Comparison of algorithms and encoding schemes for workflow scheduling

Julia Plewa
Joanna Sieńko

Supervisor dr inż. Katarzyna Rycerz

Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie
AGH University of Science and Technology

9.11.2021

Agenda

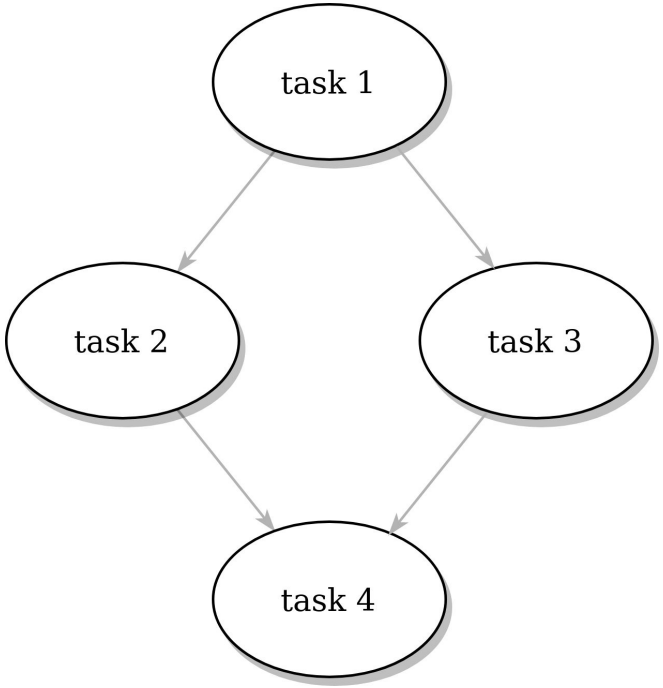
1. Goals of our thesis
2. Workflow scheduling
3. Hybrid algorithms
4. Encodings
5. QAOA mixers
6. Experiment design
7. Evaluation of the results
8. Conclusions & Future work

Goals

- to analyze and improve the **existing implementations** of the workflow scheduling problem
- to implement and compare **different encoding schemes** and use them to solve larger problems
- to test and compare the performance of **QAOA and VQE**
- to experiment with different **classical optimizers**
- to implement and analyze other possible improvements such as **QAOA mixers**
- to test the efficacy of the implementation using a **quantum computer simulator** and **noise models** based on real hardware

Workflow scheduling

- given:
  - *N* tasks
  - *M* types of machines
  - a directed acyclic graph (DAG) describing the order of tasks
  - matrix $c_{i,j}$ detailing the cost of executing task *i* on machine of type *j*
  - matrix $t_{i,j}$ detailing the time of executing task *i* on machine of type *j*
  - deadline *d*
- goal: to minimize the overall cost of executing all tasks while maintaining the required order and meeting the deadline



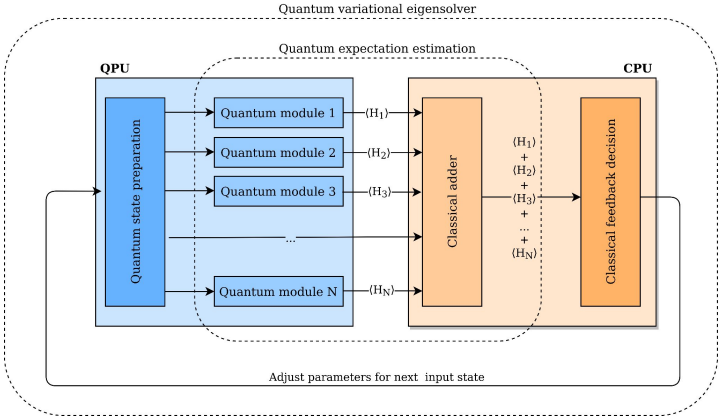
Objective function

- algorithm output:
  - matrix X: $X(i,j) = \begin{cases} 1 & \text{if task } i \text{ is to be run on machine of type } j \\ 0 & \text{otherwise} \end{cases}$
  - matrix Y: $Y(l,k)$ determines the value of the *l*th slack on path *k*

