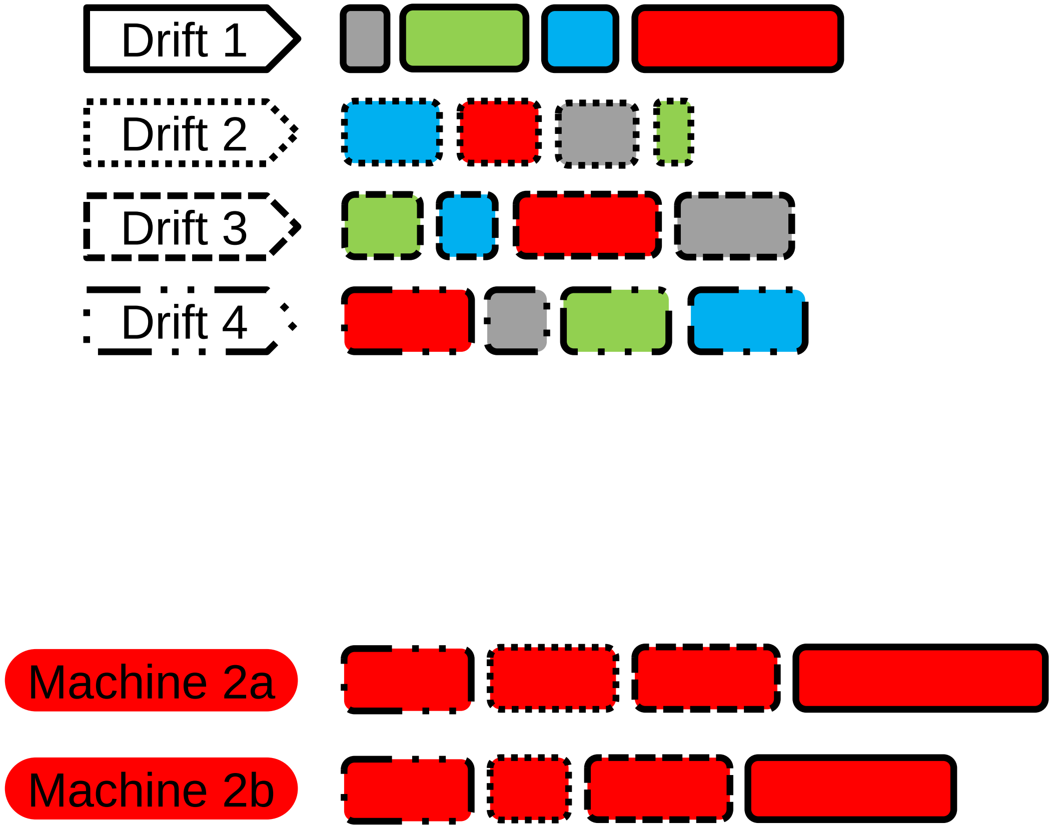
MinePROPT as a Rich Job Shop Scheduling

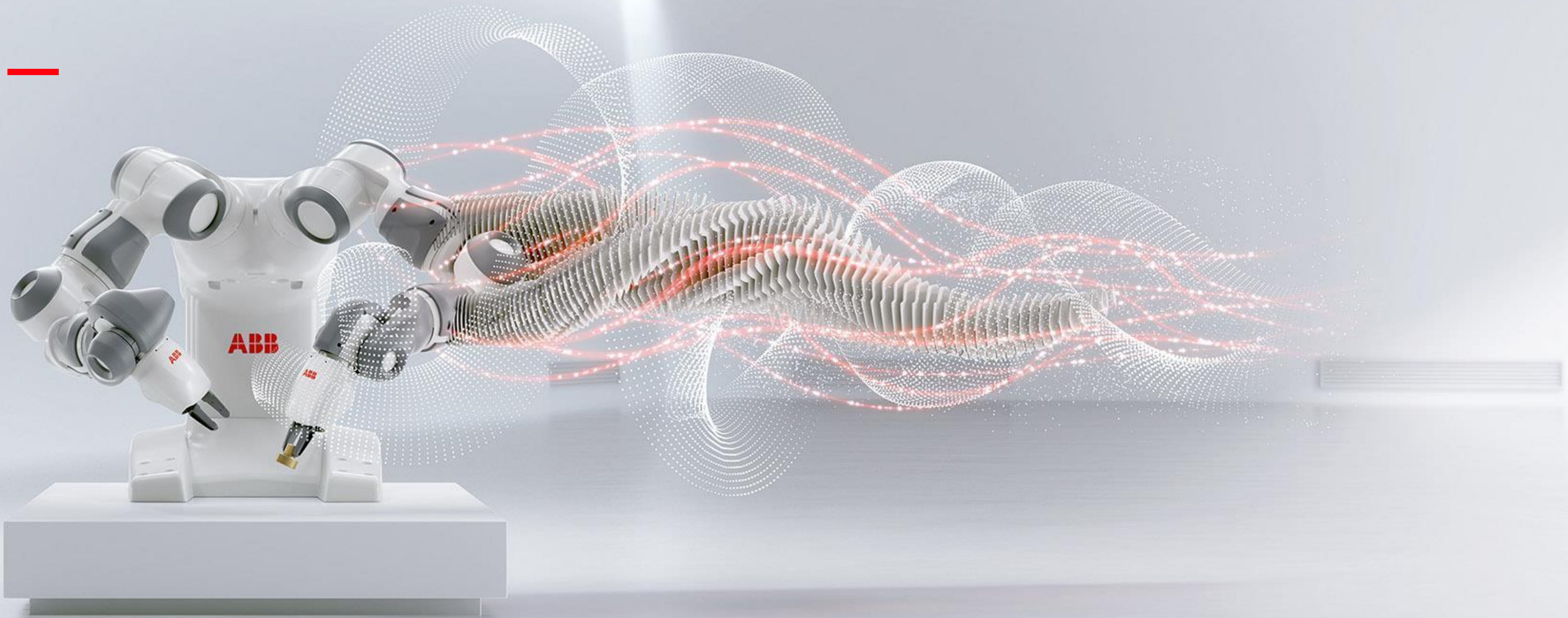
Adding Travelling time



MinePROPT as a Rich Job Shop Problem

Alternative Machines

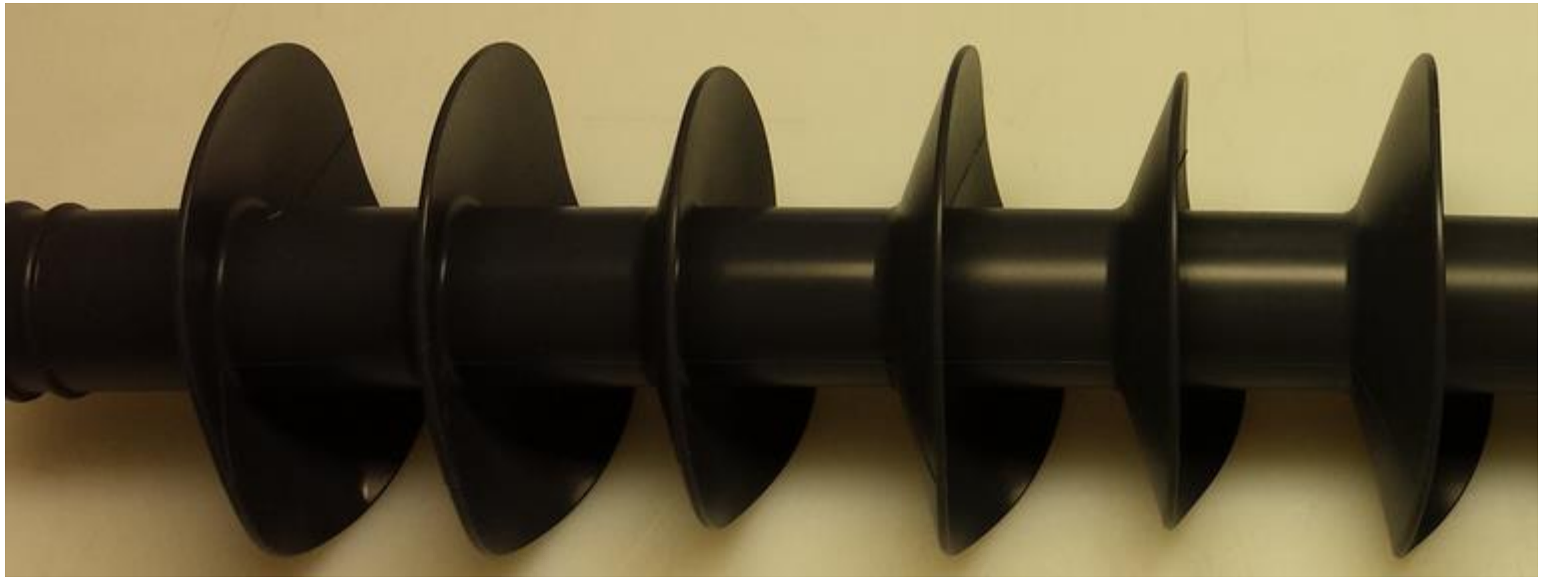




Case study

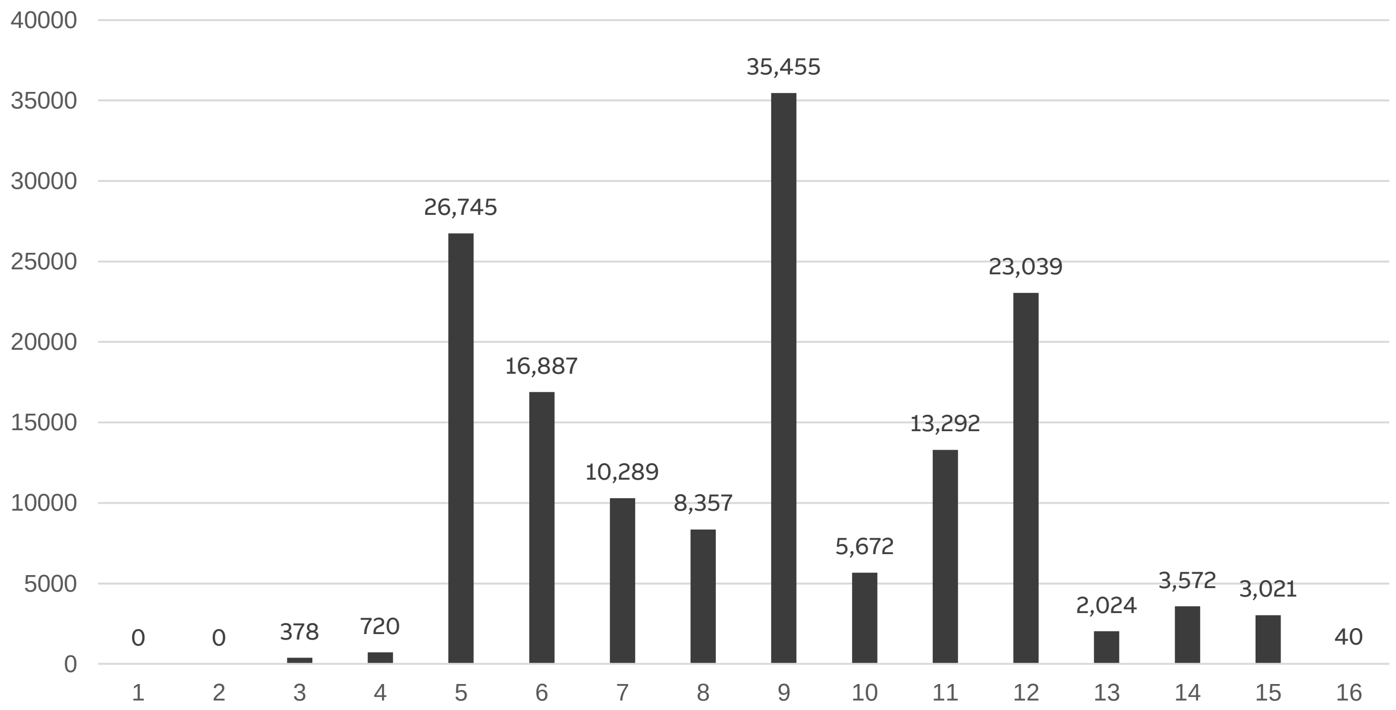
Cutting Stock Problem

Production of plastic pieces used in disaster recovery



Initial input

- A mold creates a piece with 16 flaps/discs
- Forecasted orders for year 2017



Understading the problem

Understading the problem

- What are the cost drivers?
