Poznan Supercomputing and Networking Center
affiliated to Institute of Bioorganic Chemistry
Polish Academy of Sciences

Poznan University of Technology

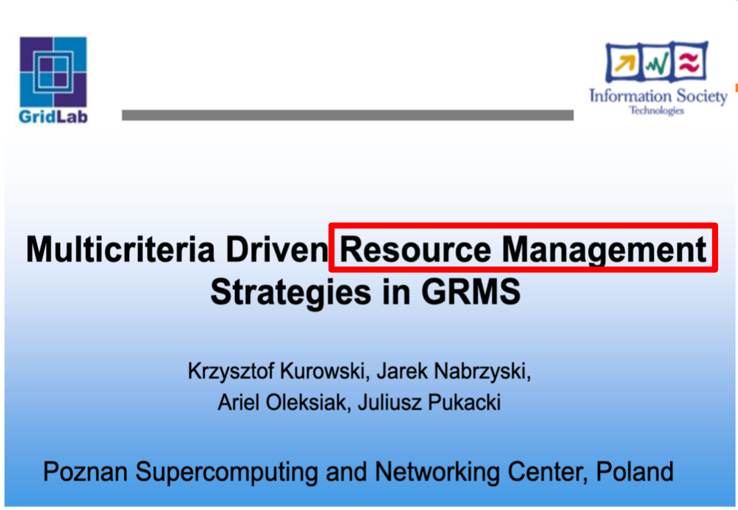
Krakow Quantum Informatics Seminar
Tuesday, 7 July 2020, 9.30-11.00 via Webex

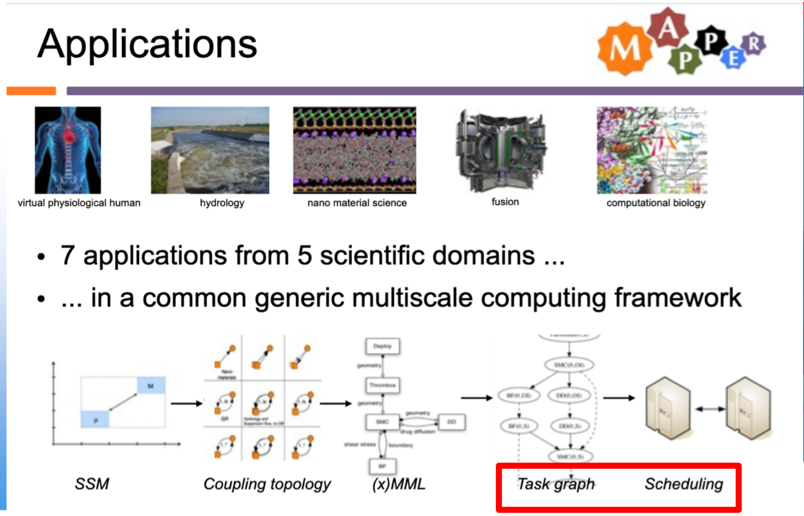
Krzysztof Kurowski, Jan Węglarz, Marek Subocz,
Grzegorz Waligóra, Rafał Różycki
krzysztof.kurowski@man.poznan.pl

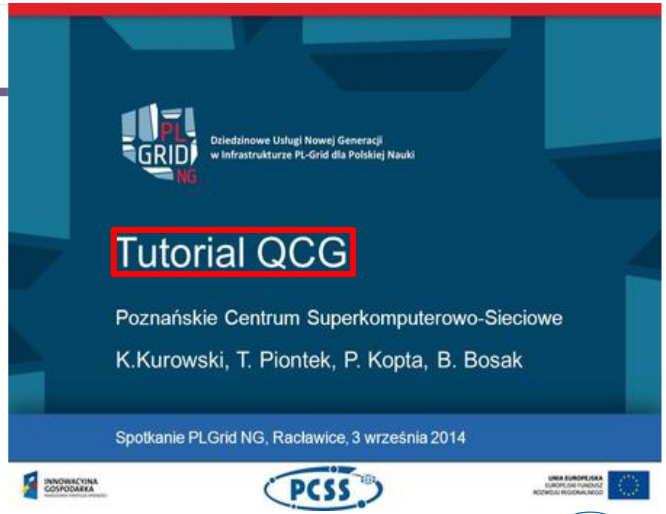
Hybrid Quantum Annealing Heuristic Method for Solving Job Shop Scheduling Problem

About me before quantum era …

Metacomputing, grid computing, OGSA, cloud computing, heterogenous/GPU computing, …