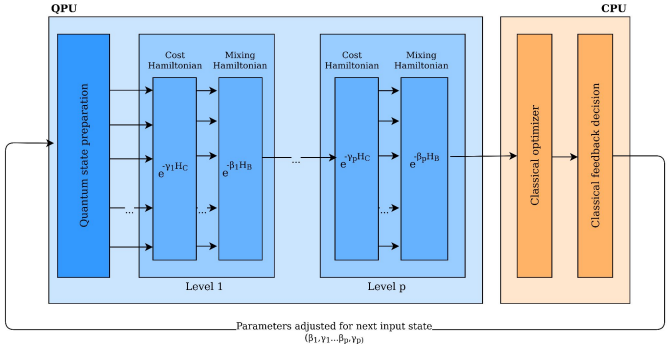
$$O(X,Y) = A\sum_{i}^{N}\sum_{j}^{M} c_{i,j}x_{i,j} + B\cdot\sum_{k}^{R}(d-(\sum_{i}^{N}\sum_{j}^{M} p_{i,k}t_{i,j}x_{i,j} + \sum_{l}^{s_k} 2^l y_{k,l}))^2$$
$$+C\cdot\sum_{i}^{N}(1-\sum_{j}^{M} x_{i,j})^2$$

Hybrid algorithms – VQE and QAOA

- hybrid algorithms perform classical and quantum computations interchangeably in a loop
- QAOA is in a way a special case of VQE with a focus on combinatorial optimization problems



VQE (*Variational Quantum Eigensolver*)



QAOA (*Quantum Approximate Optimization Algorithm*)

Types of encoding

Name	Favorite color
Eve	red
Bob	yellow
Alice	blue
Jane	green

one-hot

Name	Prefers red	Prefers green	Prefers blue	Prefers yellow	Prefers pink
Eve	1	0	0	0	0
Bob	0	0	0	1	0
Alice	0	0	1	0	0
Jane	0	1	0	0	0

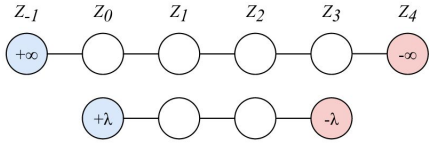
Name	Favorite color
Eve	10000
Bob	00010
Alice	00100
Jane	01000

binary

Name	Favorite color
Eve	$0_{10} = 000_2$
Bob	$3_{10} = 011_2$
Alice	$2_{10} = 010_2$
Jane	$1_{10} = 001_2$

domain wall

Name	Favorite color
Eve	0000
Bob	1110
Alice	1100
Jane	1000

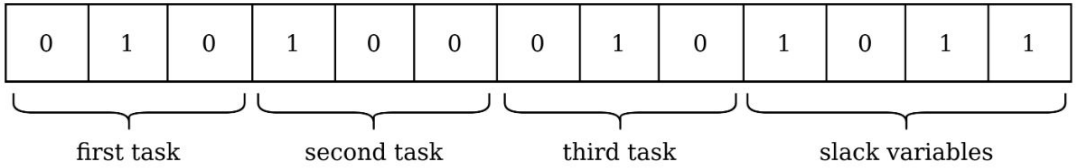


Types of encoding – comparison

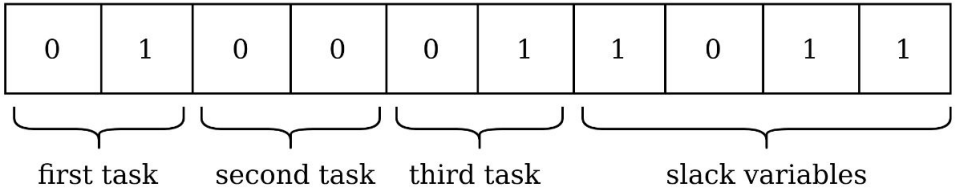
	one-hot	binary	domain wall
number of bits needed to encode *N* values	N	$\lceil log_2 N \rceil$	$N-1$
maximum order needed to decode single values	1	$\lceil log_2 N \rceil$	1^*
maximum order needed to decode two-variable interactions	2	$2\lceil log_2 N \rceil$	2^*
number of couplers	$N(N-1)$	*0* if *N* is a power of *2*, otherwise complicated	$N-2$
intra-variable connectivity	complete	N/A or complicated	linear

