Hexaly Optimizer for Scheduling

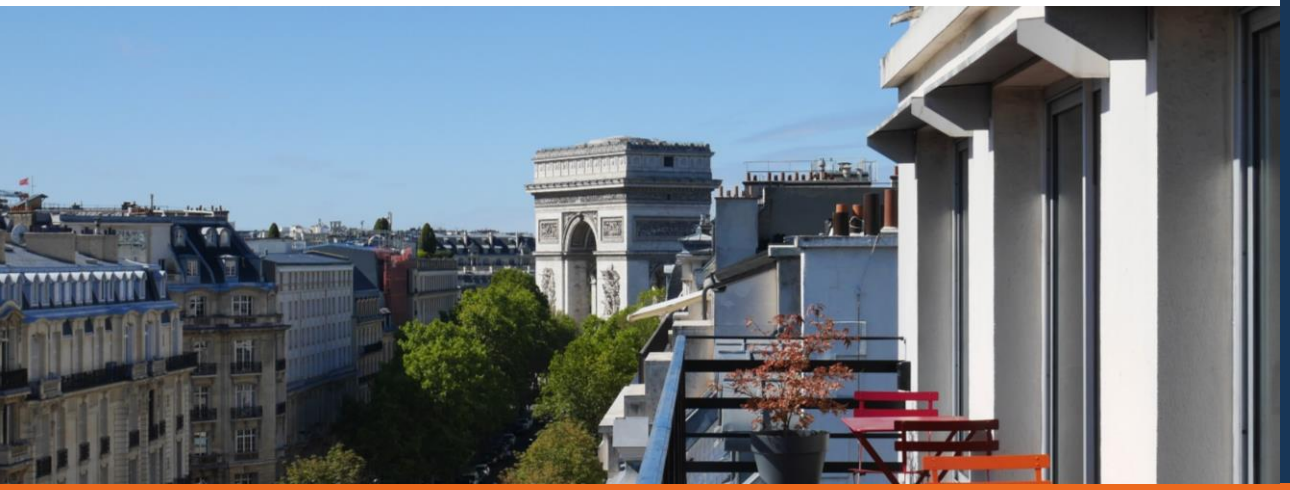
Philippe Laborie (plaborie@hexaly.com)

Scheduling seminar (schedulingseminar.com)
November 20, 2024

hexaly

Software company specialized in Mathematical Optimization, Operations Research, and Decision Science

- Powerful optimization solver & platform used by Amazon, FedEx, Starbucks, …
- Turnkey, custom optimization and planning applications for Air Liquide, Toyota, …



- Fast and scalable solver for Routing, Scheduling, Packing, and more
- 20 years of experience
- 200 clients, 400 applications, and 20,000 users in 25 countries
- Offices in Brooklyn, NY, and Paris, France

200 companies trust us

A complete platform for building and deploying mathematical optimization applications

Hexaly Optimizer

Fast and scalable mathematical optimization solver

Hexaly Modeler

Low-code modeling language for optimization

Hexaly Studio

No-code studio to build optimization apps in days

Hexaly Cloud

Mathematical Optimization as a Service

Hexaly Optimizer

Problem Specific approach vs. Mathematical solver

Business problem

Business solution

Algorithm

Mathematical modeling

Automated resolution

Hexaly Optimizer for Scheduling

- Industrial scheduling problems
- Basic Mathematics
- Hexaly Optimizer
    - Modeling language
    - Resolution techniques
    - Performance

Hexaly Optimizer for Scheduling

Industrial scheduling problems

Heterogeneous

- Several types of resources : disjunctive, cumulative, inventories, …
- Resource allocation decisions
- Different types of constraints: temporal, calendars, setup times, batches, …
- Diverse objectives : tardiness, allocation costs, resource setup costs, … (makespan is rare)

Large

- Some problems with up to n=1,000,000 activities, 1,000 resources
- Fine grain granularity of time, typically T=1,000,000 time points

Usually, MIP technology does not scale well on these problems

- Manne's formulation for disjunctive resources: $O(n^2)$ 1,000,000,000,000
- Discrete time formulations are in $O(n.T)$ 1,000,000,000,000

Hexaly Optimizer for Scheduling

Basic Mathematics

From elementary school

• Numbers:	1, 2, 3, …	u_i
• Comparison between numbers:	$2 \leq 4$	$u_i \leq RHS$
• Addition tables:	$2 + 3 = 5$	Σu_i
• Multiplication tables:	$2 * 3 = 6$	$\Sigma a_i u_i$

Hexaly Optimizer for Scheduling

Basic Mathematics

From elementary school

• Numbers:	1, 2, 3, …	u_i
• Comparison between numbers:	$2 \leq 4$	$u_i \leq RHS$
• Addition tables:	$2 + 3 = 5$	Σu_i
• Multiplication tables:	$2 * 3 = 6$	$\Sigma a_i u_i$

☛ MIP

Hexaly Optimizer for Scheduling

Basic Mathematics

From secondary school

- Sets

$A,\ u \in A,\ |A|,\ A \subseteq B,\ \text{partition}(A, \{B_i\}), \ldots$

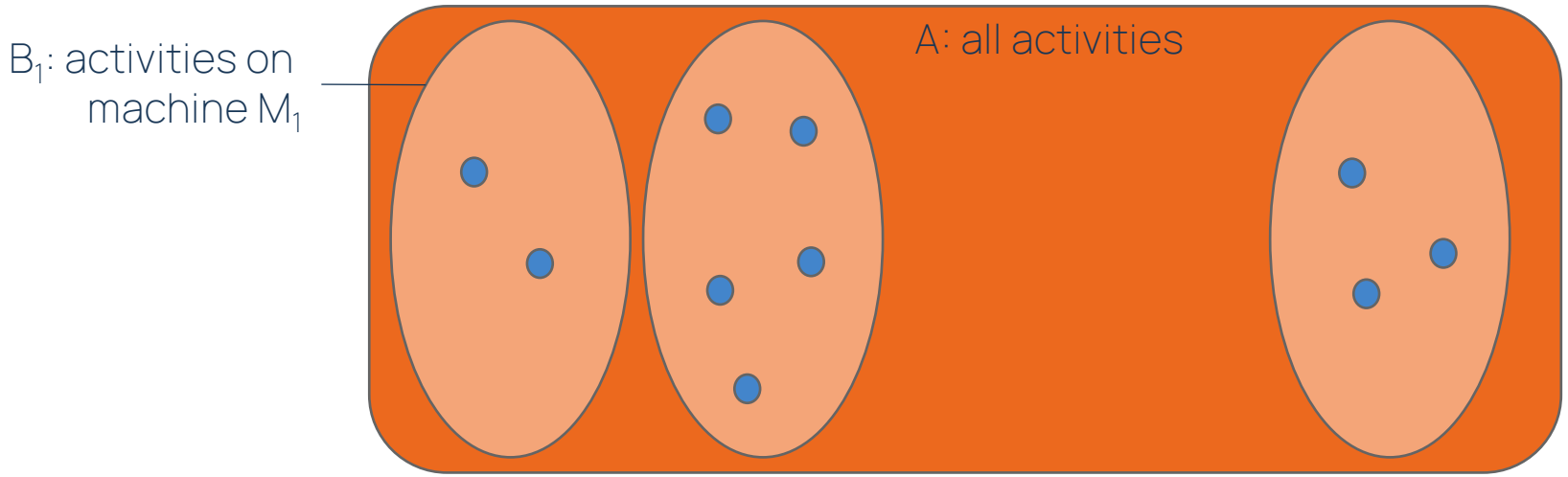
Hexaly Optimizer for Scheduling

Basic Mathematics

From secondary school

- Sets

$A,\ u \in A,\ |A|,\ A \subseteq B,\ \text{partition}(A, \{B_i\}), \ldots$



Hexaly Optimizer for Scheduling

Basic Mathematics

From secondary school

- Sets
- Permutations

$A,\ u \in A,\ |A|,\ A \subseteq B,\ \text{partition}(A, \{B_i\}), \ldots$

$S(A), \sigma \in S(A), \sigma(0), \sigma^{-1}(5), \ldots$

Hexaly Optimizer for Scheduling

Basic Mathematics

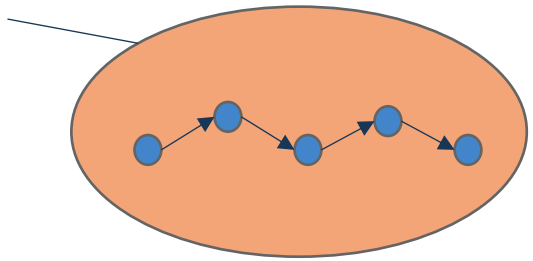
From secondary school

- Sets
- Permutations

$A,\ u \in A,\ |A|,\ A \subseteq B,\ \text{partition}(A, \{B_i\}), \ldots$

$S(A),\ \sigma \in S(A),\ \sigma(0),\ \sigma^{-1}(5), \ldots$

S: sequence of activities on a machine M



Hexaly Optimizer for Scheduling

Basic Mathematics

From secondary school

- Sets — $A,\ u \in A,\ |A|,\ A \subseteq B,\ \text{partition}(A, \{B_i\}), \ldots$
- Permutations — $S(A),\ \sigma \in S(A),\ \sigma(0),\ \sigma^{-1}(5), \ldots$
- Intervals — $x = [u,v),\ t \in x,\ \text{hull}(x,y,z), \ldots$

