UNIVERSITÀ DI PARMA

Quantum Portfolio Optimization

Corso di Quantum Computing (a.a. 2024/25)

Merenda Saverio Mattia

Contenuto della discussione

- ***Descrivere*** il problema di ottimizzazione del portafoglio finanziario

Contenuto della discussione

- ***Descrivere*** il problema di ottimizzazione del portafoglio finanziario
- ***Introdurre*** l'applicazione del quantum computing al problema

Contenuto della discussione

- ***Descrivere*** il problema di ottimizzazione del portafoglio finanziario
- ***Introdurre*** l'applicazione del quantum computing al problema
- ***Analizzare*** i risultati ottenuti e ***confrontare*** le metodologie

Problema dell'ottimizzazione del portafoglio

- ***Obiettivo:*** selezionare un insieme di asset che **massimizzano** i rendimenti e **minimizzano** il rischio, rispettando un budget

Problema dell'ottimizzazione del portafoglio

- ***Obiettivo:*** selezionare un insieme di asset che <u>massimizzano</u> i rendimenti e <u>minimizzano</u> il rischio, rispettando un budget

$$\min_{x} \left(q x^T \Sigma x - x \mu^T + (1^T x - B)^2 \right)$$

Problema dell'ottimizzazione del portafoglio

- ***Obiettivo:*** selezionare un insieme di asset che <u>massimizzano</u> i rendimenti e <u>minimizzano</u> il rischio, rispettando un budget

$$\min_{x} \left(q x^T \Sigma x - x \mu^T + (1^T x - B)^2 \right)$$

  - $x \in \{0,1\}^n$: vettore delle variabili decisionali binarie (quali asset selezionare)

Problema dell'ottimizzazione del portafoglio

- ***Obiettivo:*** selezionare un insieme di asset che <u>**massimizzano**</u> i rendimenti e <u>**minimizzano**</u> il rischio, rispettando un budget

$$\min_{x}\left(qx^T\Sigma x - x\mu^T + (1^T x - B)^2\right)$$

  - $x \in \{0,1\}^n$: vettore delle variabili decisionali binarie (quali asset selezionare)
  - $\mu \in \mathbb{R}^n$: rendimenti attesi degli asset

Problema dell'ottimizzazione del portafoglio

- ***Obiettivo:*** selezionare un insieme di asset che <u>**massimizzano**</u> i rendimenti e <u>**minimizzano**</u> il rischio, rispettando un budget

$$\min_x \left(qx^T \Sigma x - x\mu^T + (1^T x - B)^2 \right)$$

- $x \in \{0,1\}^n$: vettore delle variabili decisionali binarie (quali asset selezionare)
- $\mu \in \mathbb{R}^n$: rendimenti attesi degli asset
- $\Sigma \in \mathbb{R}^{n \times n}$: covarianza tra gli asset

Problema dell'ottimizzazione del portafoglio

- ***Obiettivo:*** selezionare un insieme di asset che <u>**massimizzano**</u> i rendimenti e <u>**minimizzano**</u> il rischio, rispettando un budget

$$\min_{x} \left(qx^T \Sigma x - x\mu^T + (1^T x - B)^2 \right)$$

  - $x \in \{0,1\}^n$: vettore delle variabili decisionali binarie (quali asset selezionare)
  - $\mu \in \mathbb{R}^n$: rendimenti attesi degli asset
  - $\Sigma \in \mathbb{R}^{n \times n}$: covarianza tra gli asset
  - $q > 0$: avversione al rischio

Problema dell'ottimizzazione del portafoglio

- ***Obiettivo:*** selezionare un insieme di asset che <u>**massimizzano**</u> i rendimenti e <u>**minimizzano**</u> il rischio, rispettando un budget

$$\min_x \left(qx^T\Sigma x - x\mu^T + (1^T x - B)^2 \right)$$

- $x \in \{0,1\}^n$: vettore delle variabili decisionali binarie (quali asset selezionare)
- $\mu \in \mathbb{R}^n$: rendimenti attesi degli asset
- $\Sigma \in \mathbb{R}^{n \times n}$: covarianza tra gli asset
- $q > 0$: avversione al rischio
- B : budget disponibile

Problema dell'ottimizzazione del portafoglio

- ***Obiettivo:*** selezionare un insieme di asset che **massimizzano** i rendimenti e **minimizzano** il rischio, rispettando un budget

$$\min_x \Big(\underbrace{q x^T \Sigma x}_{\text{risk}} \; \underbrace{- x\mu^T}_{\text{return}} \; \underbrace{+ (1^T x - B)^2}_{\text{budget penalty}} \Big)$$

Perché il quantum computing?

- ***Limiti classici:*** complessità cresce <u>**esponenzialmente**</u> con il numero di asset

Perché il quantum computing?

- ***Limiti classici:*** complessità cresce <u>**esponenzialmente**</u> con il numero di asset
- ***Vantaggi***:

Perché il quantum computing?