Types of encoding – comparison

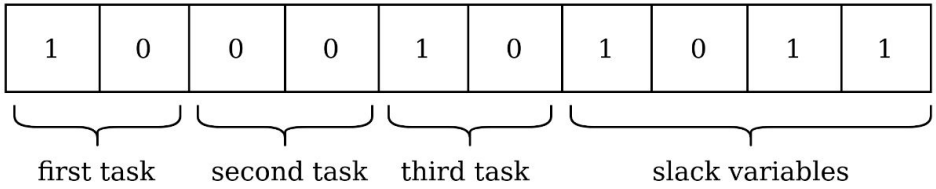
one-hot



binary



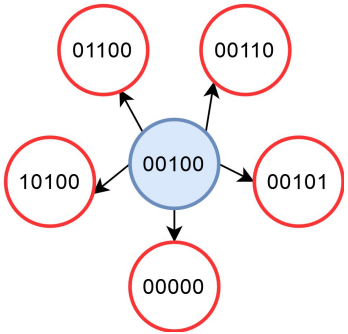
domain wall



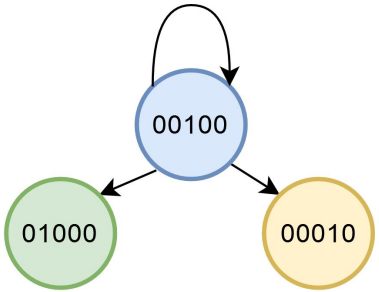
QAOA mixers

- QAOA uses a mixing operator (mixer) which is used to move through the configuration space
- with an encoding-specific mixer it's possible to only consider feasible states

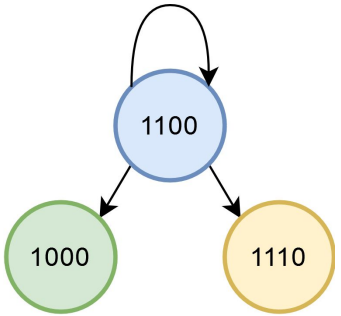
X mixer



XY mixer (one-hot)



domain wall mixer

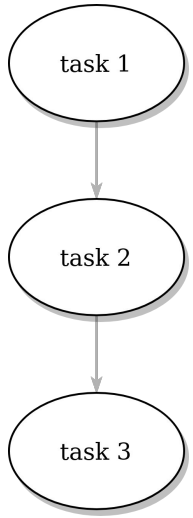


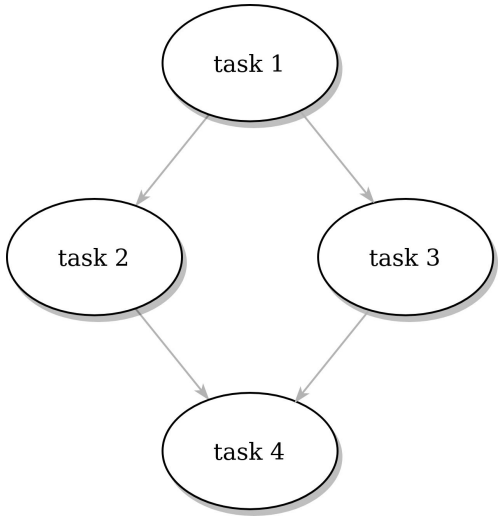
Experiments design

- testing 3 various encodings
- testing 3 algorithms variants
  - QAOA with default X mixer (for all 3 encodings)
  - QAOA with custom mixer dedicated for the given encoding (for two encodings)
  - VQE (for all 3 encodings)
- using *p* and *reps* parameter ranging from 1 to 3
- comparing 4 classical optimizers
  - COBYLA
  - L-BFGS-B
  - NELDER-MEAD
  - POWELL
- each experiment was repeated 1000 times
- every repetition measures the quantum circuit 1024 times (*shots*)
- experiments were conducted on the *Prometheus* supercomputer

Tested problem instances

- **smaller** problem instance:
  3 tasks, 3 machines types, 1 path,
- **larger** problem instance:
  4 tasks, 3 machines types, 2 paths,
- larger instance chosen to contain more than 1 path and to fit in 15 qubits,
- for the larger instance of the problem, only domain wall and binary encoding were tested,
- experiments count:
  - smaller instance $8 \cdot 3 \cdot 4 = 96$
  - larger instance $5 \cdot 3 \cdot 4 = 60$





Types of solutions

- the **optimal** solution,
- a **correct** solution – a solution that represents a correct mapping and fits within the deadline (note: we do not include the optimal solution in the correct solution count),
- a **semi-optimal** solution – a solution identical to the optimal solution, but with an invalid slack configuration,
- a **semi-correct** solution – a solution identical to some correct solution, but with an invalid slack configuration,
- an **incorrect** solution – a solution that is not optimal, correct, semi-optimal, or semi-correct, meaning it either exceeds the dead-line (**incorrect feasible**) or corresponds to an invalid configuration (**incorrect infeasible**, eg. 101 for one-hot).