Hexaly Optimizer for Scheduling

Basic Mathematics

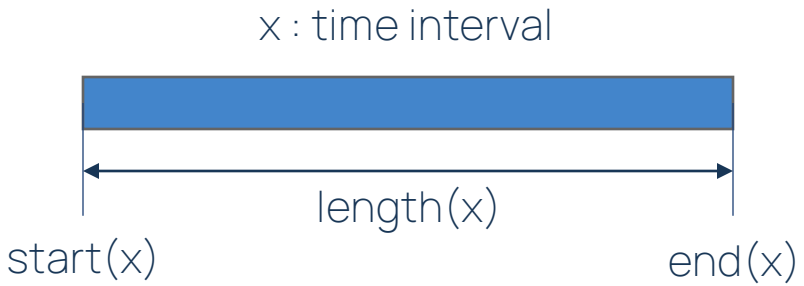
From secondary school

- Sets
- Permutations
- Intervals

$A,\ u \in A,\ |A|,\ A \subseteq B,\ \text{partition}(A, \{B_i\}), \ldots$

$S(A),\ \sigma \in S(A),\ \sigma(0),\ \sigma^{-1}(5), \ldots$

$x = [u,v),\ t \in x,\ \text{hull}(x,y,z), \ldots$



Hexaly Optimizer for Scheduling

Basic Mathematics

From secondary school

- Sets — $A,\ u \in A,\ |A|,\ A \subseteq B,\ \text{partition}(A, \{B_i\}), \ldots$
- Permutations — $S(A),\ \sigma \in S(A),\ \sigma(0),\ \sigma^{-1}(5), \ldots$
- Intervals — $x = [u,v),\ t \in x,\ \text{hull}(x,y,z), \ldots$
- Functions — $f(t)$

Hexaly Optimizer for Scheduling

Basic Mathematics

From secondary school

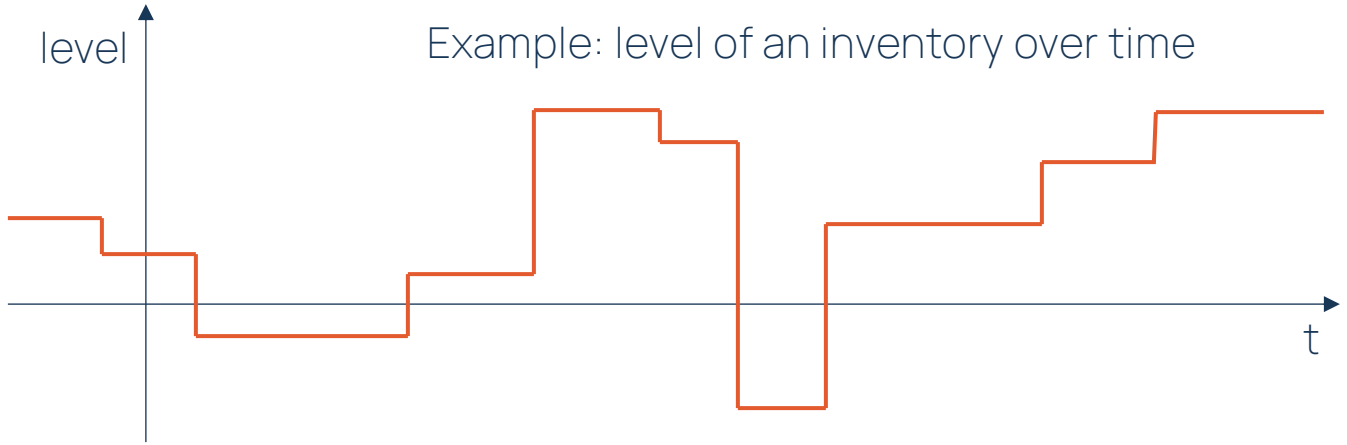
- Sets
- Permutations
- Intervals
- Functions

$A,\ u \in A,\ |A|,\ A \subseteq B,\ \text{partition}(A, \{B_i\}), \ldots$

$S(A),\ \sigma \in S(A),\ \sigma(0),\ \sigma^{-1}(5), \ldots$

$x = [u,v),\ t \in x,\ \text{hull}(x,y,z), \ldots$

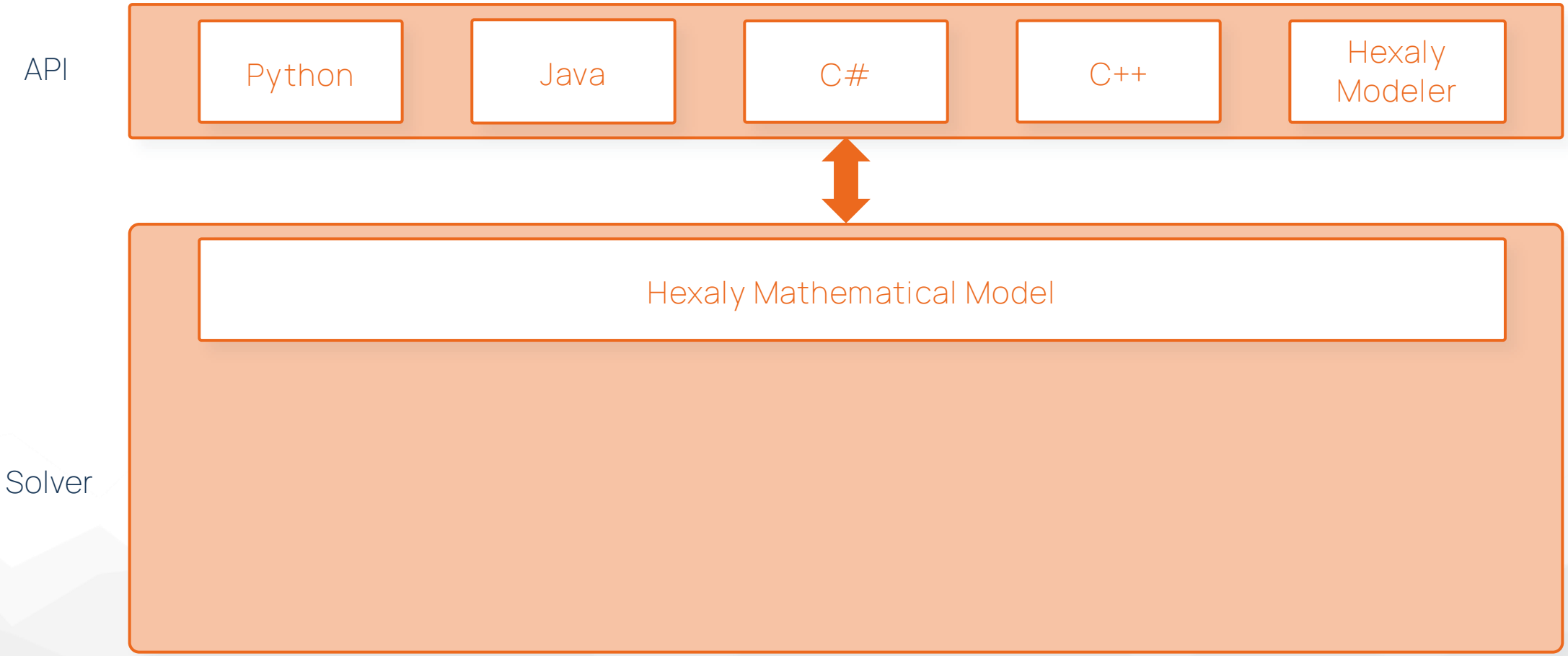
$f(t)$



Example: level of an inventory over time

Hexaly Optimizer for Scheduling

Hexaly Optimizer



Hexaly Optimizer for Scheduling

Hexaly Optimizer - Modeling

The Hexaly mathematical model is grounded on simple mathematical concepts :

- Numerical variables (Boolean, integer, floating point)
- Set variables
- List (permutation) variables
- Interval variables

Classical algebraical, logical and set theory operators (sum, min, max, and, or, union, hull …)

No need to introduce aggregated constructs like global constraints (even for scheduling)

Multi-objective (lexicographical)

Support for Blackbox (external) functions

Support for initial solutions (warm start)

Hexaly Optimizer for Scheduling

Hexaly Optimizer - Modeling

Boolean variable

```
x <- bool();
```

Possible value: `x=true`

Integer variable

```
x <- int(-10, 10);
```

Possible value: `x=-2`

Floating point variable

```
PI = 3.14159265359;
x <- float(-PI, PI);
```

Possible value: `x=1.0471975512`

Hexaly Optimizer for Scheduling

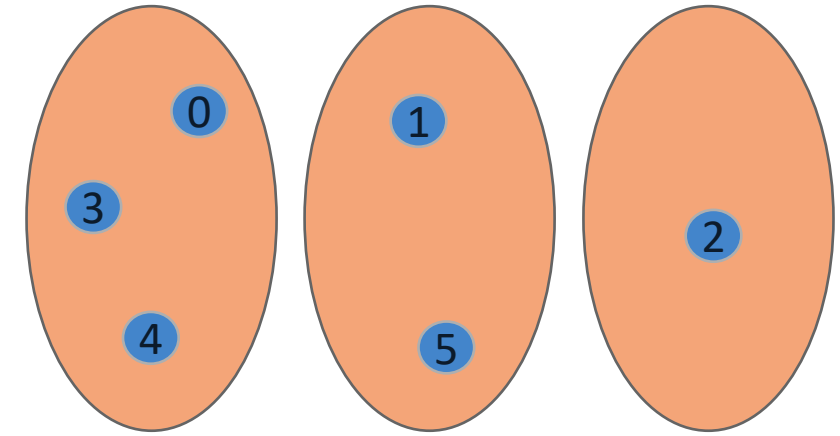
Hexaly Optimizer - Modeling

Set variable

```
A <- set(n);
```

A : a subset of {0, 1, …, n-1}

- Uniqueness of items
- Variable size



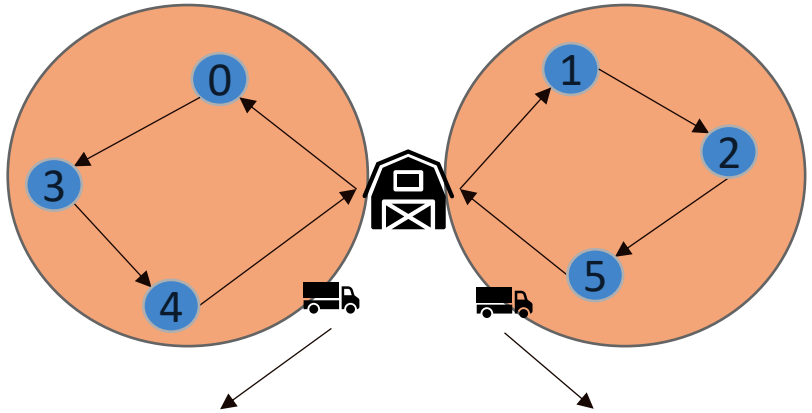
Set = {0, 3, 4} Set = {1, 5} Set = {2}

List variable

```
S <- list(n);
```

S : permutation of a subset of {0, 1, …, n-1}

- Uniqueness of items
- Variable size
- Ordering matters



List = [0, 3, 4] List = [1, 2, 5]