  - Total time of production, waste, total plastic used, overproduction, cutting costs

- Is there the possibility to build a new mold?
  - Will different molds have the same yield?
  - Will different molds have the same throughput?

- Are the production requirements constant or they may vary on subsequent years (i.e. stochastic)?

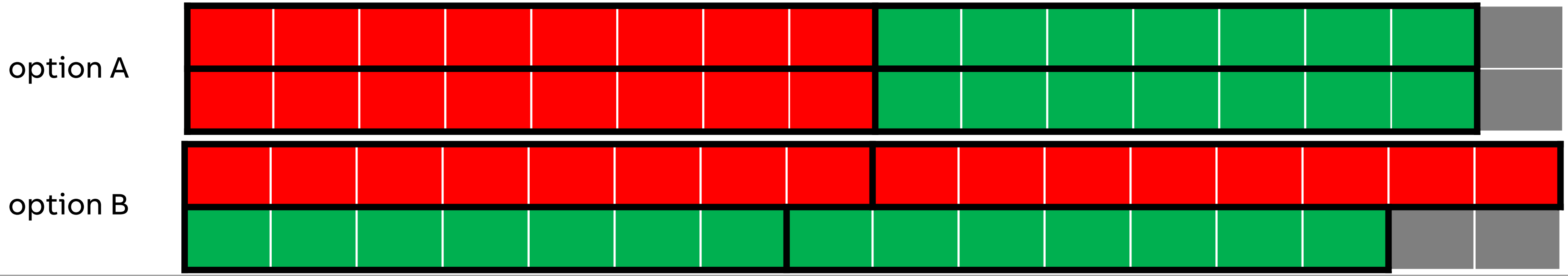
- Is the yield of the cutting procedure constant?

- *Size of the problem?*

Actual problem

- Decision variables
  - Which mold length to create
  - Which combination of molds to use subject to given production requirements
  - Which cutting patterns to use subject to given production requirements

- Minimize
  - Waste
  - Over-production
  - Number of cuts



Item-based formulation (Kantorovich)

Second Stage problem

Variables

$x_{ij} = k$ → integer variable, item "i" is cut out of stock "j", "k" times

$y_j = \{0,1\}$ → binary variable, whether stock "j" is used or not

$z_j = \{0,1\}$ → binary variable, whether stock "j" produces waste or not

Constraints

$\sum_j x_{ij} \geq d_i$ for all i → all the production requirements must be met

$\sum_i l_i x_{ij} \leq L y_j$ for all j → the total length of item in stock j must not exceed stock length