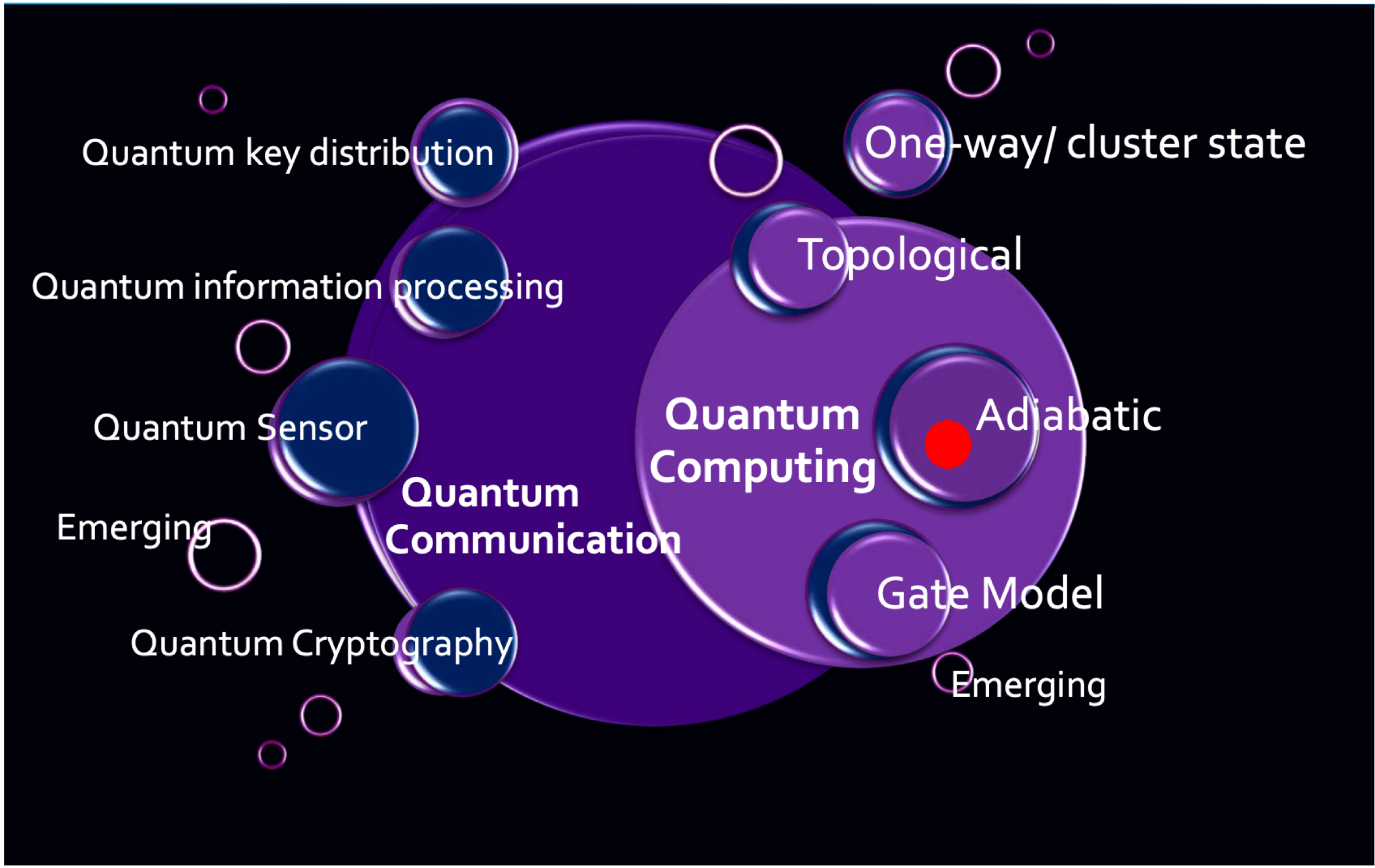


Agenda

- Introduction & Background
- Motivations
- Initial ideas for Job Shop Scheduling Problem (JSSP)
- Problem formulation
- Problem complexity
- Mapping to QUBO
- Variable Pruning
- Our heuristic approach
- Results
- Current Work
- Future Work
- Conclusions

More at: Kurowski K., Węglarz J., Subocz M., Różycki R., Waligóra G. (2020) Hybrid Quantum Annealing Heuristic Method for Solving Job Shop Scheduling Problem. In: Krzhizhanovskaya V. et al. (eds) Computational Science – ICCS 2020. ICCS 2020. Lecture Notes in Computer Science, vol 12142

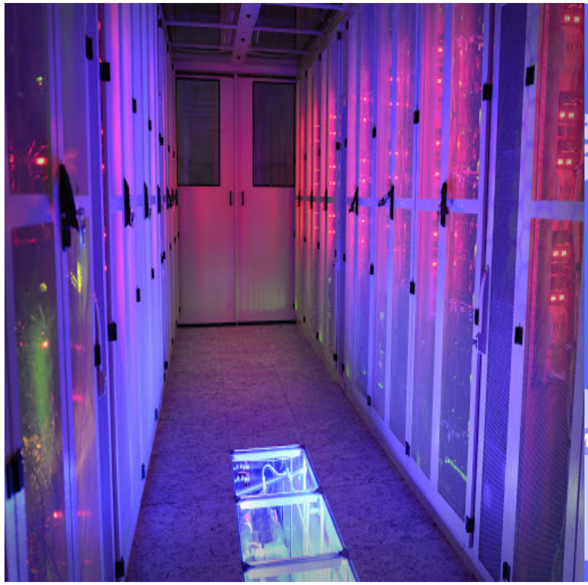
Quantum R&D activities



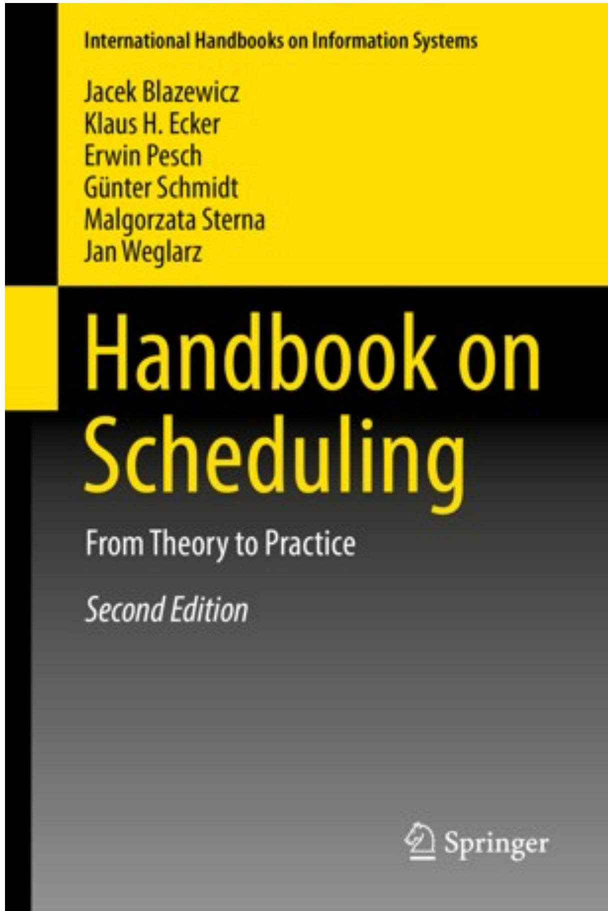
Introduction & Background

Quantum Communication & Computing strategic program at PSNC

- QCC program established at PSNC back in 2016
- Internal and external quantum testbeds available
- Participation in various national and EU R&D activities
- Long-term collaboration with Poznan University of Technology in discrete optimization and scheduling domains