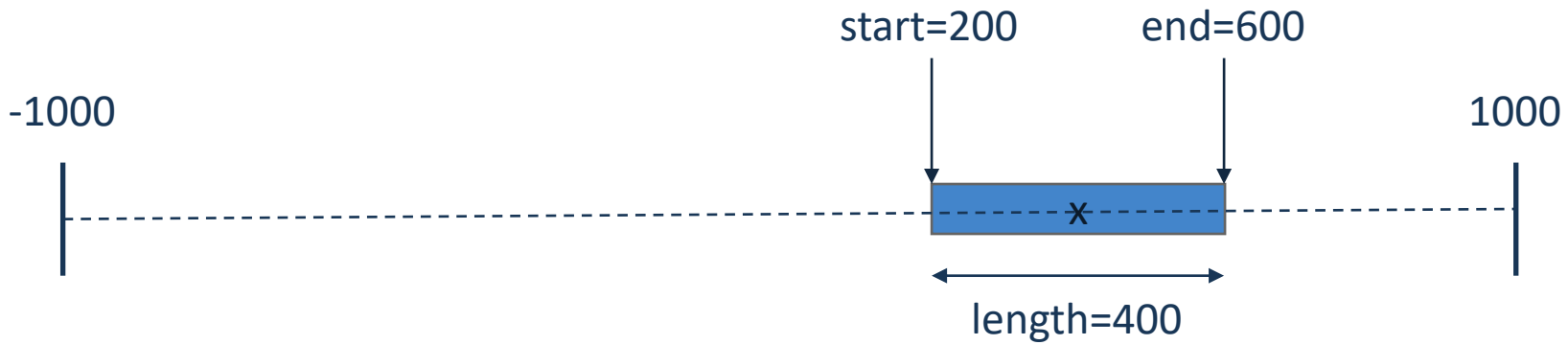
Hexaly Optimizer for Scheduling

Hexaly Optimizer - Modeling

Interval variable

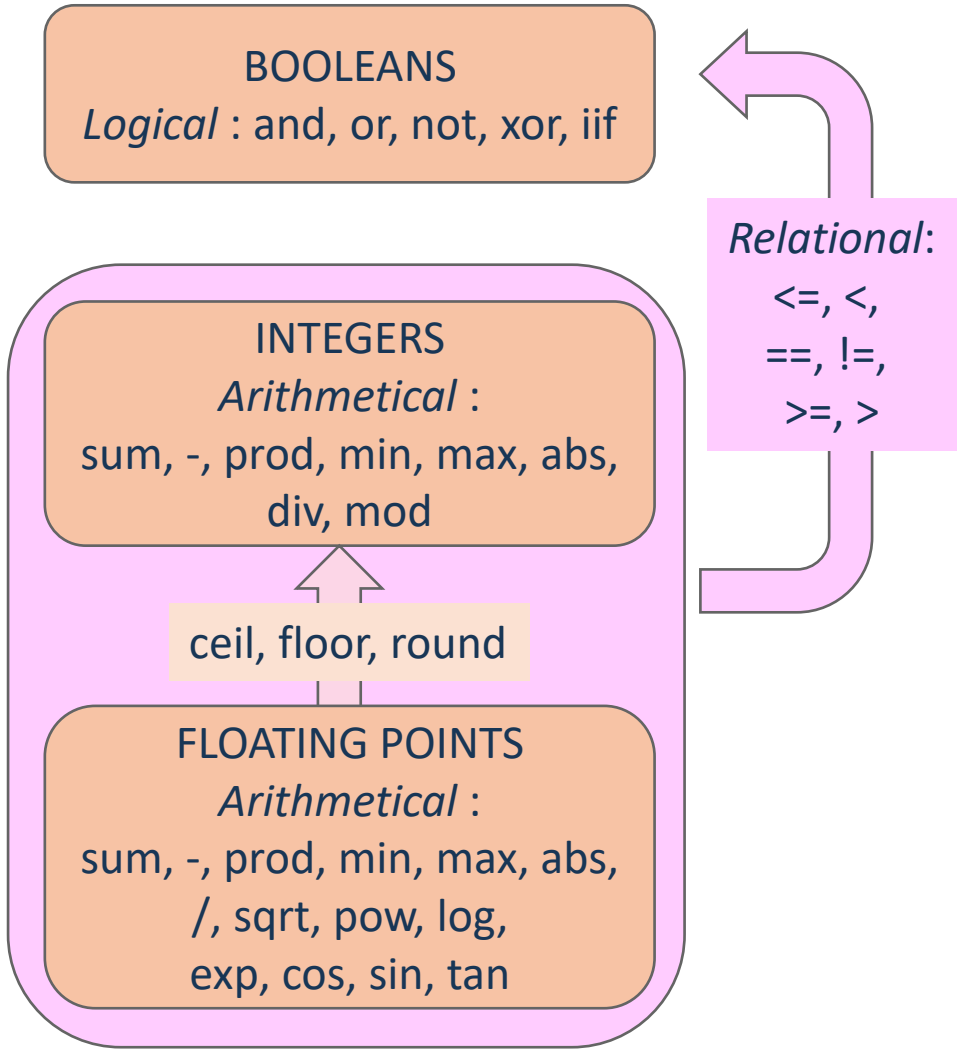
```
x <- interval(-1000, 1000);
```

Possible value: `x=[200,600)`



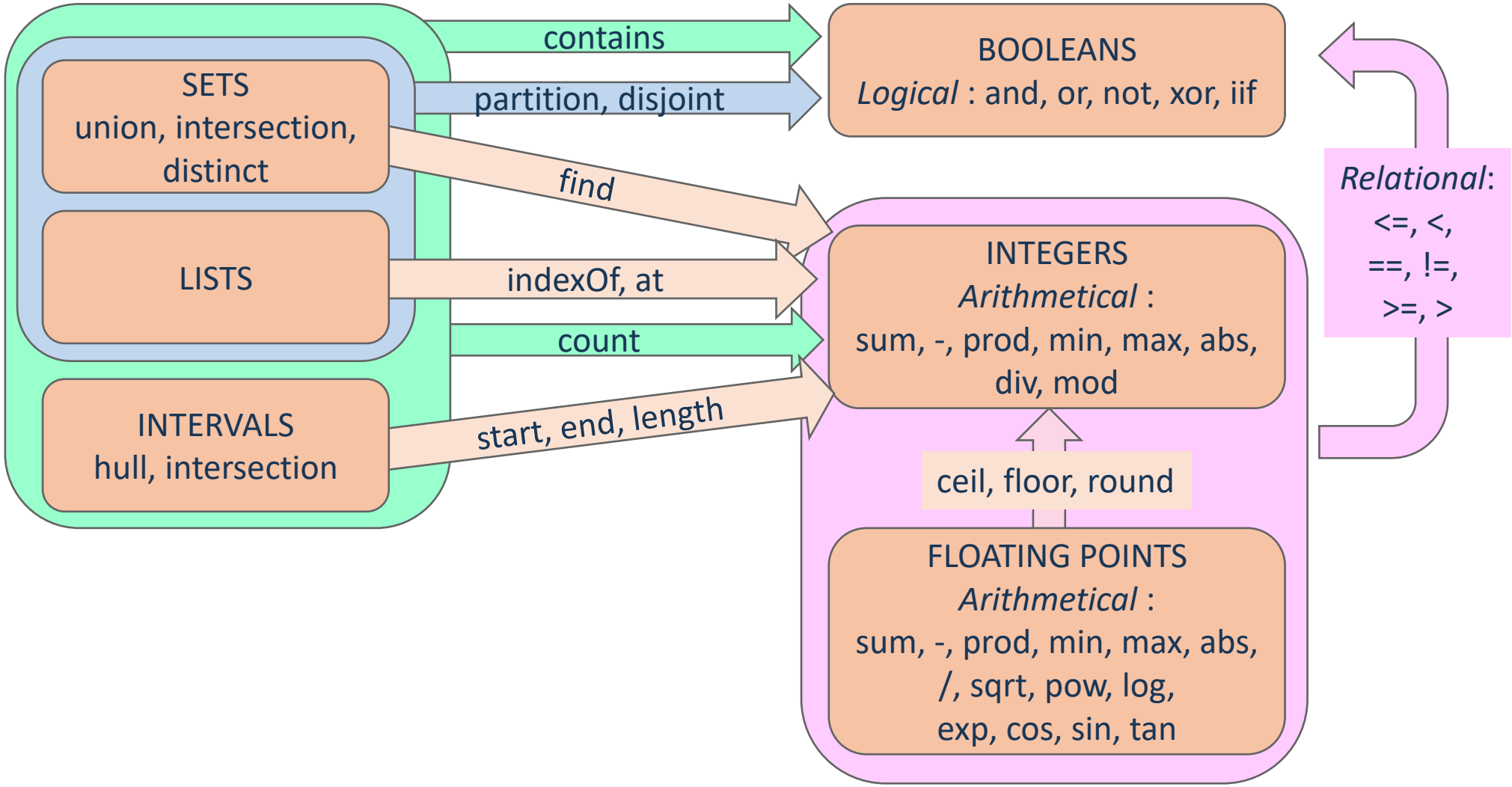
Hexaly Optimizer for Scheduling

Hexaly Optimizer - Modeling



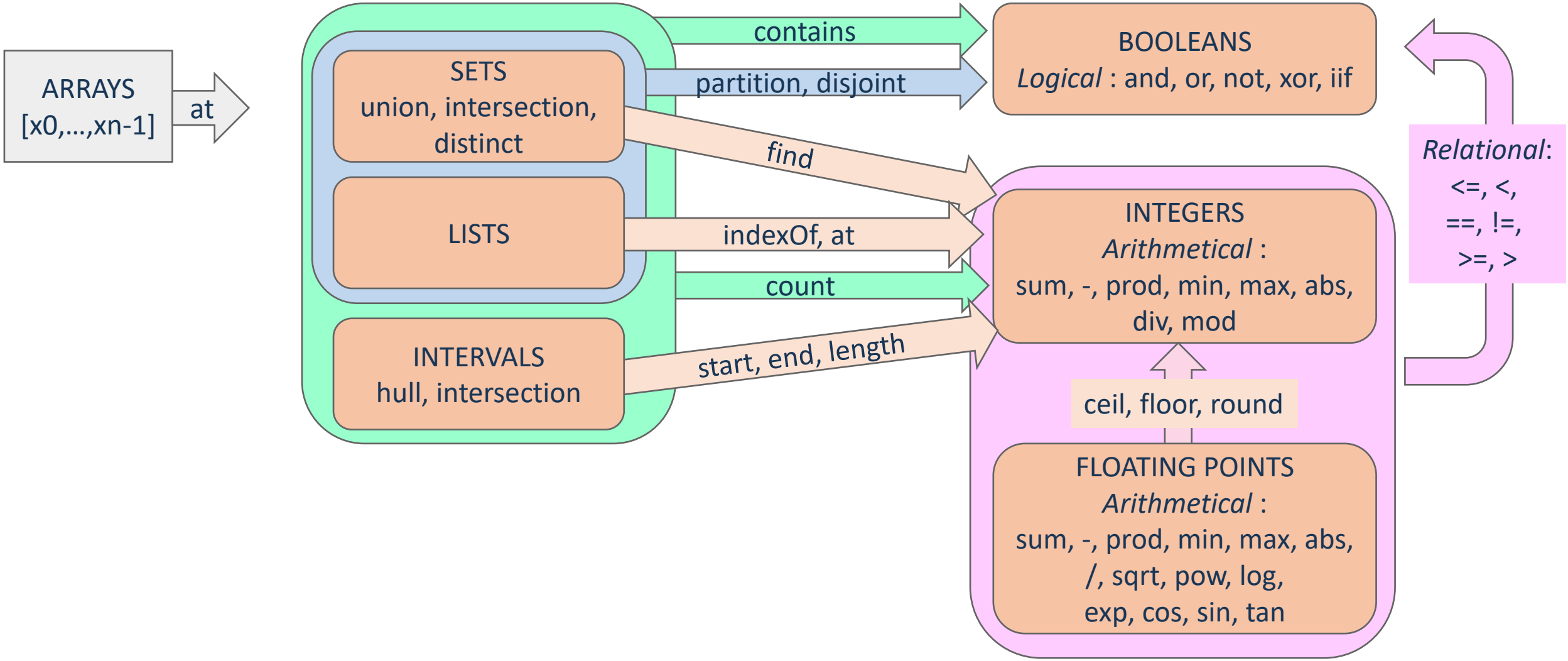
Hexaly Optimizer for Scheduling

Hexaly Optimizer - Modeling



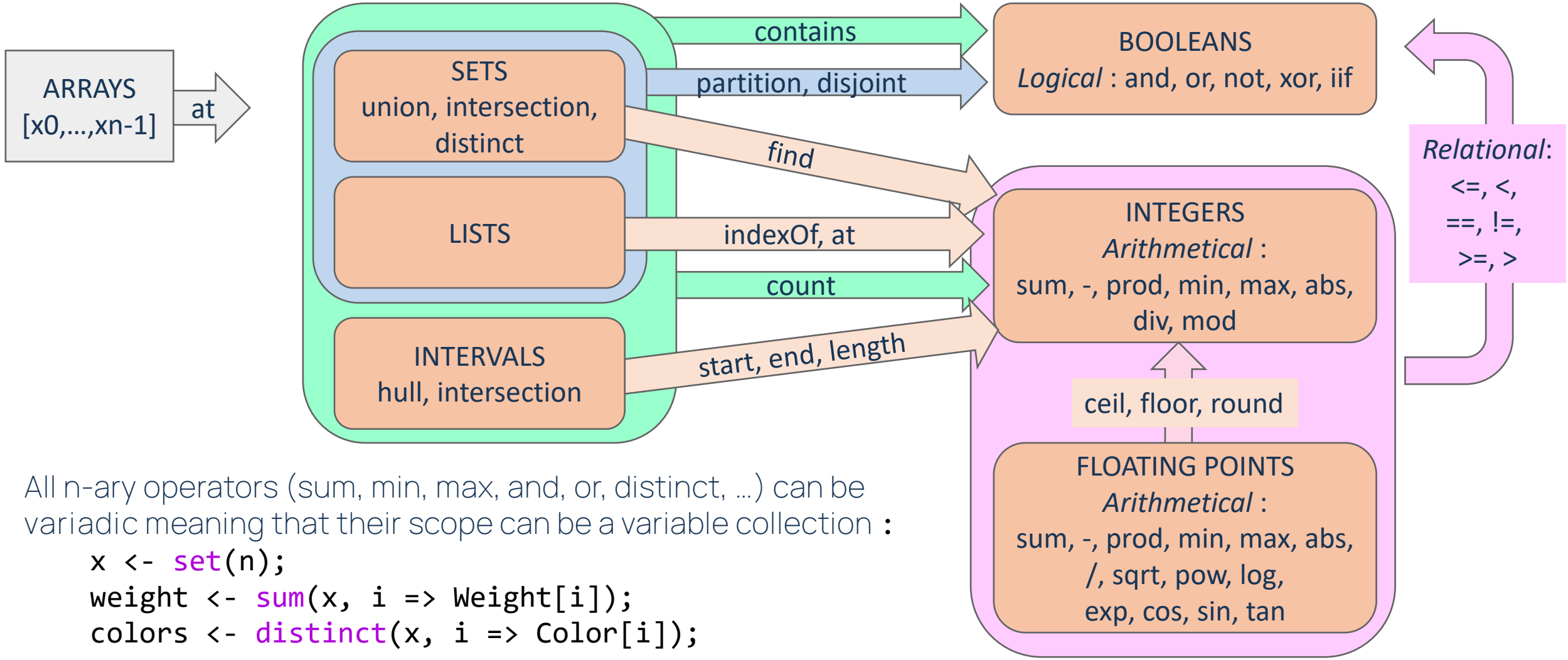
Hexaly Optimizer for Scheduling

Hexaly Optimizer - Modeling



Hexaly Optimizer for Scheduling

Hexaly Optimizer - Modeling



All n-ary operators (sum, min, max, and, or, distinct, ...) can be variadic meaning that their scope can be a variable collection :

```
x <- set(n);
weight <- sum(x, i => Weight[i]);
colors <- distinct(x, i => Color[i]);
```

Hexaly Optimizer for Scheduling

Hexaly Optimizer - Modeling

Example: precedence constraint between two activities

```