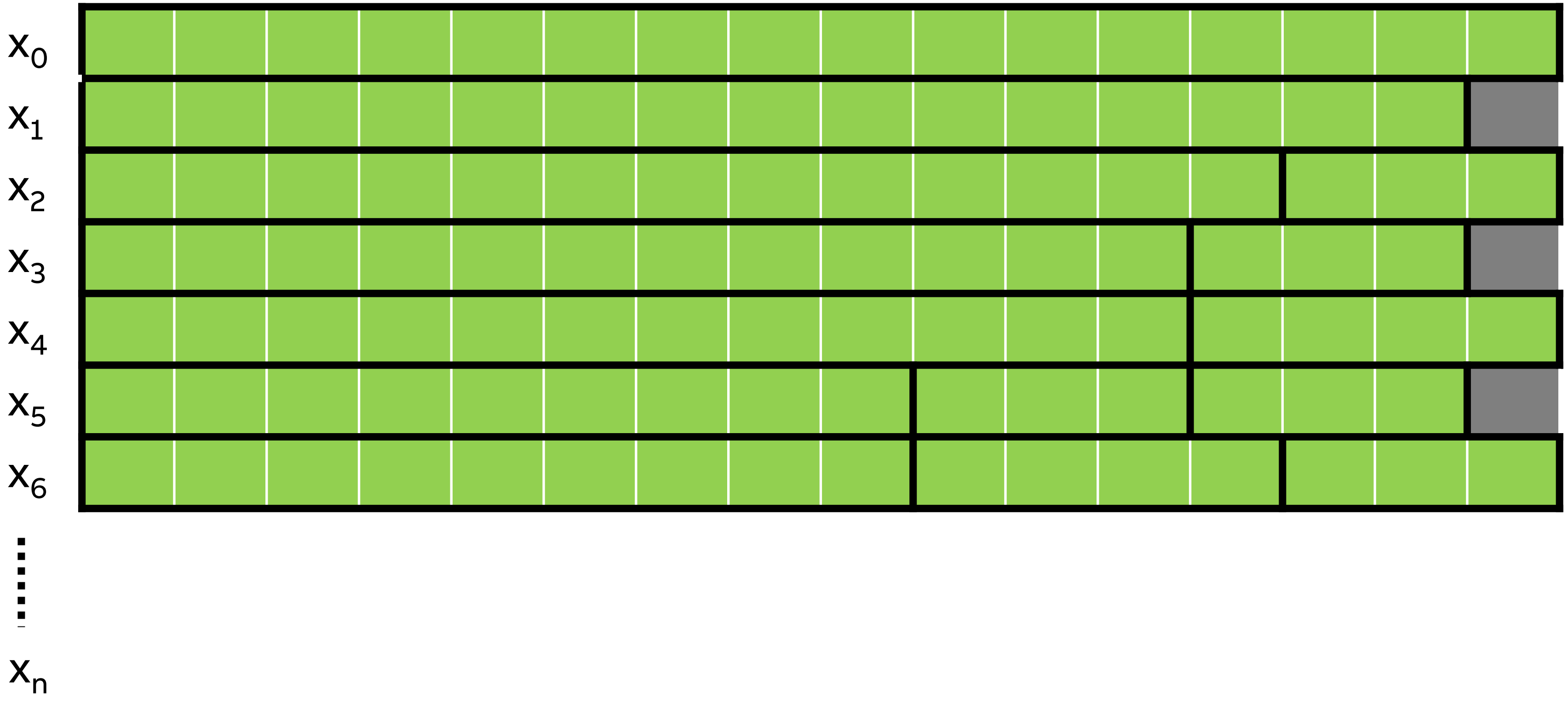
$L_j y_j - \sum_i l_i x_{ij} \leq M z_j$ for all j → z must be equal to 1 if stock j creates waste

Objective function

$$\min \alpha_1 \underbrace{\sum_i c_i (\sum_j x_{ij} - d_i)}_{\text{overproduction}} + \alpha_2 \underbrace{\sum_j (L y_j - (\sum_i l_i x_{ij}))}_{\text{waste}} + \alpha_3 \underbrace{(\sum_j z_j)}_{\text{number of cuts*}}$$

Pattern-based formulation (Gilmore and Gomory)

Second Stage problem



Resolution method

Pattern-based formulation (Gilmore and Gomory)

Generation of patterns

Variables

$z_i = k$ → integer variable, number item "i" is cut out "k" times

$w = \{0,\ldots, L\}$ → integer variable, waste of the pattern

$o = \{0,\ldots, L-1\}$ → integer variable, number of cutting operations

Constraints

$L = \sum_i l_i z_i + w$ → length constraint

$o = \sum_i l_i z_i - 1 + (Q > 0)$ → number of cutting operations

Pattern-based formulation (Gilmore and Gomory)

Second Stage problem

Variables

$x_j = q$ → integer variable, pattern "j" is used "q" times

Constraints

$\sum_j p_i x_j \geq d_i$ for all i → all the production requirements must be met