National Laboratory for Photonics and Quantum Technologies in Poland

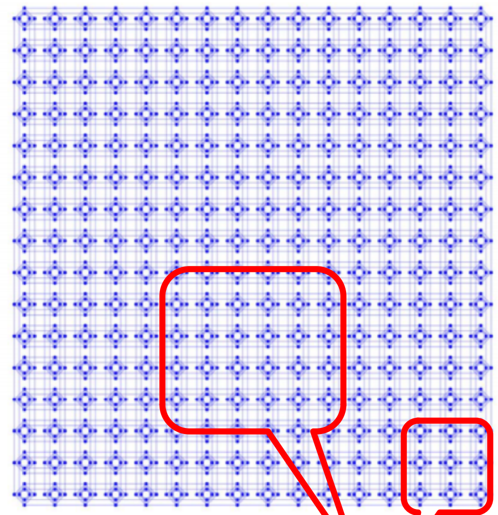


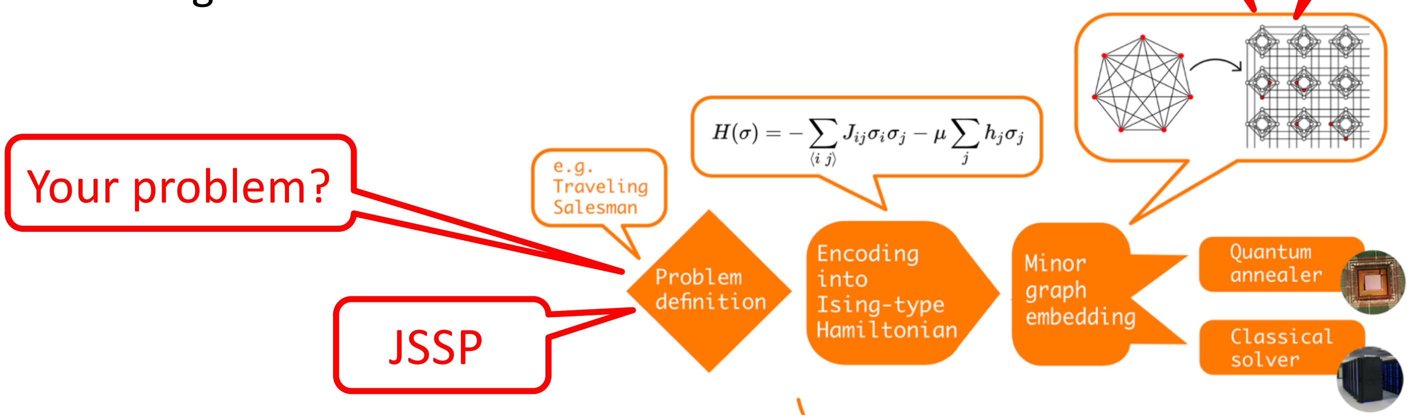
Motivations

Perform real experiments of hybrid algorithms on QPU

- Check experimentally how the recent Chimera D-Wave 2000Q QPU can be used for solving real optimization problems, in particular taken from the scheduling domain, e.g. reference Job Shop Scheduling Problem (JSSP) benchmarks
- Design a new hybrid heuristic approach for solving bigger JSSP instances
- Discover limits and constraints of existing Chimera C16 D-Wave 2000Q QPU in the light of the selected JSSP instances

Chimera C16 - DW 2000Q





Initial ideas

Decompose a main problem into a set of subproblems

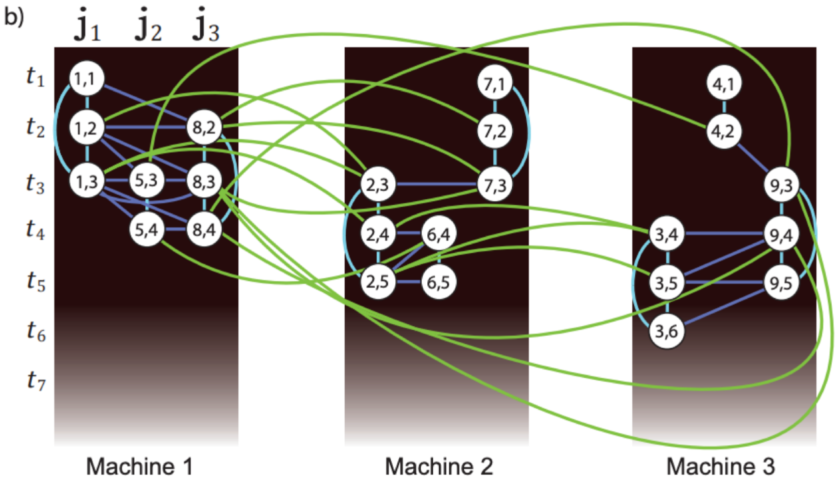
- Decompose the considered JSSP into a set of smaller optimization problems
- Generally, smaller problems fit better into a limited D-Wave 2000Q quantum hardware capacity, but what is ‘small’?
- Possible limitations with regard to precision, connectivity and number of variables already identified
- Identify what and how you can improve your code to be ready for the new D-Wave QPU and Quantum Application Environment (Leap2)
- Tune experimentally various parameters for the heuristic as well as variable pruning and minor-embedding procedures

*

a)

	operation 1	operation 2	operation 3
$\mathbf{j}_1$	$\mathbf{m}_1,\ p=2$	$\mathbf{m}_2,\ p=1$	$\mathbf{m}_3,\ p=1$
$\mathbf{j}_2$	$\mathbf{m}_3,\ p=2$	$\mathbf{m}_1,\ p=1$	$\mathbf{m}_2,\ p=2$
$\mathbf{j}_3$	$\mathbf{m}_2,\ p=1$	$\mathbf{m}_1,\ p=1$	$\mathbf{m}_3,\ p=2$

b)

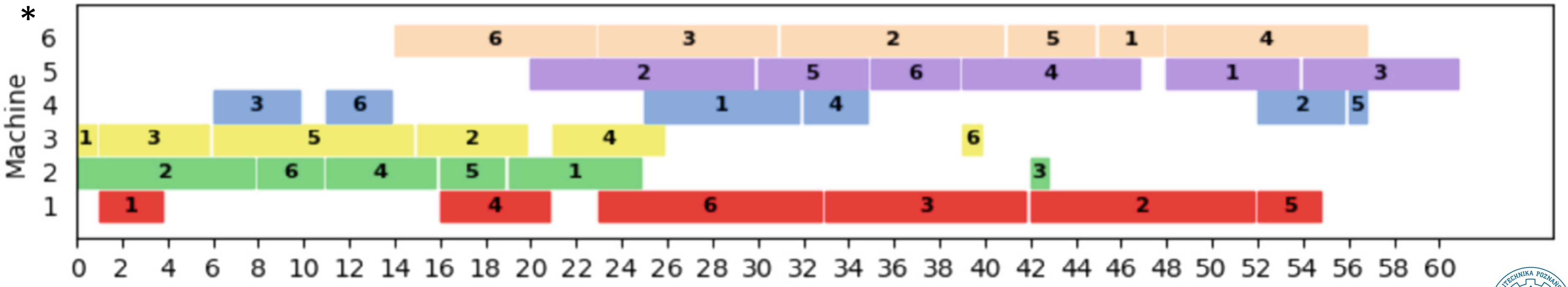


* D. Venturelli, D. Marchand, and G. Rojo, Quantum Annealing Implementation of Job-Shop Scheduling. https://arxiv.org/abs/1506.08479v2

Problem formulation

- The JSSP can be described by a set of jobs $J = \{j_1,\ldots,j_N\}$ that must be scheduled on a set of machines $M = \{m_1,\ldots,m_R\}$. Each job j_n consists of a sequence of L_n operations that have to be performed in a predefined order: $j_n = \{O_{n1} \rightarrow O_{n2} \rightarrow \cdots \rightarrow O_{nLn}\}$
- The processing time of an operation O_i, $i = 1,2,\ldots,k_N$, is p_i which is a positive integer
- Each operation requires for its processing a particular machine
- There can only be one operation running on any given machine at any given point in time
- Each operation of a job needs to be completed before the following one can start