x <- interval(0, 1000);
y <- interval(0, 1000);
constraint x < y;
```

Hexaly Optimizer for Scheduling

Hexaly Optimizer - Modeling

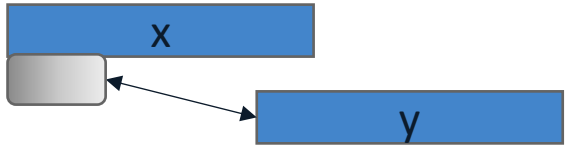
Example: precedence constraint between two activities

```
x <- interval(0, 1000);
y <- interval(0, 1000);
constraint x < y;
```

Example: precedence constraints between activities endpoints

```
constraint end(x) + Delay <= start(y);
```


```
constraint start(x) + delayExpr == start(y);
```



Hexaly Optimizer for Scheduling

Hexaly Optimizer - Modeling

Example: activity with fixed duration

```
x <- interval(0, 1000);
constraint length(x) == Duration;
```

Hexaly Optimizer for Scheduling

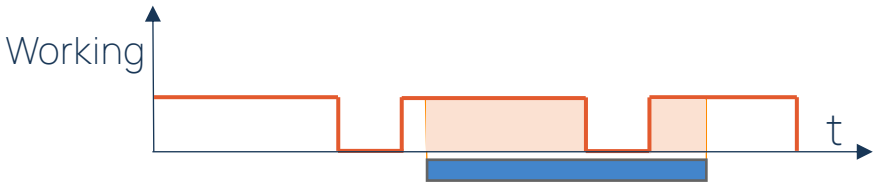
Hexaly Optimizer - Modeling

Example: activity with fixed duration

```
x <- interval(0, 1000);
constraint length(x) == Duration;
```

Example: activity with intensity function

```
x <- interval(0, 1000);
constraint sum(x, t => Working[t]) == WorkDuration;
```