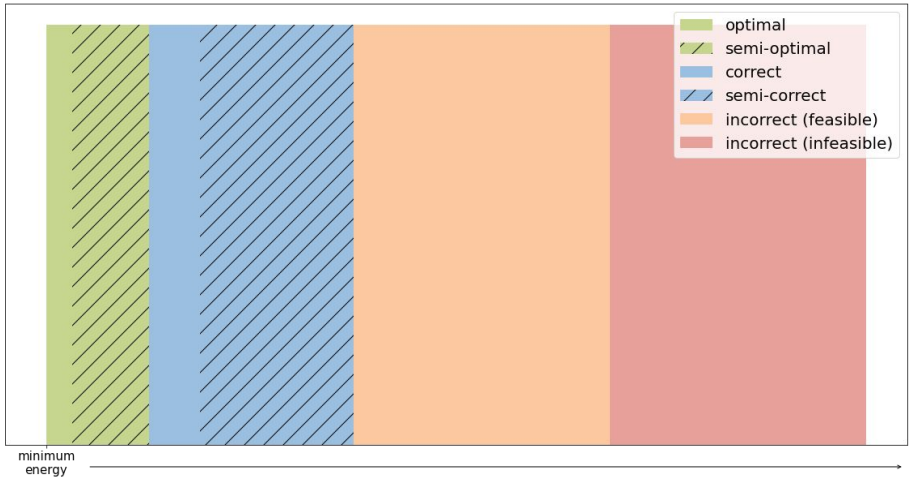


Fig. *Ideal* ordering

Solutions orderings

- we introduced two types of orderings
  - *naive*
  - *feasibility-jump*
- for naive ordering energies of the worst possible solutions are mixed with the energies of semi-optimal solutions, e.g.

solution type	TASK1 TASK2 TASK3 SLACK
optimal	100 010 001 1100
infeasible	110 000 111 1101
semi-optimal	100 010 001 1101

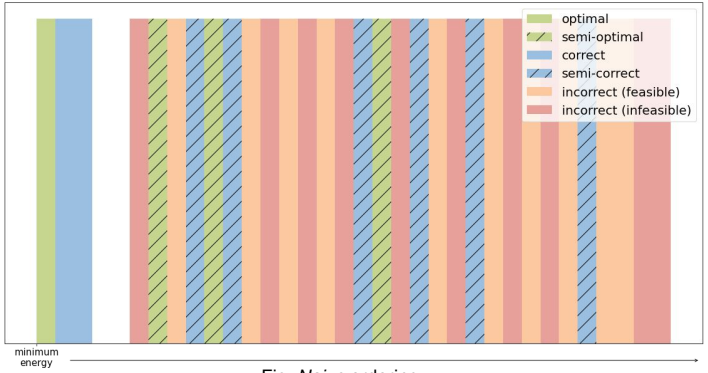


Fig. *Naive* ordering

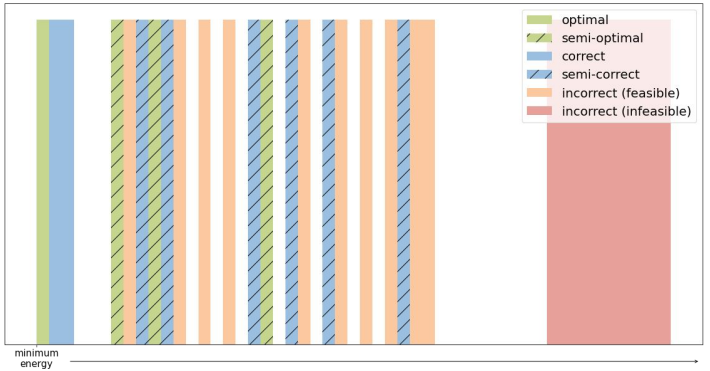
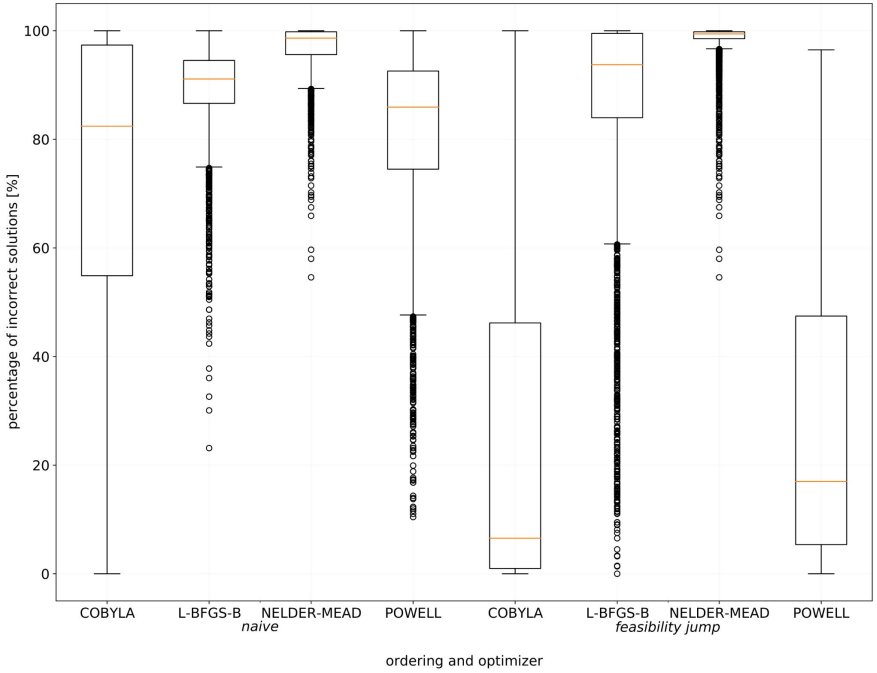