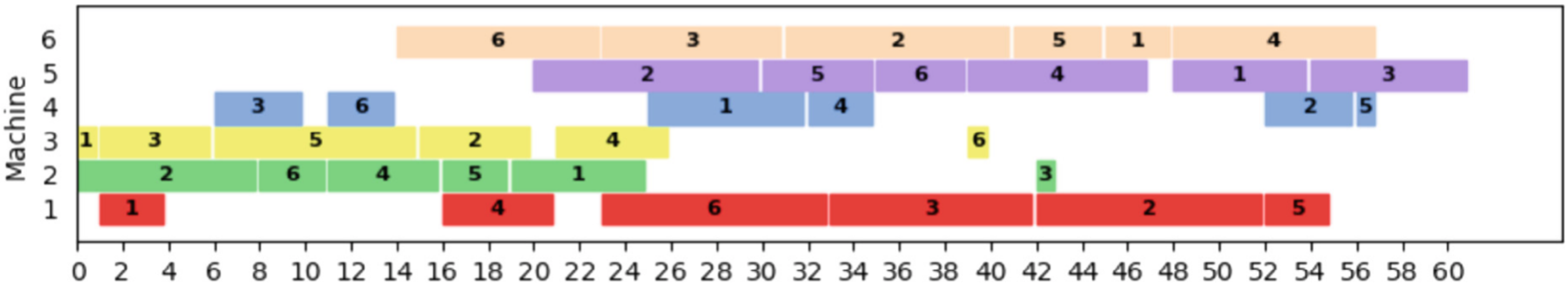
*

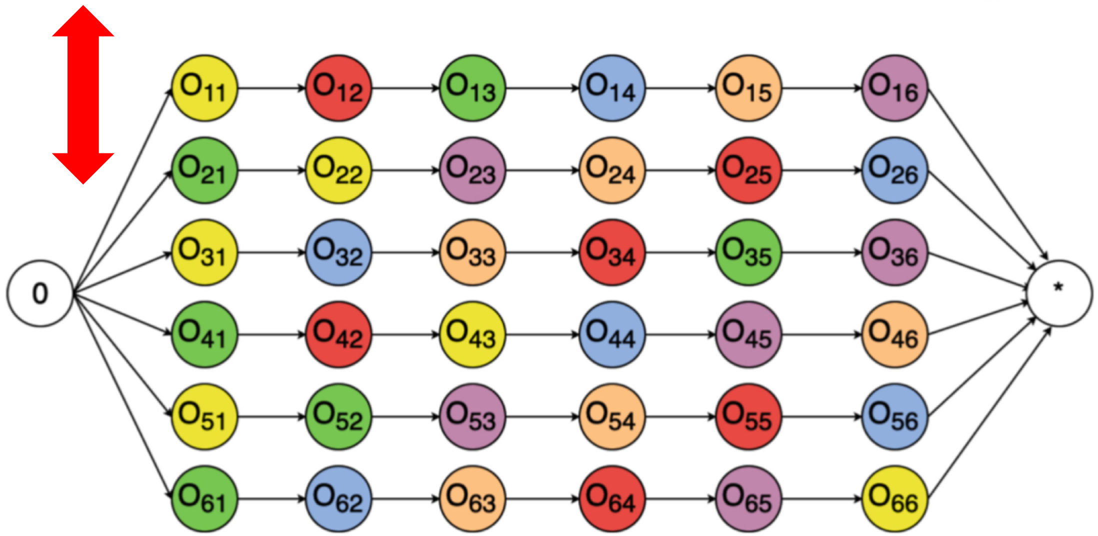


* JSSP ft06 (6x6) instance: http://jobshop.jjvh.nl/instance.php?instance_id=6

Problem formulation

Gantt chart vs. alternative disjunctive graph representation





The objective is to schedule **all operations in a valid sequence** in order to **minimize the schedule length** (or the makespan), which is the completion time of the last running job.

Decision version of JSSP is considered – whether or not a valid solution exists within a given makespan (deadline).

Problem complexity

Complex NP-hard problem

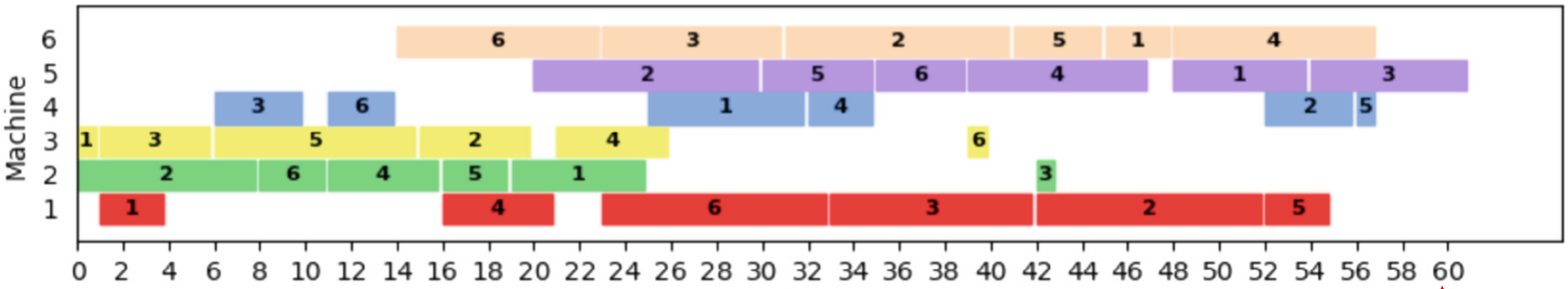
- Job Shop Scheduling Problem is NP-hard in the strong sense, 20x20 not solved yet!
- Optimal solutions can not be found within a reasonable time for bigger JSSP instances
- Several polynomial-time heuristics have been developed for finding suboptimal solutions and tested experimentally
- Half of classical heuristics for JSSP have been based on tabu search algorithms, followed by local search, shifting bottleneck, branch and bound, and also simulated annealing techniques *

Instance	Jobs	Machines	Lower bound	LB reference	Upper bound	UB reference	Number of solutions found
⊞ Adams, Balas and Zawack [3]							
⊞ Demirkol, Mehta, and Uzsoy [17]							
⊟ Fisher and Thompson [18]							
ft06	6	6	55	[20] *(1971)*	55	[20] *(1971)*	**53**
ft10	10	10	930	[11] *(1986)*	930	[31] *(1984)*	**13120**
ft20	20	5	1165	[35] *(1975)*	1165	[35] *(1975)*	**3**
⊞ Lawrence [32]							
⊞ Applegate and Cook [4]							
⊞ Storer, Wu and Vaccari [51]							
⊞ Taillard [52]							
⊞ Yamada and Nakano [63]							

* van Hoorn, J.J.: The current state of bounds on benchmark instances of the job-shop scheduling problem. J. Sched. 21(1), 127–128 (2017). https://doi.org/10.1007/s10951-017-0547-8

JSSP mapping to QUBO

Quadratic Constraints and the Objective Function



Minimum of H_T is 0 if a schedule satisfies all constraints (objective function)

$$Obj(x) = \sum_{i=0}^{N-1} a_i x_i + \sum_{i=0}^{N-2} \sum_{j=i+1}^{N-1} b_{ij} x_i x_j$$