Hexaly Optimizer does not “unroll" the expression over the entire interval of x

Hexaly Optimizer for Scheduling

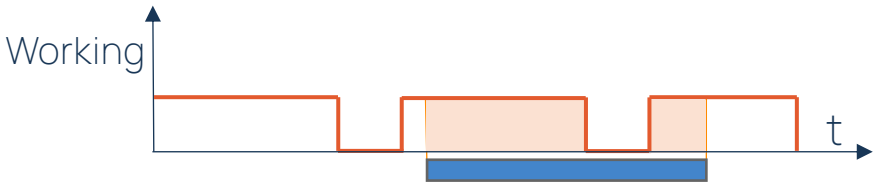
Hexaly Optimizer - Modeling

Example: activity with fixed duration

```
x <- interval(0, 1000);
constraint length(x) == Duration;
```

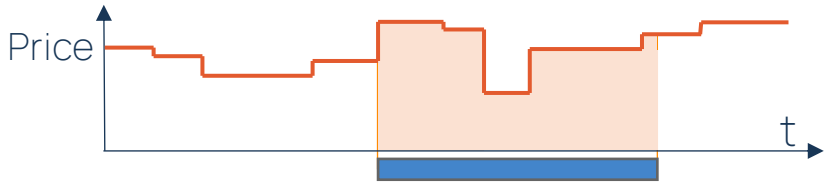
Example: activity with intensity function

```
x <- interval(0, 1000);
constraint sum(x, t => Working[t]) == WorkDuration;
```



Example: activity with time-dependent price

```
x <- interval(0, 1000);
cost <- sum(x, t => Price[t])
```



Hexaly Optimizer does not “unroll" the expression over the entire interval of x

Hexaly Optimizer for Scheduling

Hexaly Optimizer - Modeling

Example: makespan

```
x[i in 0…N] <- interval(0, 1000);
minimize max[i in 0…N] end(x[i]);
```

Example: weighted sum of tardiness cost

```
minimize sum[i in 0…N] ( Weight[i] * max(0, end(x[i])- DueDate[i]) );
```

Example: Net Present Value

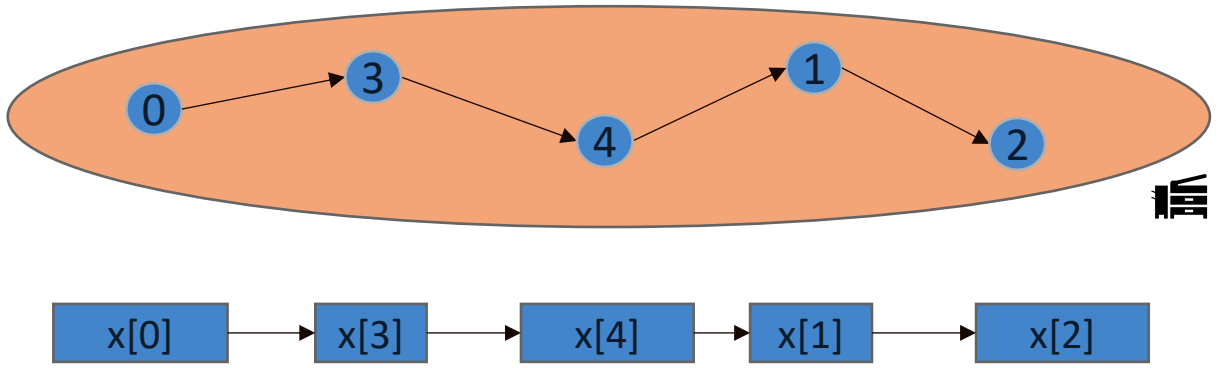
```
maximize sum[i in 0…N] ( NetCashFlow[i] / pow(1+DiscountRate, end(x[i])) );
```

Hexaly Optimizer for Scheduling

Hexaly Optimizer - Modeling

Example: disjunctive resource (machine)

```
seq <- list(n);
x[i in 0…n] <- interval(0, 1000);
constraint count(seq) == n;
constraint and(1…n, i => x[seq[i-1]] < x[seq[i]]);
```



seq = [0, 3, 4, 1, 2]

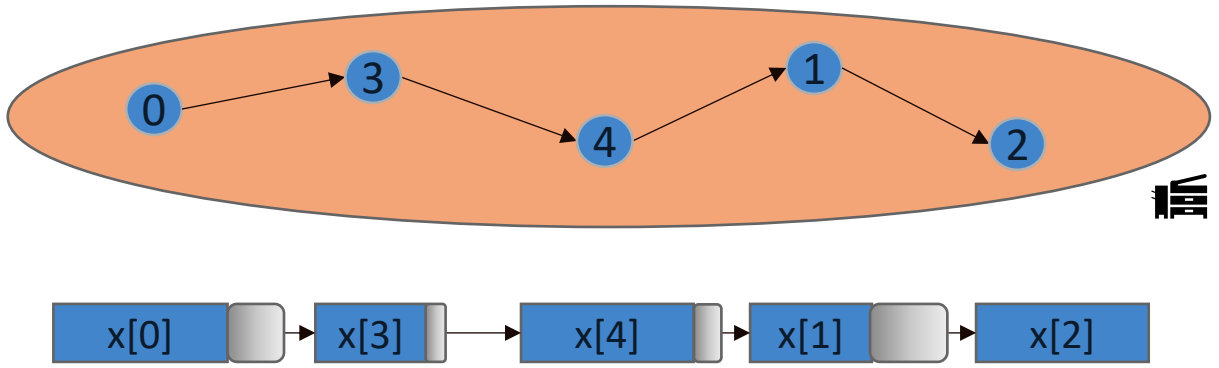
The formulation of the constraint uses a variadic "and" (equivalent to a "forall" expression): size is O(n)

Hexaly Optimizer for Scheduling

Hexaly Optimizer - Modeling

Example: disjunctive resource (machine) with sequence-dependent setup times

```
seq <- list(n);
x[i in 0…n] <- interval(0, 1000);
constraint count(seq) == n;
constraint and(1…n, i => end(x[seq[i-1]]) + SetupTime[seq[i-1]][seq[i]] <= start(x[seq[i]]));
```



seq = [0, 3, 4, 1, 2]

The formulation of the constraint uses a variadic "and" (equivalent to a "forall" expression): size is O(n)

Hexaly Optimizer for Scheduling

Hexaly Optimizer - Modeling

Example: disjunctive resource (machine) with sequence-dependent setup times

```
seq <- list(n);
x[i in 0…n] <- interval(0, 1000);
constraint count(seq) == n;
constraint and(1…n, i => end(x[seq[i-1]]) + SetupTime[seq[i-1]][seq[i]] <= start(x[seq[i]]));
```

Example: setup costs

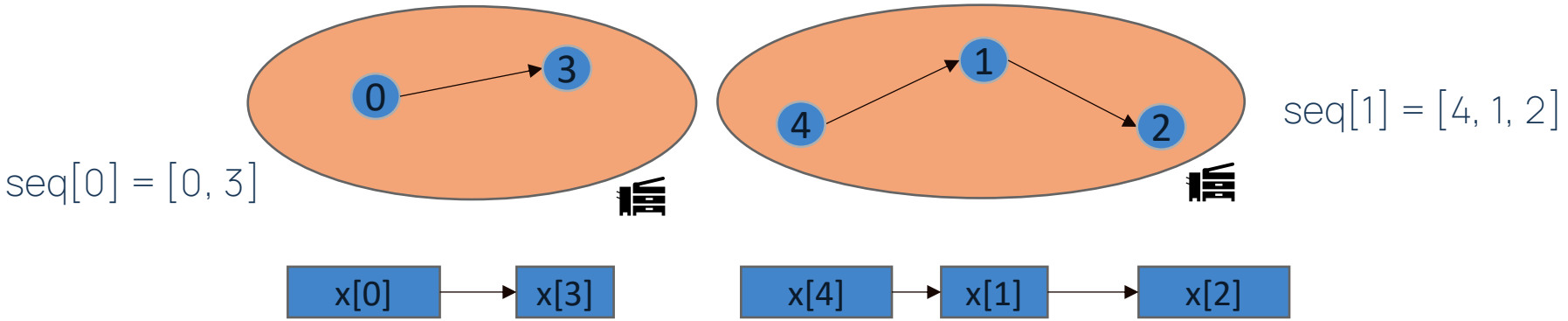
```
minimize sum(1…n, i => SetupCost[seq[i-1]][seq[i]]);
```

Hexaly Optimizer for Scheduling

Hexaly Optimizer - Modeling

Example: disjunctive resource (machine) allocation

```
seq[j in 0…m] <- list(n);
x[i in 0…n] <- interval(0, 1000);
constraint partition(seq);
for [j in 0…m] constraint and(1…count(seq[j]), i => x[seq[j][i-1]] < x[seq[j][i]]);
```



The formulation of the constraint uses a variadic "and" (equivalent to a "forall" expression): size is O(n)

Hexaly Optimizer for Scheduling

Hexaly Optimizer - Modeling

Example: disjunctive resource (machine) allocation

```
seq[j in 0…m] <- list(n);
x[i in 0…n] <- interval(0, 1000);
constraint partition(seq);
for [j in 0…m] constraint and(1…count(seq[j]), i => x[seq[j][i-1]] < x[seq[j][i]]);
```

Example: resource-dependent features

```
constraint  contains(seq[0], 3); // Compulsory machine
constraint !contains(seq[0], 2); // Incompatible machine
constraint  contains(seq[0], 1) <= !contains(seq[1], 4) ; // Dependency constraints
```