Fig. *Feasibility-jump* ordering

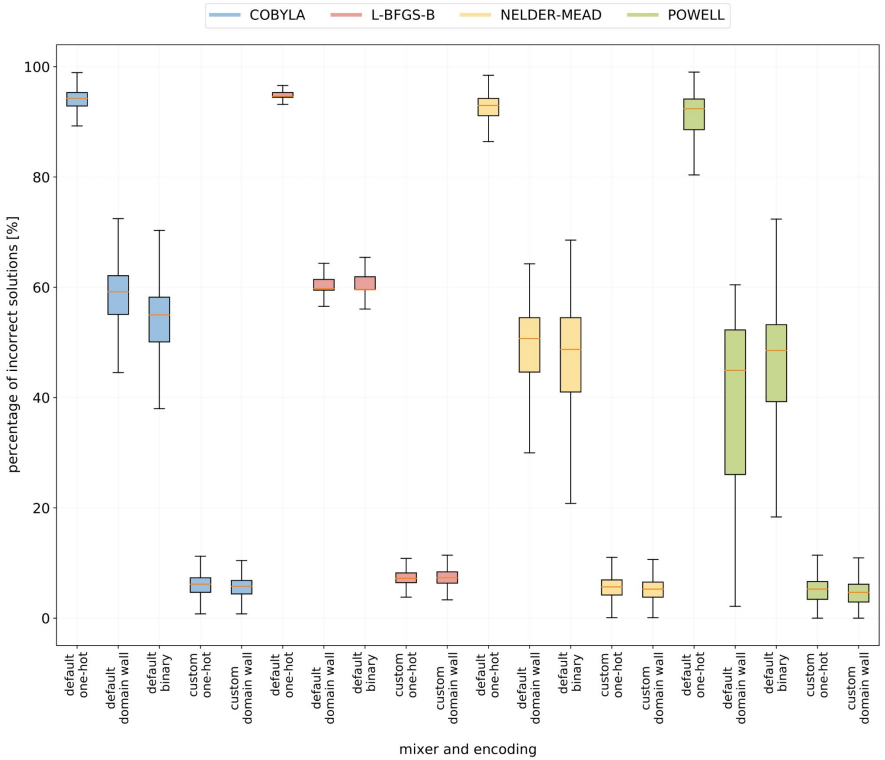
Solutions ordering impact

- results for smaller problem instance, VQE algorithm and one-hot encoding
- for COBYLA optimizer median of incorrect solutions percentage **decreased ~8 times** after switching from *naive* to *feasibility-jump* ordering
- therefore we based our experiments on *feasibility-jump* ordering