Objective function

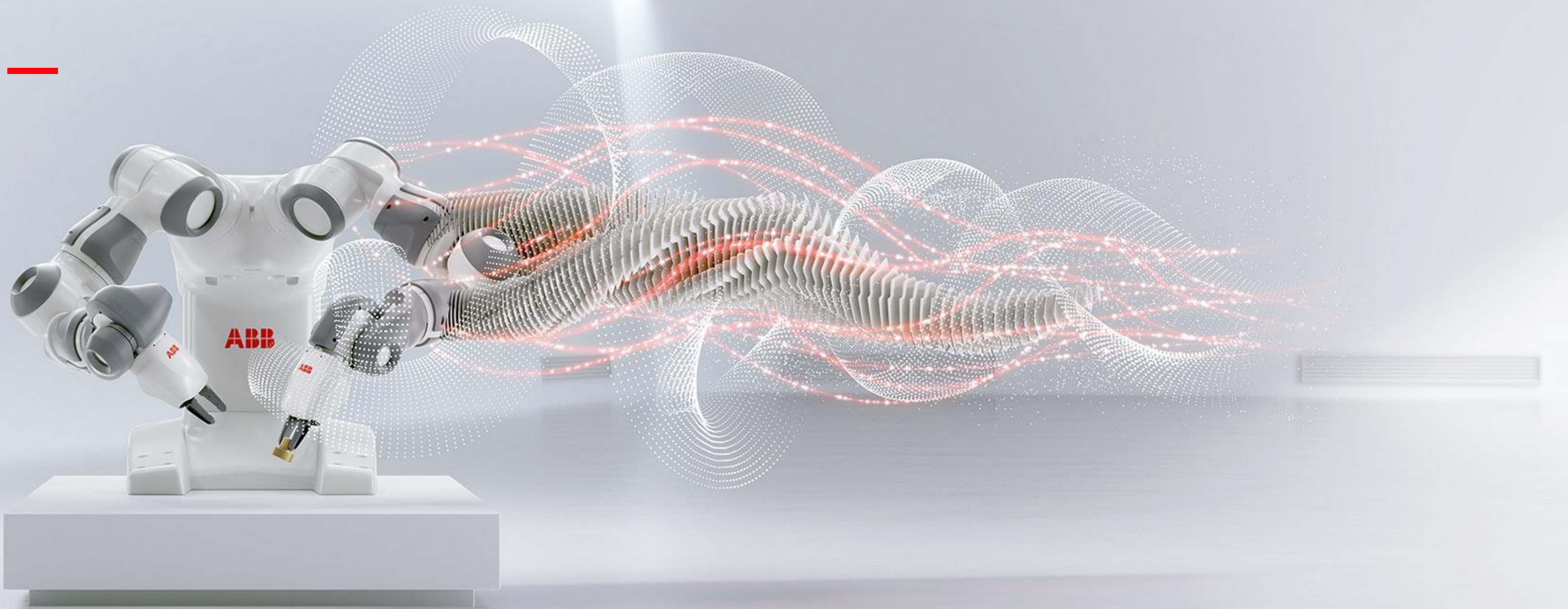
$$\min \alpha_1\Big(\underbrace{\textstyle\sum_i c_i \big(\sum_j p_i x_j - d_i\big)}_{\text{overproduction}}\Big) + \alpha_2\Big(\underbrace{\textstyle\sum_j w_j x_j}_{\text{waste}}\Big) + \alpha_3\Big(\underbrace{\textstyle\sum_j o_j x_j}_{\text{number of cuts*}}\Big)$$

Experimental results and observations

- Item-based formulation performed poorly when adding over-production, and number of cuts

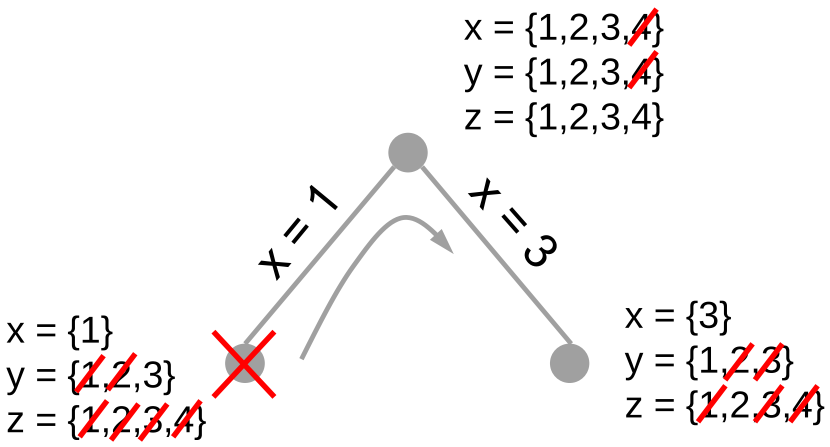
- Pattern enumeration
  - Length 15 → 40 patterns (2 msec)
  - Length 25 → 328 patterns (28 msec)
  - Length 35 → 1995 patterns (300 msec)

- Instances solved within one second (length 16)
- Linear relaxation → within 0.03% of optimal integral solution
- Given the optimal solution in term of waste and overproduction, difference in term of cutting operations is 10% (for 150thousands items → ~50hours of work)



Constraint Programming in a nutshell

Constraint Programming in a Nutshell



- $x + y + z \leq 6$
- $x + y = 4$
- alldifferent(x,y,z)
- $x * z > 2$

CP = Model + Search

Pro and Cons of Constraint Programming

PROS

Formulation strengths

y[x] = 1 (Element constraint)

((x = 1) AND (y ≤ 2)) → ((z + q > 5) OR (q = 1)) (Reified constraint)