Hexaly Optimizer for Scheduling

Hexaly Optimizer - Modeling

Example: disjunctive resource (machine) allocation

```
seq[j in 0…m] <- list(n);
x[i in 0…n] <- interval(0, 1000);
constraint partition(seq);
for [j in 0…m] constraint and(1…count(seq[j]), i => x[seq[j][i-1]] < x[seq[j][i]]);
```

Example: resource-dependent features

```
mach[i in 0…m] <- find(seq, i); // Machine of task i (element of the partition i belongs to)
for [i in 0…n] {
  constraint length(x[i]) == Duration[i][mach[i]];        // Machine-dependent duration
  constraint start(x[i]) >= AvailableTime[mach[i]];       // Machine start time
}
```

Hexaly Optimizer for Scheduling

Hexaly Optimizer - Modeling

Example: disjunctive resource (machine) allocation

```
seq[j in 0…m] <- list(n);
x[i in 0…n] <- interval(0, 1000);
constraint partition(seq);
for [j in 0…m] constraint and(1…count(seq[j]), i => x[seq[j][i-1]] < x[seq[j][i]]);
```

Example: resource-dependent features

```
mach[i in 0…n] <- find(seq, i); // Machine of task i (element of the partition i belongs to)
constraint end(x[i]) + TravelTime[mach[i]][mach[j]] <= start(x[j]); // Change-over time
```

Hexaly Optimizer for Scheduling

Hexaly Optimizer - Modeling

Example: cumulative resource

At any time t, the sum of the weights of the tasks running on the resource must be less than the capacity of the resource

```
x[i in 0…n] <- interval(0, 1000);
```

Hexaly Optimizer for Scheduling

Hexaly Optimizer - Modeling

Example: cumulative resource

At any time *t*, the sum of the weights of the tasks running on the resource must be less than the capacity of the resource

```
x[i in 0…n] <- interval(0, 1000);
constraint and(0...1000,
               t =>
               …);
```

Hexaly Optimizer for Scheduling

Hexaly Optimizer - Modeling

Example: cumulative resource

At any time *t*, the sum of the weights of the tasks running on the resource must be less than the capacity of the resource

```
x[i in 0…n] <- interval(0, 1000);
constraint and(0...1000,
               t =>
               sum[i in 0...n](Weight[i] …
```

Hexaly Optimizer for Scheduling

Hexaly Optimizer - Modeling

Example: cumulative resource

At any time *t*, the sum of the weights of the tasks running on the resource must be less than the capacity of the resource

```
x[i in 0…n] <- interval(0, 1000);
constraint and(0...1000,
               t =>
               sum[i in 0...n](Weight[i] * contains(x[i], t)) …
```

Hexaly Optimizer for Scheduling

Hexaly Optimizer - Modeling

Example: cumulative resource

At any time *t*, the sum of the weights of the tasks running on the resource must be less than the capacity of the resource

```
x[i in 0…n] <- interval(0, 1000);
constraint and(0...1000,
               t =>
               sum[i in 0...n](Weight[i] * contains(x[i], t)) <= Capacity);
```

The constraint is written in intention: Hexaly Optimizer does not unroll the loop on *t*

Hexaly Optimizer for Scheduling

Hexaly Optimizer - Modeling

Example: cumulative resource

The temporal scope can be a variable range or an interval variable

```
x[i in 0…n] <- interval(0, 1000);
constraint and(0...makespan,
               t =>
               sum[i in 0...n](Weight[i] * contains(x[i], t)) <= Capacity);
```

The constraint is written in intention: Hexaly Optimizer does not unroll the loop on *t*

Hexaly Optimizer for Scheduling

Hexaly Optimizer - Modeling

Example: cumulative resource

The capacity may be time-dependent

```
x[i in 0…n] <- interval(0, 1000);
constraint and(0...makespan,
               t =>
               sum[i in 0...n](Weight[i] * contains(x[i], t)) <= Capacity[t]);
```

The constraint is written in intention: Hexaly Optimizer does not unroll the loop on *t*

Mathematical solvers for scheduling problems

Hexaly Optimizer - Modeling

Example: cumulative resource

The task scope may be a (variable) set

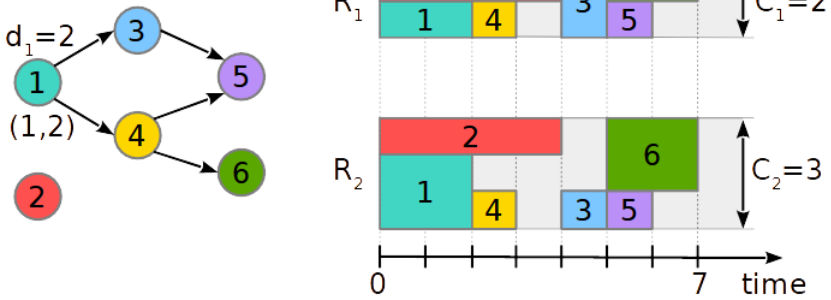
```
x[i in 0…n] <- interval(0, n);
constraint and(0...makespan,
               t =>
               sum(taskset, i => (Weight[i] * contains(x[i], t)) <= Capacity[t]);
```

The constraint is written in intention: Hexaly Optimizer does not unroll the loop on *t*

Hexaly Optimizer for Scheduling

Hexaly Optimizer - Modeling

Wrap-up: Complete HXM model for the Resource Constrained Project Scheduling Problem (RCPSP)

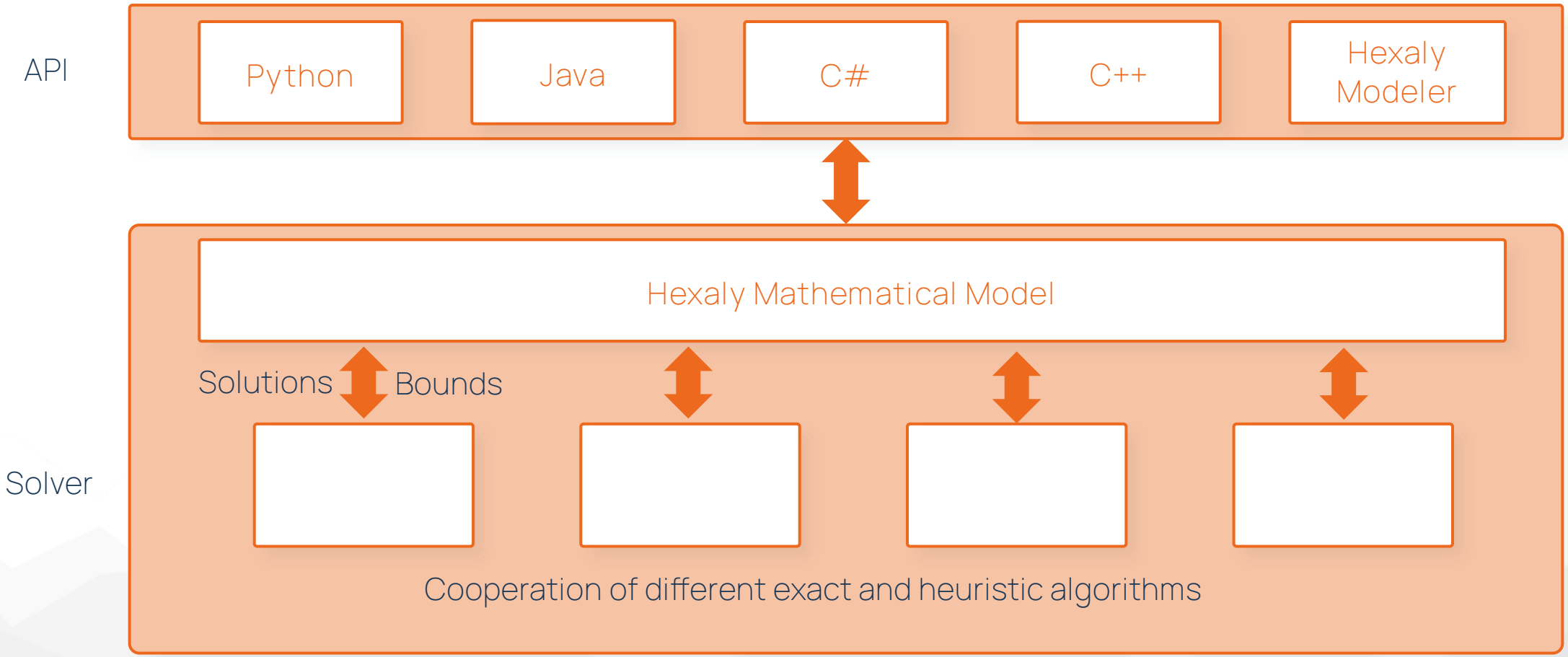


H:	Schedule horizon
DUR[i]:	Duration of task i
NSUCC[i]:	Number of successors of task i
SUCC[i][j]:	j^{th} successor of task i
CAP[r]:	Capacity of resource r
USE[i][r]:	Quantity of resource r used by task i
task[i]:	Interval representing task i

```
function model() {
  task[i in 0...n] <- interval(0, H);
  for [i in 0...n] constraint length(task[i]) == DUR[i];
  for [i in 0...n][j in 0...NSUCC[i]] constraint task[i] < task[SUCC[i][j]];
  makespan <- max[i in 0...n](end(task[i]));
  for [r in 0...m]
    constraint and(0...makespan, t => sum[i in 0...n](USE[i][r] * contains(task[i],t)) <= CAP[r]);
  minimize makespan;
}
```

Hexaly Optimizer for Scheduling

Hexaly Optimizer



Hexaly Optimizer

Hybridizes a myriad of exact and heuristic methods under the hood

Simplex	Interior-Point	Augmented Lagrangian
Black-box	Derivative-free methods	Surrogate modeling
Spatial Branch-and-Bound	Cutting planes	Interval methods
Constraint propagation	Clause learning	Dantzig-Wolfe reformulations
Primal heuristics	Large neighborhood search	Branch-Cut-Price
Local search	Exact scheduling algorithms on relaxations	
Multi-objective optimization	Statistical learning techniques for autotuning	

Performance

Hexaly 13.0 v.s. Gurobi 11.0 on the Flexible Job Shop Scheduling Problem (FJSP)

Flexible Job Shop (330 classical instances)

	Hexaly 13.0	Gurobi 11.0
1 – 100	0.3	1.1
101 – 200	0.8	6.4
201 – 300	0.9	26.1
301 – 400	0.9	67.1
401 – 500	0.2	91.6

Average gaps (%) obtained in 1 minute of running time. All gaps over 100% are reported as 100%.