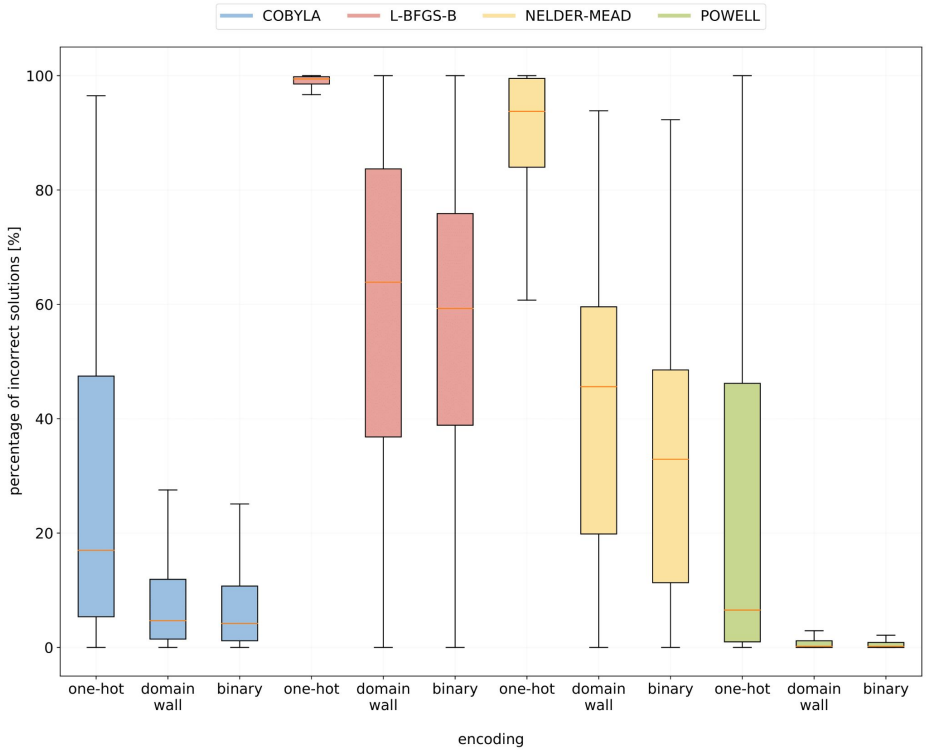
Smaller problem – QAOA

- results are presented in the form of *box-plots*
- one-hot encoding and mixer X returned the biggest number of incorrect solutions
- for default X mixer – but domain wall and binary encodings – decrease of incorrect solutions percentage by around half
- one-hot and domain wall encodings with dedicated mixers performed best, median < 10%
- results do not differ between optimizers, they all share similar results pattern