Problem Expression	Terms			
	Linear coefficient	Quadratic coefficient	Variable	States
QUBO (scalar)	a_i	$b_{i,j}$	q_i	{0 , 1}
QUBO (matrix)	$Q_{i,i}$	$Q_{i,j}$	x_i	{0 , 1}
Ising	h_i	$J_{i,j}$	s_i	{-1, 1}
Graph	Node Weight	Edge Strength	Node	
QPU	Qubit Bias	Coupling Strength	Qubit State	{Spin Up, Spin Down}

Assign a set of binary variables for each operation O_i

$$x_{it} = \begin{cases} 1 : \text{operation } O_i \text{ starts at time } t \\ 0 : \text{otherwise} \end{cases}$$

t is bounded by makespan

JSSP mapping to QUBO

Quadratic Constraints and the Objective Function

Count the number of precedence violations among operations

$$h_1(x) = \sum_n \Big(\sum_{\substack{k_{n-1} < i < k_n \\ t + p_i > t'}} x_{it} x_{i+1,t'} \Big)$$

Only one job running on each machine at any time

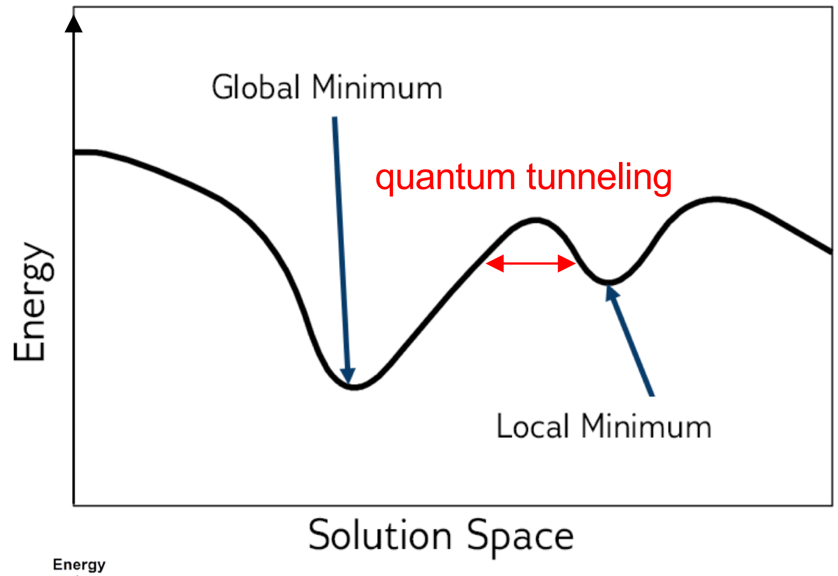
$$h_2(x) = \sum_m \Big(\sum_{(i,t,k,t') \in R_m} x_{it} x_{kt'} \Big)$$

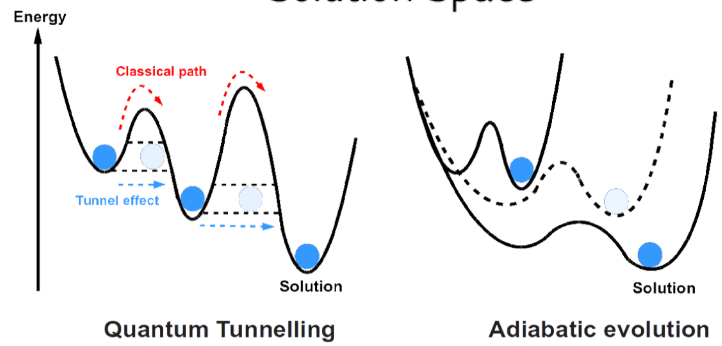
An operation must start once and only once

$$h_3(x) = \sum_i \Big(\sum_t x_{it} - 1 \Big)^2$$

$$H_T(x) = \eta h_1(x) + \alpha h_2(x) + \beta h_3(x)$$

Think of it more as declarative programming not procedural programming!





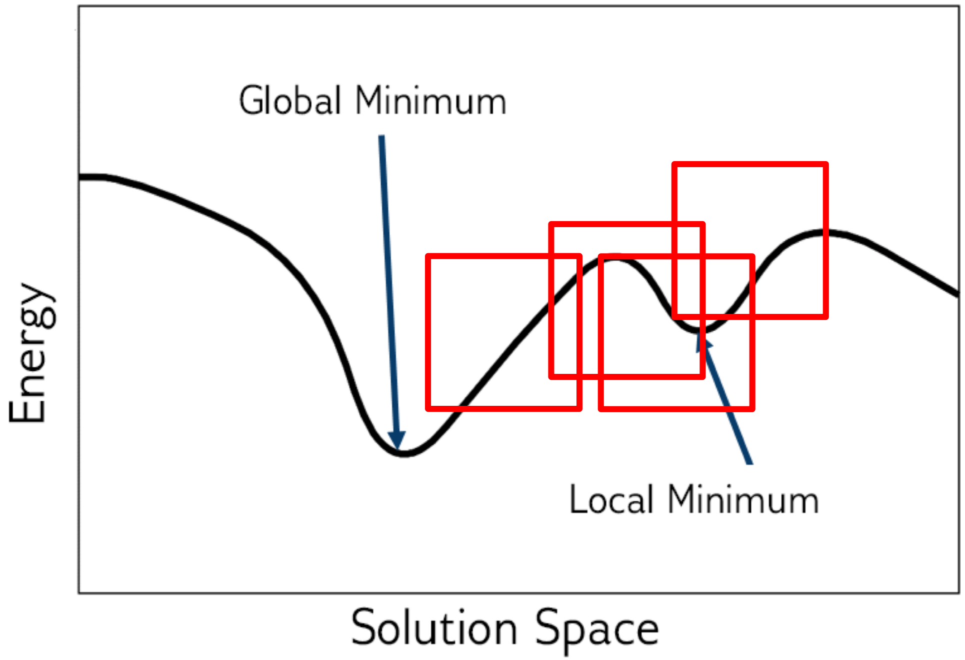
Variable Pruning

Global vs local optimum

- Practically, we can roughly estimate a number of variables according to the following formula:

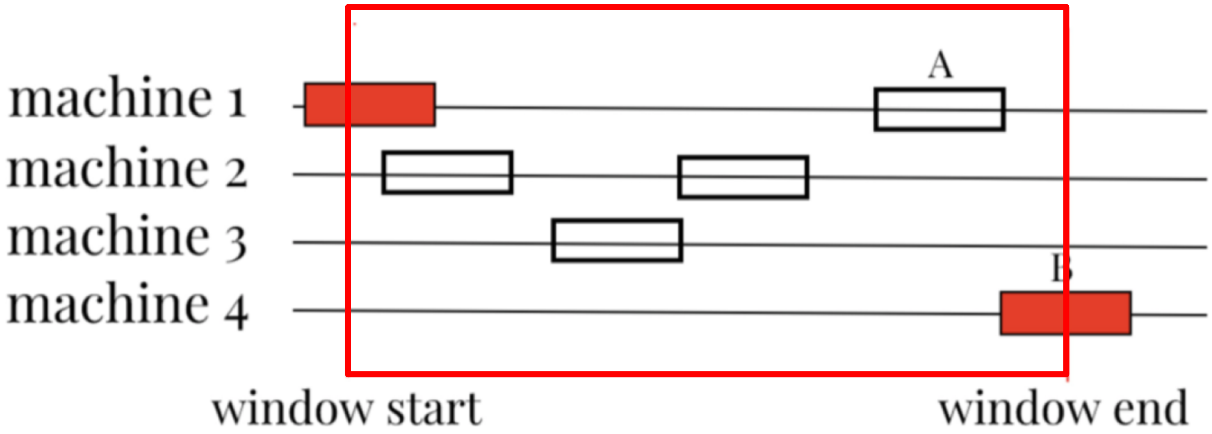
  $number\_Of\_Operations * investigated\_Time\_Span$

- Discover variables that can be disabled during a pre-processing phase (~ half of variables disabled during our experiments)
- Keep in mind that some operations are very long (even up to 15 time units) and have to deal with the time-indexed problem
- With default settings it is getting difficult to define BQM (Binary Quadratic Model) even for around 30 or more variables on D-Wave 2000Q



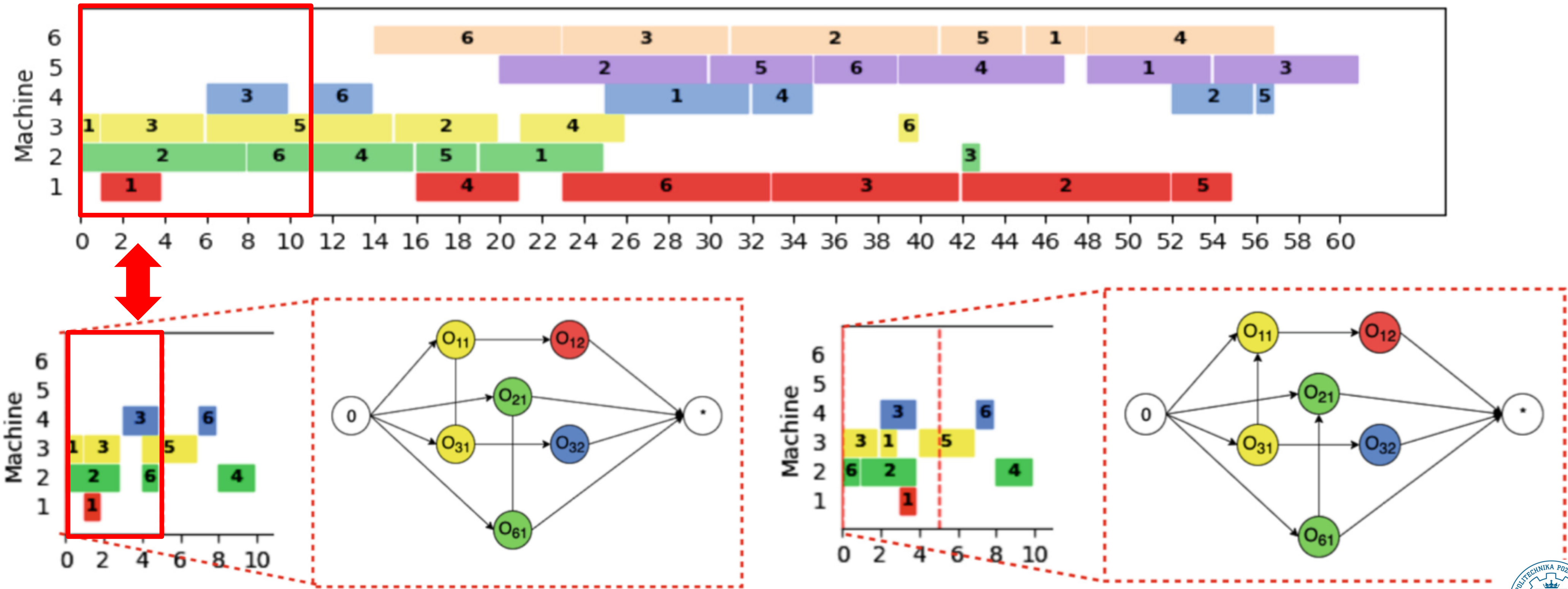
Our heuristic approach

Dividing a JSSP schedule into *a set of **processing windows*** for all machines
and only some job operations (different processing window configurations possible)



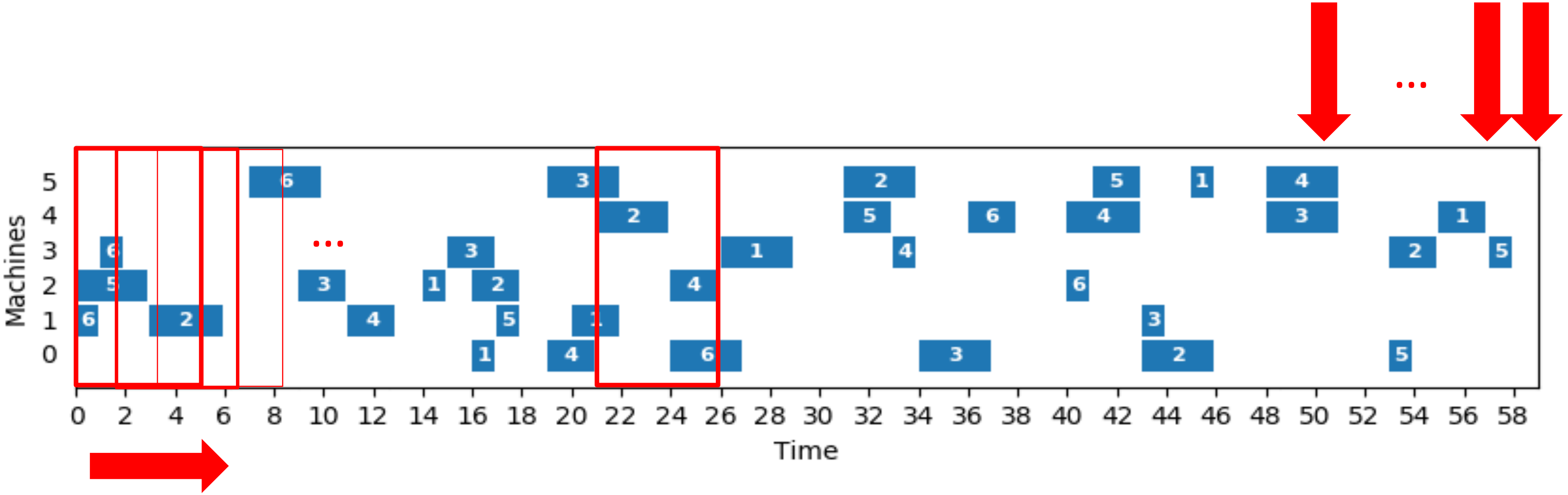
Our heuristic approach (processing windows)