Performance

Hexaly 13.0 v.s OR-Tools 9.10 (CPSAT) v.s. Gurobi 11.0 on the Resource Constrained Scheduling Problem (RCPSP) with 300 tasks (RG300)

RCPSP – RG300

	Hexaly	OR-Tools	Gurobi
Gap < 10%	100%	100%	0%
Gap < 5%	91%	66%	0%
Gap < 2.5%	62%	48%	0%
Gap < 1%	41%	29%	0%

Percentage of instances below a certain gap to the best known solution within 1 minute of running time.

Performance (Primal solutions)

Deviation to the best-known solutions in 60s with Hexaly 13.0

Job Shop
1.7% deviation,
133 instances, up to 2000 operations

Flexible Job Shop (FJSP)
0.5% deviation,
330 instances, up to 500 operations

FJSP with transition time
0.96% deviation,
instances up to 500 tasks

FJSP with calendars
2.25% deviation,
instances up to 500 jobs

Open Shop
0.00% deviation,
60 instances, up to 400 tasks

RCPSP (cumulative)
1.42% deviation,
instances up to 300 tasks

Performance (Scaling on large problems)

Large instances of job-shop problem (Da Col and Teppan 2022) 1,000 jobs and 1,000 machines (1,000,000 operations).

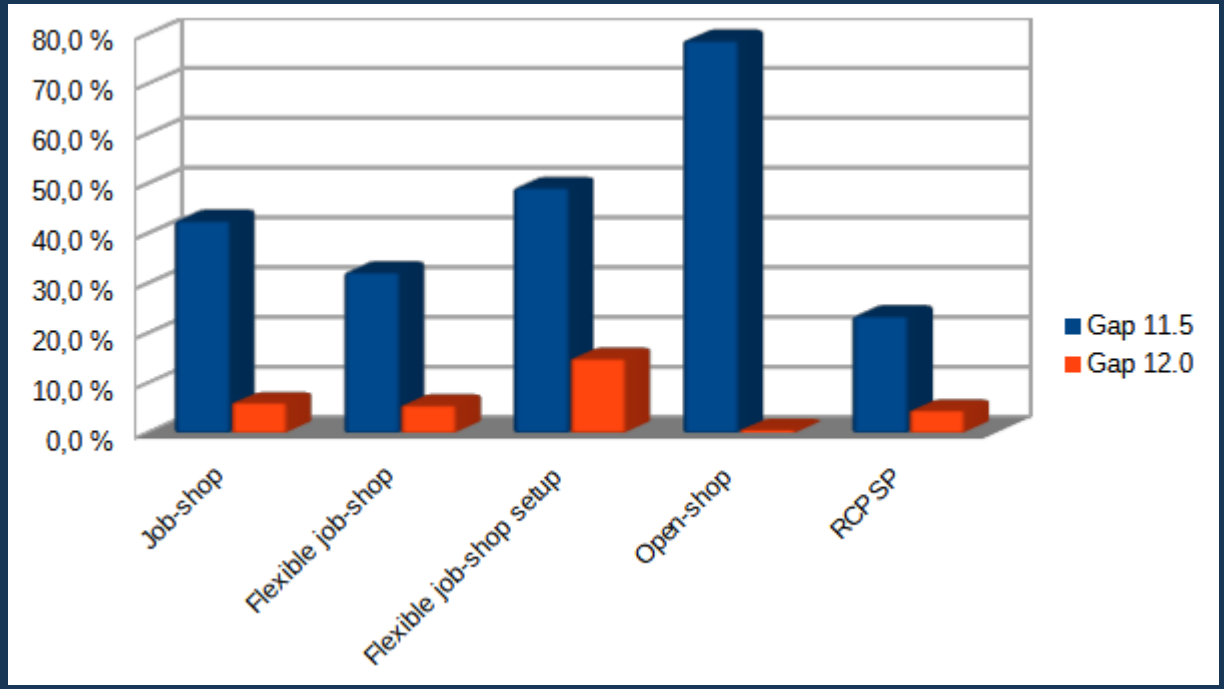
Job-shop 1000x1000

	Hexaly (10m)	Hexaly (10m) vs CP Optimizer (6h)	CP Optimizer (6h)	OR-Tools (6h)	Gurobi (6h)	Cplex (6h)
tai_j1000_m1000_1	877042	-8.6%	959945	-	-	-
tai_j1000_m1000_2	877064	-8.8%	961924	-	-	-
tai_j1000_m1000_3	878786	-8.2%	957455	-	-	-
tai_j1000_m1000_4	876565	-8.8%	961039	-	-	-
tai_j1000_m1000_5	877610	-9.0%	964267	-	-	-
tai_j1000_m1000_6	876067	-8.5%	957143	-	-	-
tai_j1000_m1000_7	876502	-8.4%	956978	-	-	-
tai_j1000_m1000_8	876503	-8.3%	955338	-	-	-
tai_j1000_m1000_9	876443	-8.4%	956860	-	-	-
tai_j1000_m1000_10	875312	-8.4%	955693	-	-	-

Results of Hexaly, CP Optimizer, OR-Tools, Gurobi, and Cplex on very large-scale instances of the Job Shop Scheduling Problem (JSSP).

Performance (Lower bounds)

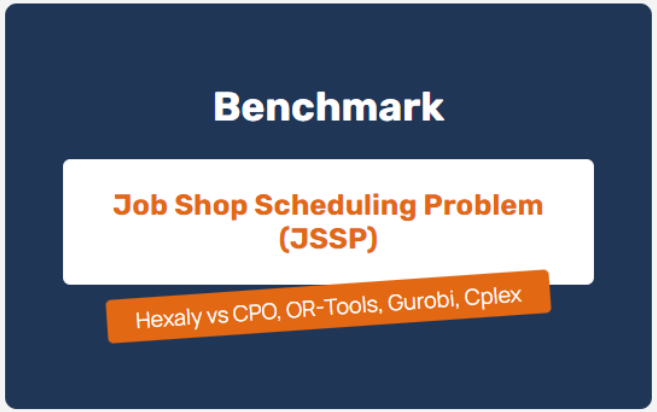
Gap between Hexaly lower bound and the best-known solution in 60s



RCPSP

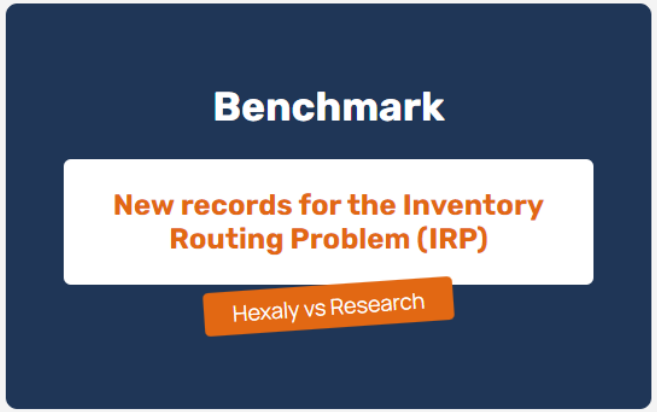
Hexaly improves the best known lower bounds on 34% of the instances from the RG300 benchmark (165 instances over 480)

Check our benchmarks: hexaly.com/benchmarks



Hexaly vs CP Optimizer, OR-Tools, Gurobi, Cplex on large-scale instances of the Job Shop Scheduling Problem (JSSP)

Read more →

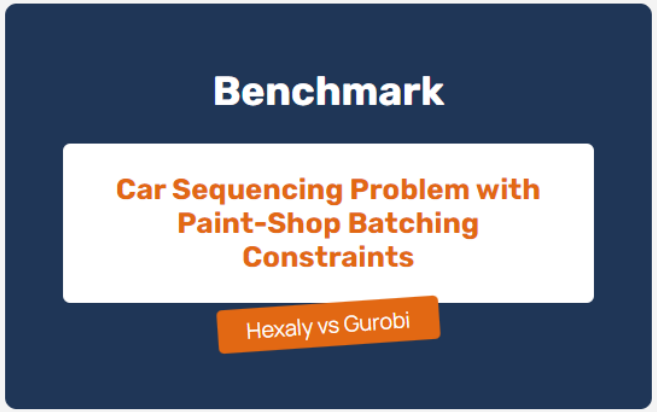


Hexaly establishes new records for the Inventory Routing Problem (IRP)

Read more →

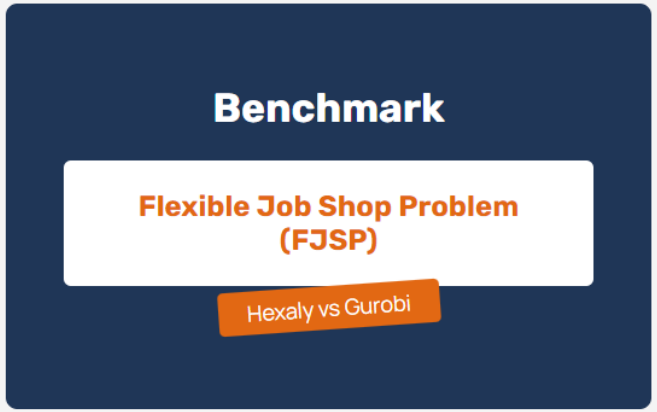
Hexaly breaks records for the Resource-Constrained Project Scheduling Problem (RCPSP)