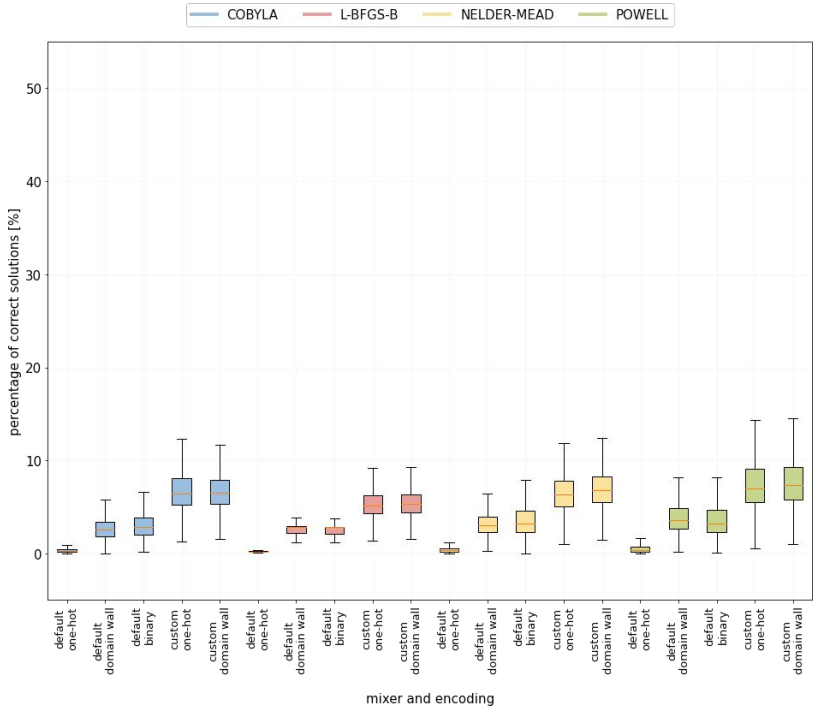


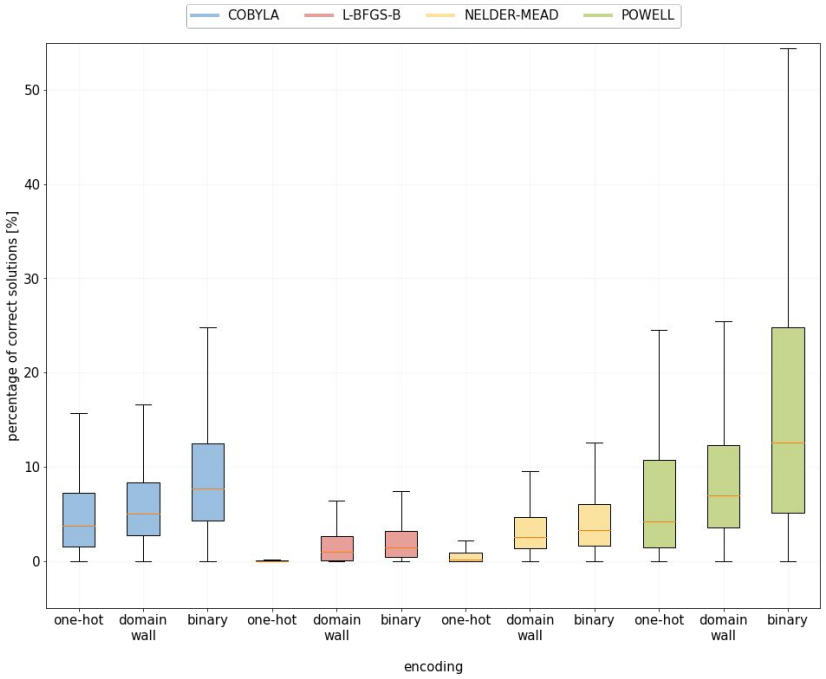
Smaller problem – VQE

- higher variance
- VQE is strongly optimizer dependent
- domain wall and binary encodings performed best
- COBYLA and POWELL optimizers work well with VQE
- for POWELL optimizer with domain wall and binary encodings, the median percentage of incorrect solutions was close to 0%, which is the best result among all results for the smaller instance



Smaller problem – QAOA vs VQE





Smaller problem – noise models

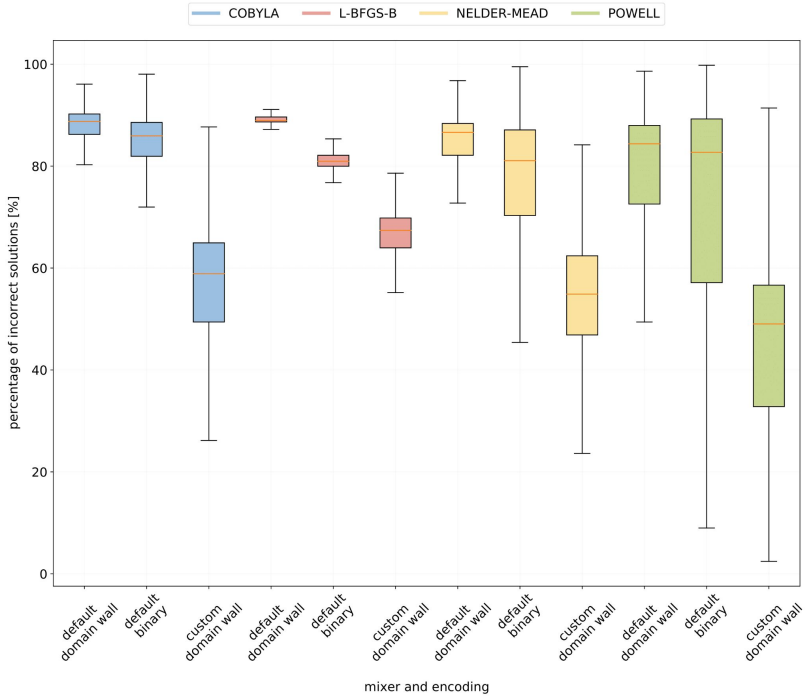
noise model	optimal	semi-optimal	correct	semi-correct	incorrect	feasible
none	98.5%	1.3%	0.0%	0.1%	0.1%	99.9%
Melbourne	0.0%	0.0%	1.8%	27.7%	70.5%	29.5%
Guadalupe	0.0%	0.2%	6.8%	21.5%	71.5%	28.5%