Example: The initial disjunctive graph representation of the feasible scheduling solution within a ***processing window*** with the undirected arcs O_{11}, O_{31} and O_{21}, O_{61} turned by the heuristic into directed arcs during the quantum annealing optimization process



Our heuristic approach (working demo)

You need to wait a bit to see some improvements ;-)

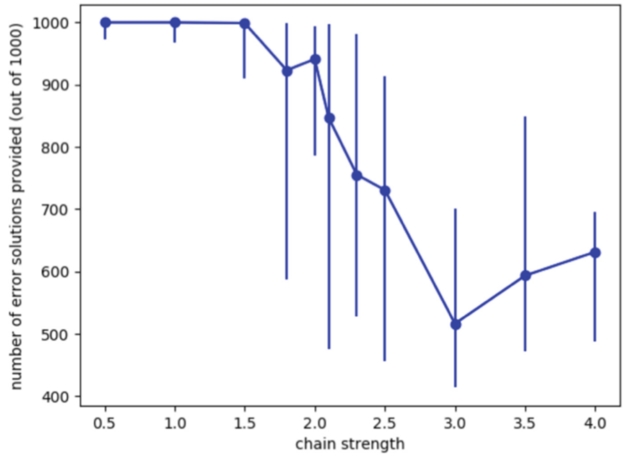


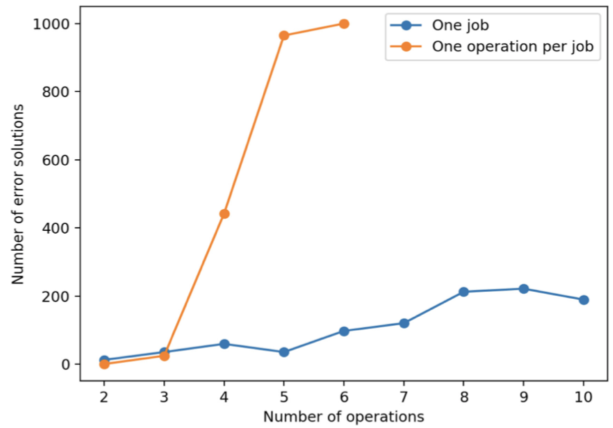
Another trick - job operation length squashing!

Results

Embedding and Qubits Chain Strength

- Various experiments with our heuristic performed to make sure that all the relevant controlling parameters and configurations were used efficiently on the QPU, including:
  - selected the *minimum_classical_gap = 2.0* to provide enough energy to differentiate the ground state from other states in the QPU (smaller values possible but more errors)
  - used the *EmbeddingComposite* method to minor-embedding our problem (other pre- and post-processing methods tested, still a room for improvements)
  - various values of the another controlling *extended_j_range* parameter tested to increase the strength of minor embedding coupling (no significant impact noticed)
  - The *chain_strength c* bars based on the standard deviation and their impact on the JSSP solutions quality, around *c=3.0* best results collected





Results

Improvements achieved

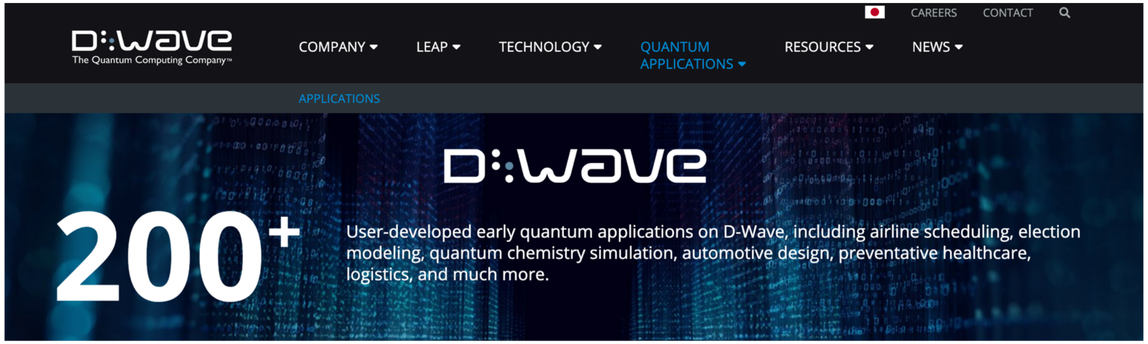
- Our quantum annealing heuristic for solving JSSP improved the quality of solutions by reducing the makespan of randomly generated schedules down to **58** (60 reported in the paper as local minimum)
- No global optimum achieved for a relatively small JSSP instance (**yet ;-)**
- Experimental studies on the D-Wave 2000Q with the well-known scheduling JSSP test instance ft06 show that classical heuristics proposed for the JSSP still outperform our quantum annealing-based approach reaching a set of many schedules with **optimal makespan 55**.

Instance	Jobs	Machines	Lower bound	LB reference	Upper bound	UB reference	Number of solutions found
Fisher and Thompson [18]							
ft06	6	6	55	[20] *(1971)*	55	[20] *(1971)*	**53**

Current Work

Smart Embedding & Scalability

- Open source code available on the github repository with benchmarking JSSP data sets *
- Official release at featured D-Wave apps, collaborating now with D-Wave engineers for further improvements
- Much bigger allocations required on Chimera/Pegasus QPU for more detailed analysis and benchmarking tests
- Higher level APIs together with data structures and practical examples needed for other scheduling problems, e.g. nurse scheduling



Job Shop Scheduling

Poznań Supercomputing and Networking Center

Manufacturing, Optimization

This is a heuristic approach on how to optimally schedule jobs using a quantum computer. Given a set of jobs and a finite number of machines, how should we schedule the jobs on those machines such that all the jobs are completed at the earliest possible time?

Device Count	C16	P6	P12	P16
Qubits	2048	680	3080	5640
Couplers	6000	4784	21764	40484
max degree	6	15	15	15