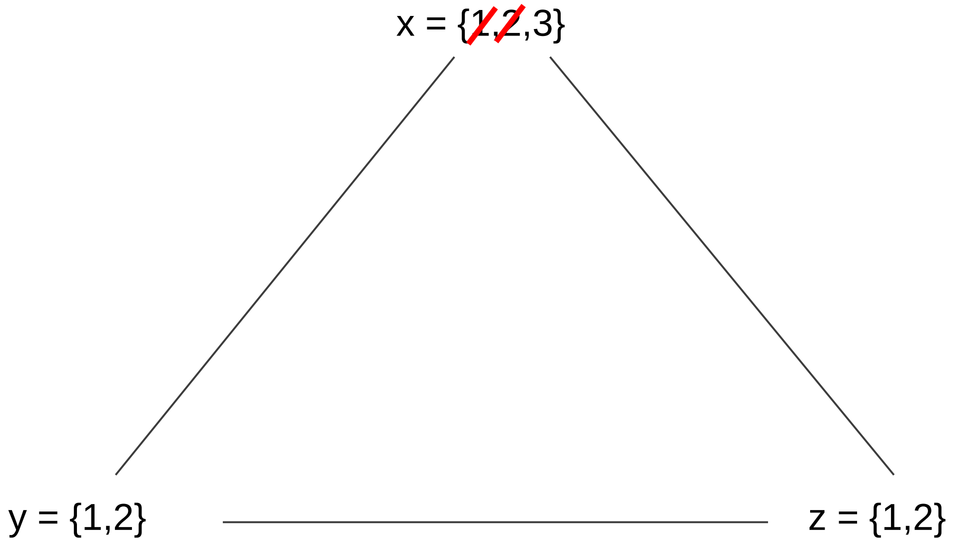
Global Constraints

- Increased filtering
- Higher level abstraction

CP Effective for problems with strong feasibility aspects

Particularly suited for Scheduling Problem

Global Constraints – alldifferent(x,y,z)



Pro and Cons of Constraint Programming

CONS

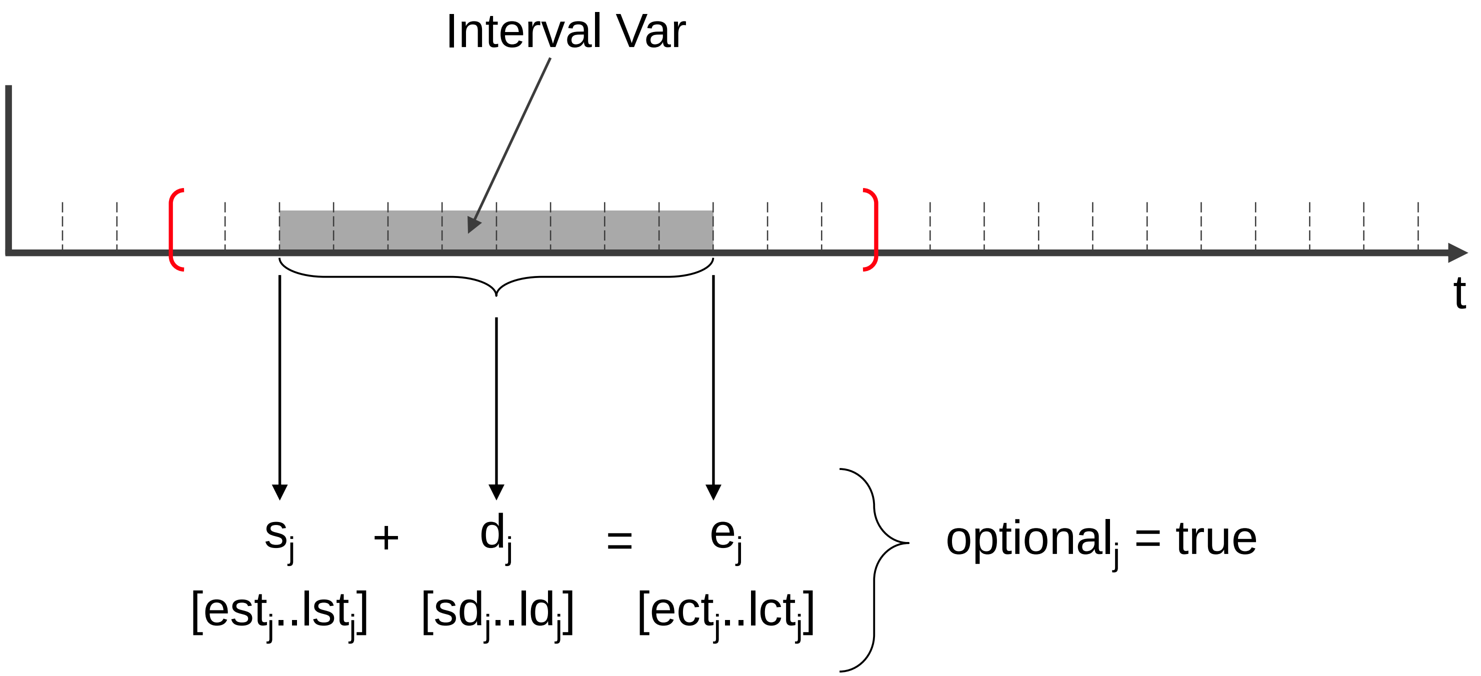
Very weak bounds compared to MIP

Inefficient for pure optimization problems
[Hybridization with MIP and/or Metaheuristic gives very good results]

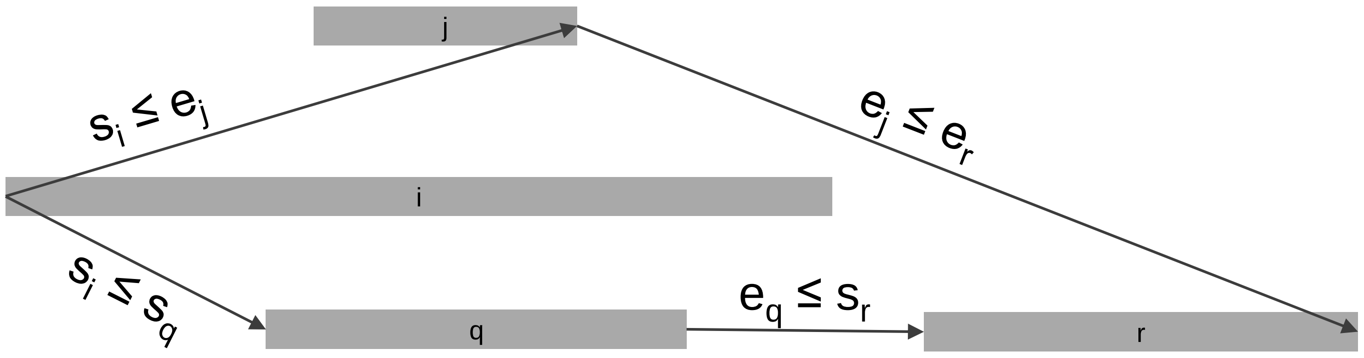
Typically needs hand-tailored heuristics for branching