Read more →



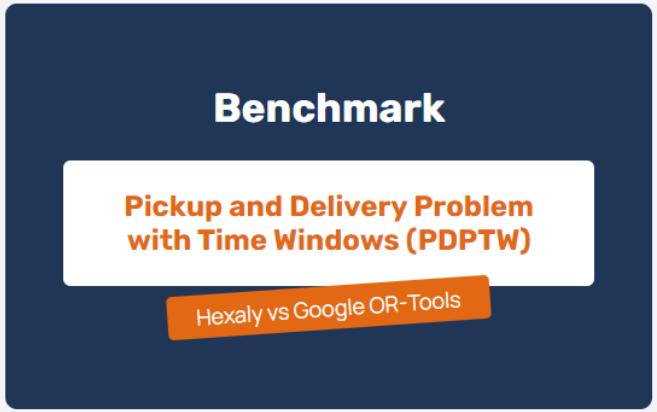
Hexaly vs Gurobi on the Car Sequencing Problem with Paint-Shop Batching Constraints

Read more →



Hexaly vs Gurobi on the Flexible Job Shop Scheduling Problem (FJSP)

Read more →

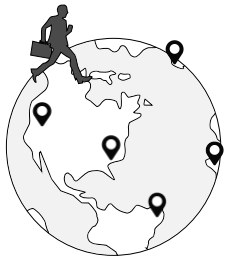


Hexaly vs Google OR-Tools on the Pickup and Delivery Problem with Time Windows (PDPTW)

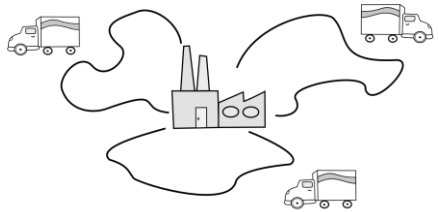
Read more →

Check our examples: www.hexaly.com/docs/last/exampletour

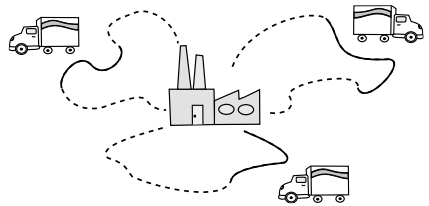
Traveling Salesman



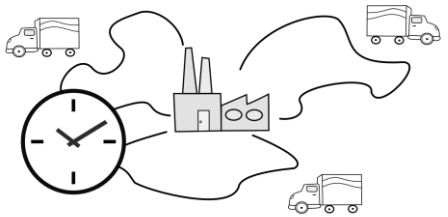
Capacitated Vehicle Routing



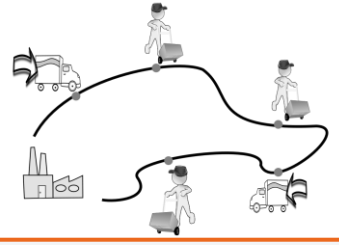
Capacitated Arc Routing



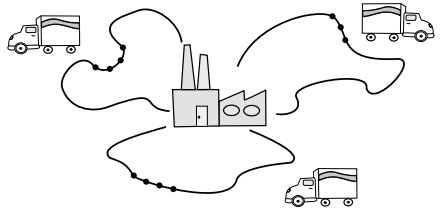
Vehicle Routing with Time Windows



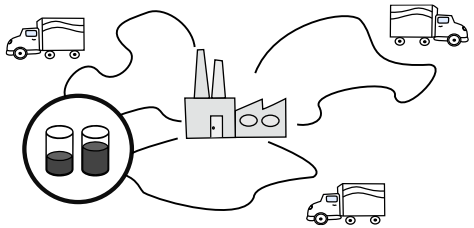
Pickup and Delivery with Time Windows



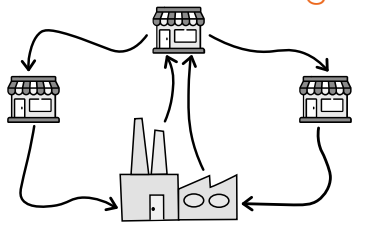
Clustered Vehicle Routing



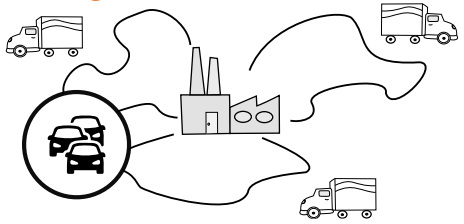
Inventory Routing



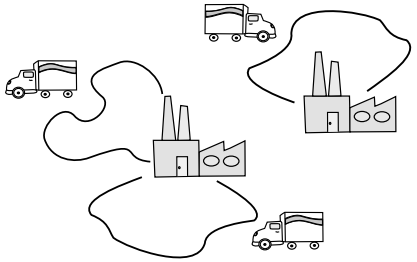
Split Delivery Vehicle Routing



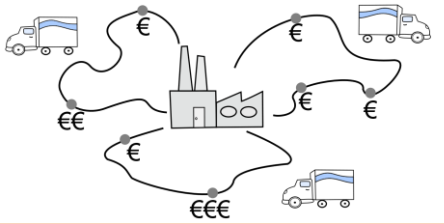
Time-Dependent Vehicle Routing with Time Windows



Location Routing

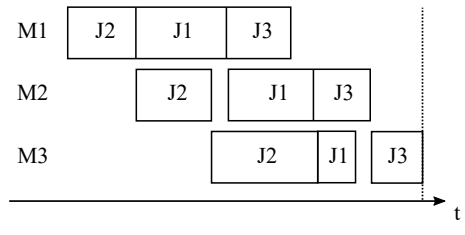


Prize-Collecting Vehicle Routing

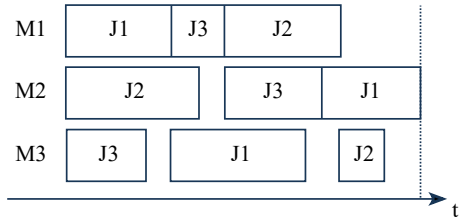


Check our examples: www.hexaly.com/docs/last/exampletour

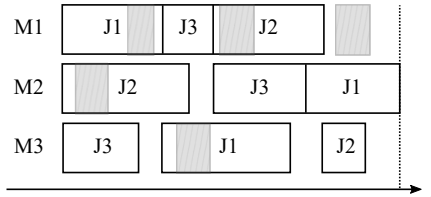
Flow Shop



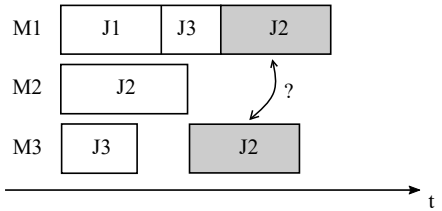
Job Shop



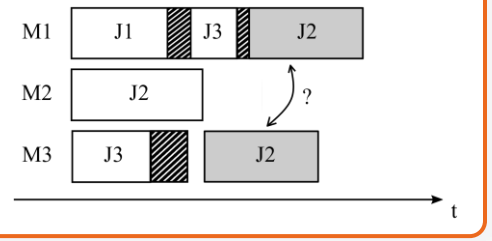
Job Shop Intensity



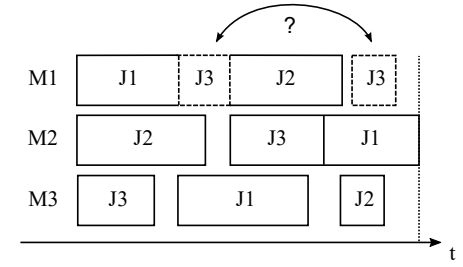
Flexible Job Shop



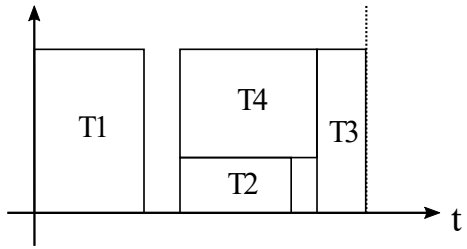
FJSP Setup Times



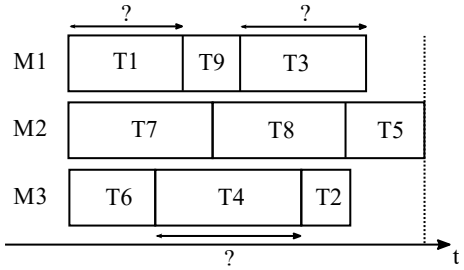
Open Shop



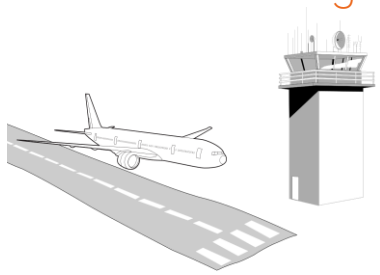
RCPSP



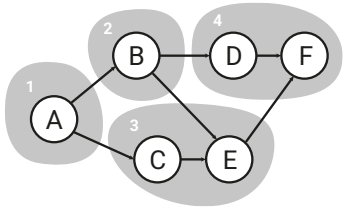
Stochastic scheduling



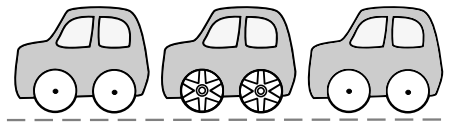
Aircraft Landing



Assembly Line Balancing



Car Sequencing



Movie Shoot Scheduling