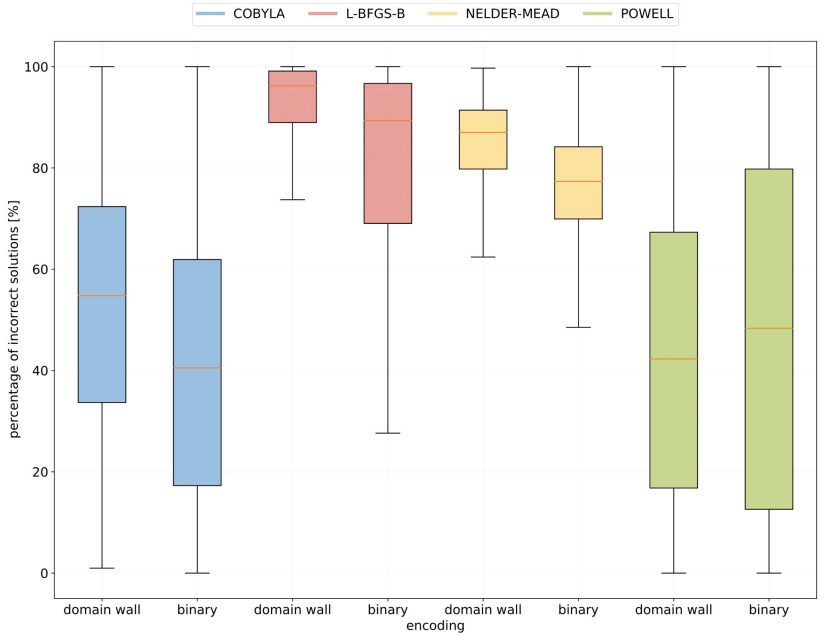
Tab. Results for one-hot encoding and VQE algorithm

noise model	optimal	semi-optimal	correct	semi-correct	incorrect	feasible
none	1.7%	1.7%	27.2%	63.9%	5.6%	100.0%
Melbourne	0.0%	0.4%	0.7%	7.4%	91.5%	9.9%
Guadalupe	0.2%	0.3%	0.3%	7.6%	91.6%	8.7%

Tab. Results for one-hot encoding and QAOA algorithm

Larger problem – QAOA vs VQE





Conclusions

1. Alternative encodings to the popular one-hot allow **larger instances** of the optimization problem to be run.
2. Optimization problems containing **inequalities** are harder to solve. The ratio of incorrect solutions to the number of solutions increases.
3. **Denser encodings** (domain wall and binary) significantly reduce the number of incorrect solutions, which can consequently lead to better results.
4. The selection of **appropriate model parameters** - *weights* - is crucial.
5. Increasing the **p/reps** parameter did not improve the results.
6. Using an encoding-specific mixer in the QAOA algorithm improved the results obtained on the simulator, but on a real quantum computer, **decoherence** from additional gates can make it the QAOA algorithm with the default mixer that gets better results.

Future work

1. Do not use slack variables when solving problems with inequalities by introducing e.g. methods based on Augmented Lagrangians.
2. Analyze how to choose proper model weights.
3. Investigate decoherence caused by the additional gates used by custom QAOA mixers when ran on a real quantum device.
4. Investigate other possible improvements to VQE/QAOA such as CVaR (Conditional Value-at-Risk), which replaces the expectation value with a measure that takes into account only the tail of the probability distribution.

Bibliography

1. Chancellor N.: Domain wall encoding of discrete variables for quantum annealing and QAOA. In: Quantum Science and Technology, vol. 4(4), p. 045004, 2019, doi: 10.1088/2058-9565/ab33c2.
2. Glos A., Krawiec A., Zimborás Z.: Space-efficient binary optimization for variational computing, 2020, arXiv: 2009.07309.
3. Hadfield S., Wang, Z., O'Gorman B., Rieffel E., Venturelli D., Biswas R.: From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz. In: Algorithms, vol 12, 2019, doi: 10.3390/a12020034.
4. Stachoń M.: Solving Optimization problems using Qiskit Aqua. Master's thesis, AGH University of Science and Technology, Kraków, 2020, http://dice.cyfronet.pl/publications/source/MSc_theses/Malgorzata_Stachon_msc.pdf.
5. Plewa J., Sieńko J.: Hybrid algorithms for workflow scheduling problem in quantum devices based on gate model. Master's thesis, AGH University of Science and Technology, Kraków, 2021. http://dice.cyfronet.pl/publications/source/MSc_theses/JPlewa_JSienko_msc.pdf.
6. Tomasiewicz D.: Analysis of D'Wave 2000Q Applicability for Job Scheduling Problems. Master's paper, AGH University of Science and Technology, Kraków, 2020, http://dice.cyfronet.pl/publications/source/MSc_theses/Dawid_Tomasiewicz_msc.pdf.
7. Tomasiewicz D., Pawlik M., Malawski M., Rycerz K.: Foundations for Workflow Application Scheduling on D-Wave System. In: V.V. Krzhizhanovskaya, G. Z'avodszky, M.H. Lees, J.J. Dongarra, P.M.A. Sloot, S. Brissos, J. Teixeira, eds., Computational Science - ICCS 2020, pp. 516-530. Springer International Publishing, Cham, 2020, doi: 10.1007/978-3-030-50433-5_40.