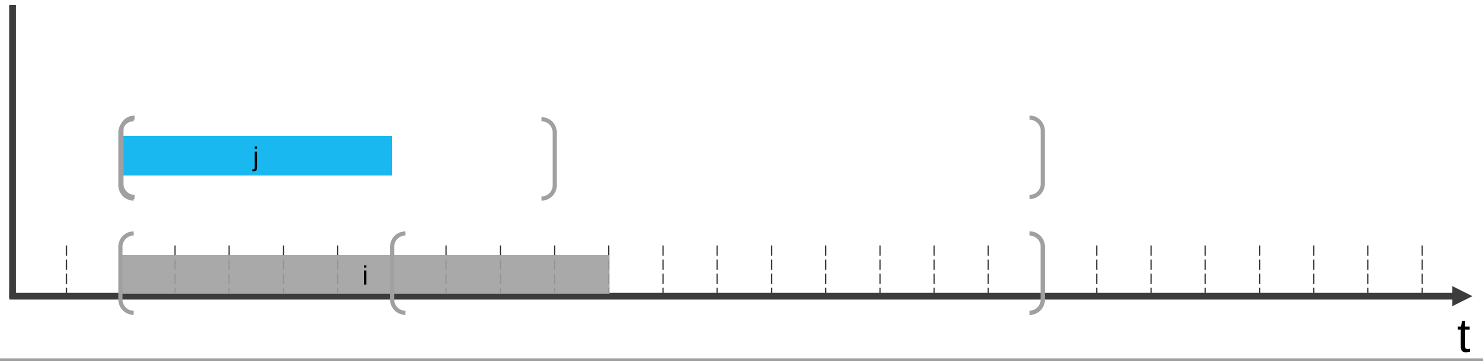
Requires good understanding of propagation techniques behind constraints