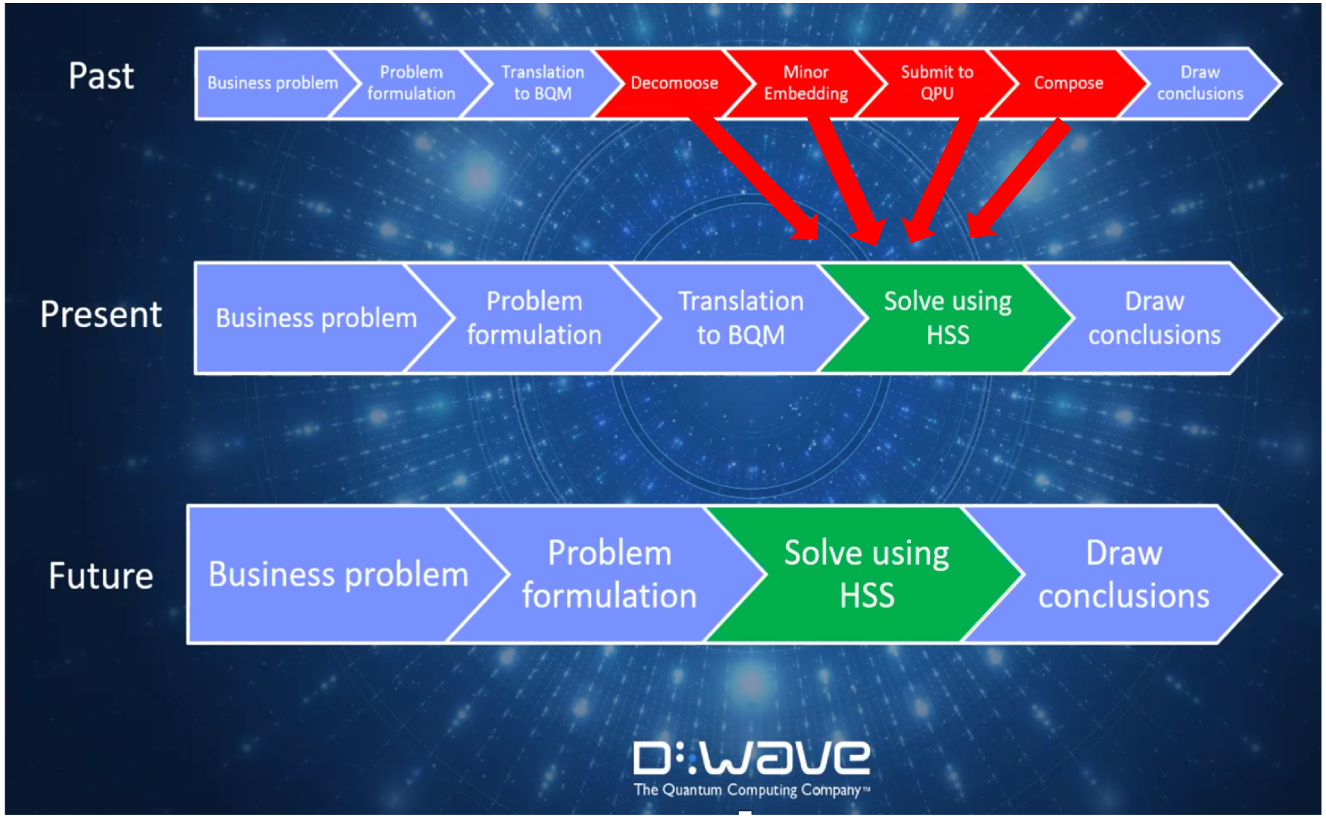
* https://github.com/mareksubocz/QuantumJSP

Future Work

Embedding and Qubits Chain Strength

- Recent Hybrid Solver Service (HSS) release gives users the ability to solve much larger real-world problems with a shorter learning curve
- Porting our heuristic implementation code with HSS and support from D-Wave engineers
- We expect to run the same heuristic successfully for bigger JSSP instances on the new Pegasus QPU
- Expect new experimental results in weeks, check out our github updates

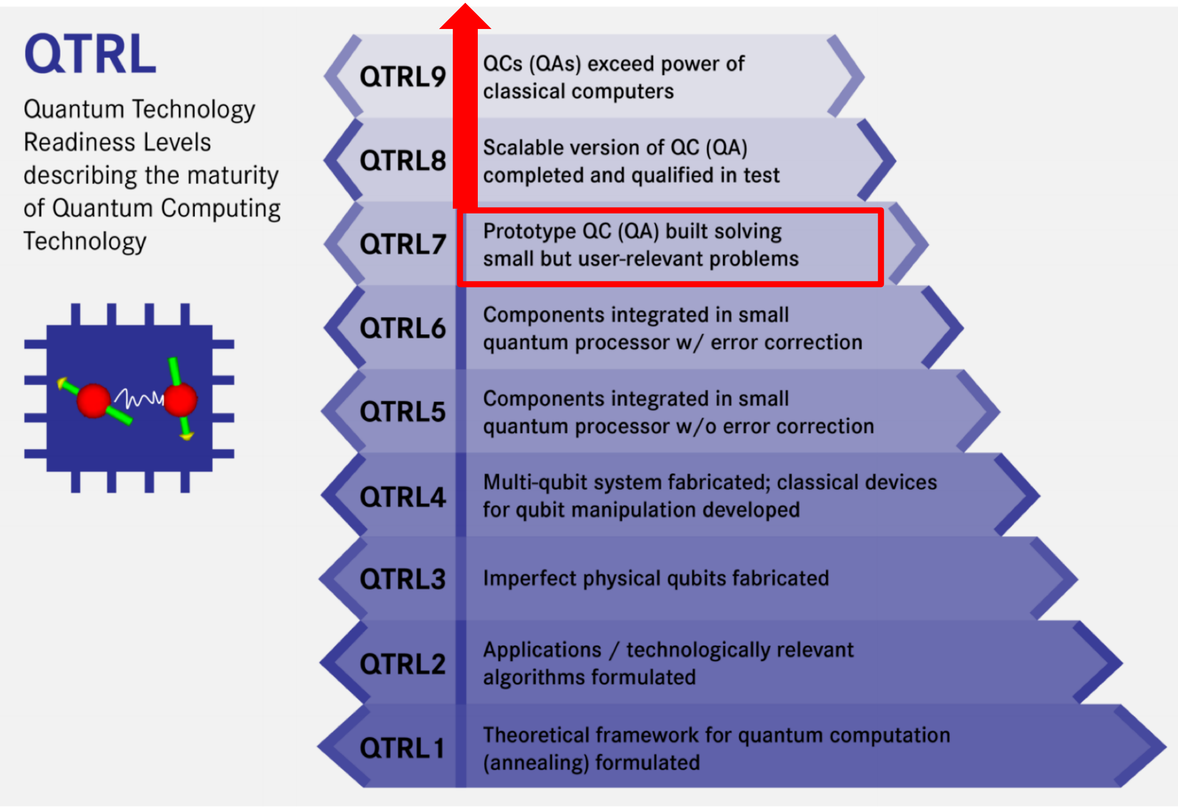
*



* Source: D-Wave company

Conclusions

- Our improved the quality of solutions by reducing the makespan of randomly generated schedules down to ~~60~~ 58 (local minima), the global optimum at 55 for the JSSP ft06 test instance not reached yet ;-)
- Our approach can be easily extended, modified and applied to other scheduling problems by researchers, as we have released the source code of our heuristic and adding HSS support – ongoing effort
- We intend to consider other classical scheduling problems, and test our hybrid heuristic approach to divide large problem domains into small subproblems, e.g. testing global convexity and *smart* windowing



* Source: https://www.fz-juelich.de/ias/jsc/EN/Research/ModellingSimulation/QIP/QTRL/_node.html

Questions?

Thank You!