Scheduling with CP



Constraint on Interval Var



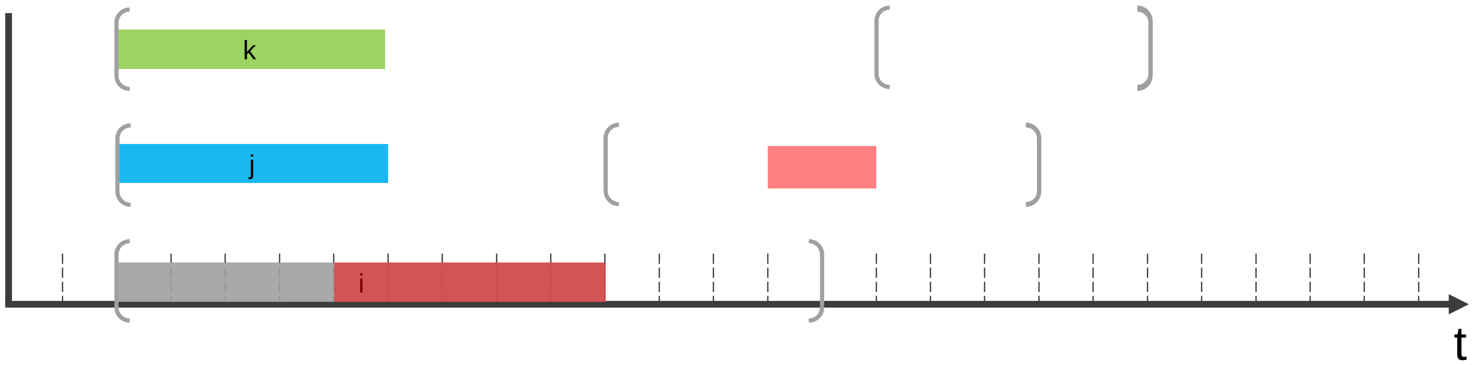


$e_j \leq s_i$

Unary Resource Constraint – Timetable propagation

Identify for each task its associated mandatory part

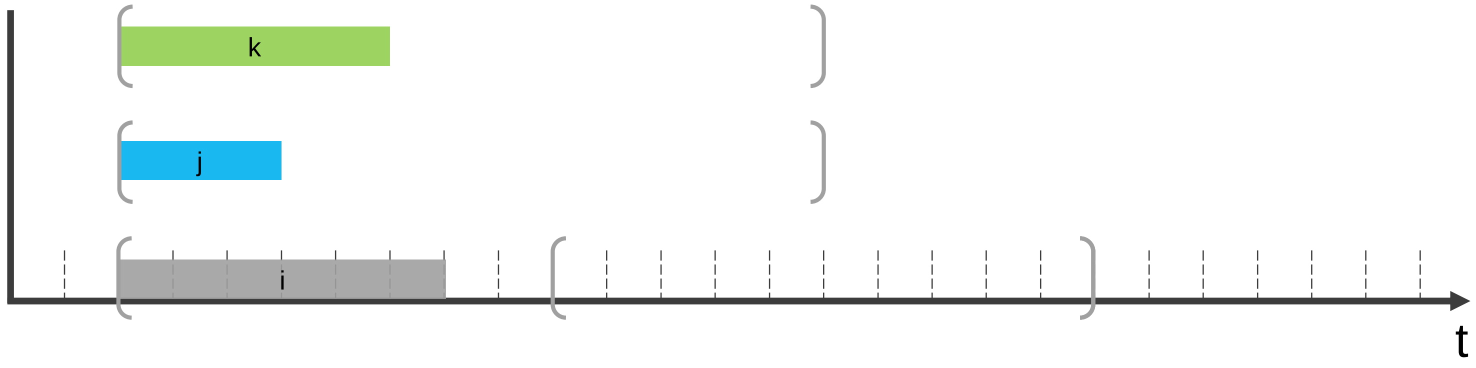
Filter the mandatory part from the other task domain



Unary Resource Constraint - Edge finding propagation

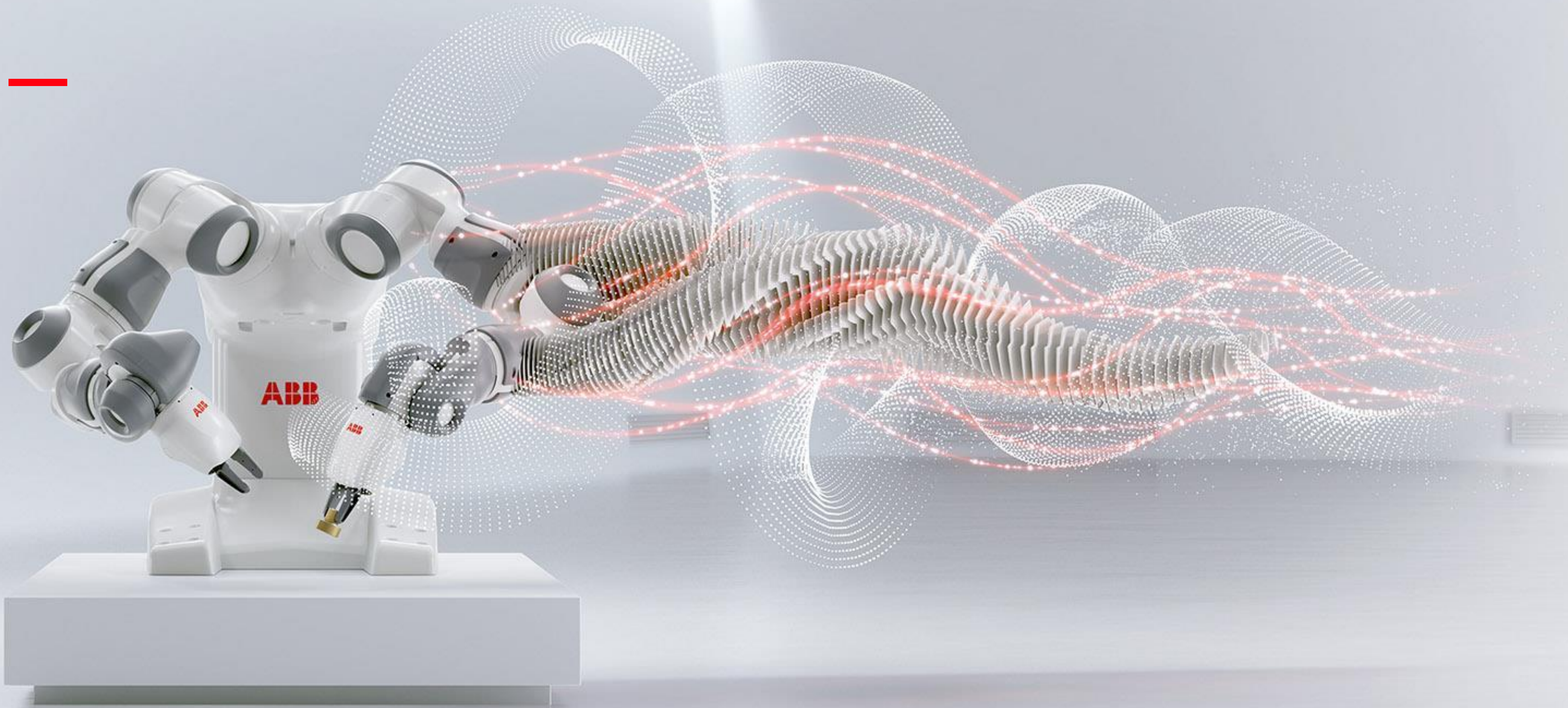
Identify a subset Ω of intervals

For each interval i ∉ Ω, verify if it can be executed before Ω



Just scratching the surface

Unary Resource
– Propagation Not-First / Not-Last
– Propagation of Detectable Precedences
– Transition Times (a.k.a. Setup Times)
– Intensity Functions (a.k.a. Calendar constraints)
Cumulative Resource
Reservoir Resource
State Resource



Conclusions

Conclusions

- Real challenge is understanding domain-specific knowledge and translate it into abstractions and mathematical formulations

- Getting access to data is key
  - Baseline for comparing optimized solution vs current solution
  - Understanding problem features and size

- Technology mastery is required to understand strengths and weaknesses of each technology and figure out which technology is suited for which problem