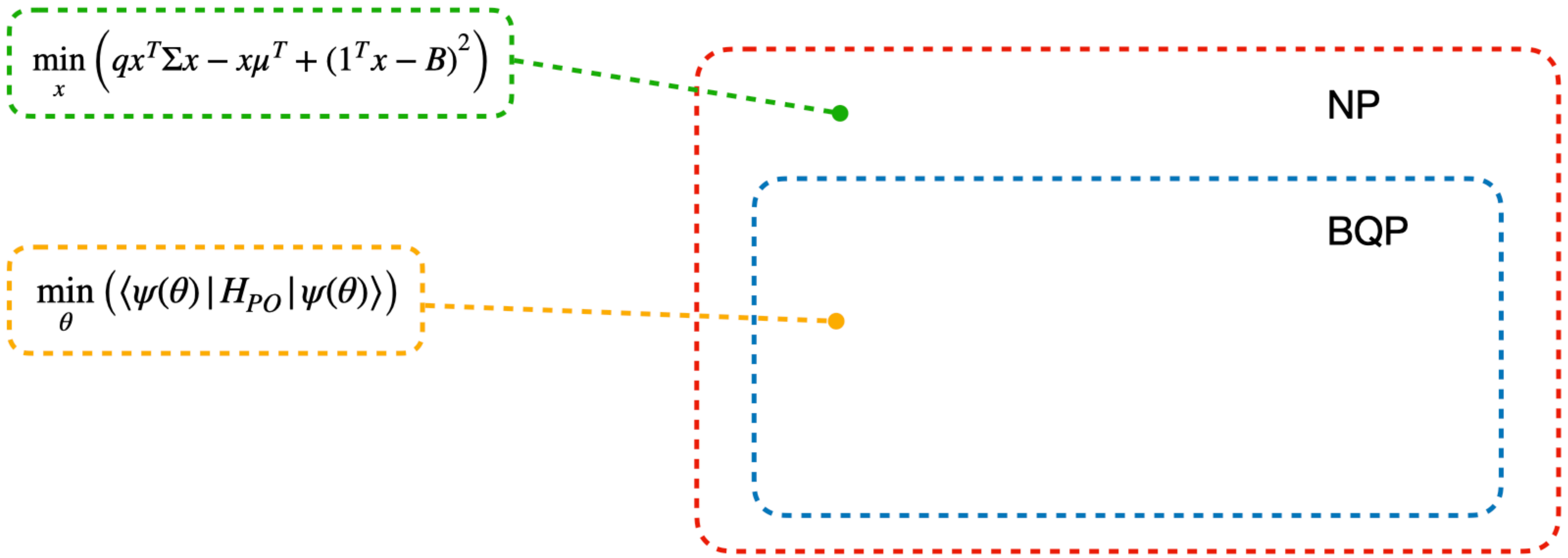
- ***Limiti classici:*** complessità cresce **esponenzialmente** con il numero di asset
- ***Vantaggi***:
  - Capacità di esplorare simultaneamente più configurazioni

Perché il quantum computing?

- ***Limiti classici:*** complessità cresce **esponenzialmente** con il numero di asset
- ***Vantaggi***:
  - Capacità di esplorare simultaneamente più configurazioni
  - Classe di complessità Bounded-error Quantum Polynomial (BQP)

Perché il quantum computing?

- ***Limiti classici:*** complessità cresce <u>**esponenzialmente**</u> con il numero di asset
- ***Vantaggi***:
  - Capacità di esplorare simultaneamente più configurazioni
  - Classe di complessità Bounded-error Quantum Polynomial (BQP)



Algoritmi proposti (1/3)

Branch-and-bound[1]

- ***Metodo classico*** di riferimento per calcolare la *ground truth*

Algoritmi proposti (1/3)

Branch-and-bound[1]

- ***Metodo classico*** di riferimento per calcolare la *ground truth*
- Complessità esponenziale: $O(2^n)$

Algoritmi proposti (2/3)

Variational Quantum Eigensolver (VQE)

- ***Algoritmo ibrido*** che utilizza un ansatz parametrizzato per esplorare lo spazio degli stati

Algoritmi proposti (2/3)

Variational Quantum Eigensolver (VQE)

- ***Algoritmo ibrido*** che utilizza un ansatz parametrizzato per esplorare lo spazio degli stati

$$\min_{\theta} \langle \psi(\theta) \,|\, H \,|\, \psi(\theta) \rangle$$

Algoritmi proposti (2/3)

Variational Quantum Eigensolver (VQE)

- ***Algoritmo ibrido*** che utilizza un ansatz parametrizzato per esplorare lo spazio degli stati

$$\min_{\theta} \langle \psi(\theta) | H | \psi(\theta) \rangle$$

- H : hamiltoniano che rappresenta il problema

Algoritmi proposti (2/3)

Variational Quantum Eigensolver (VQE)

- ***Algoritmo ibrido*** che utilizza un ansatz parametrizzato per esplorare lo spazio degli stati

$$\min_{\theta} \langle \psi(\theta) | H | \psi(\theta) \rangle$$

  - H : hamiltoniano che rappresenta il problema

- ***CPU:*** aggiorna iterativamente i parametri dell'ansatz quantistico θ

Algoritmi proposti (2/3)

Variational Quantum Eigensolver (VQE)

- ***Algoritmo ibrido*** che utilizza un ansatz parametrizzato per esplorare lo spazio degli stati

$$\min_{\theta} \langle \psi(\theta) \,|\, H \,|\, \psi(\theta) \rangle$$

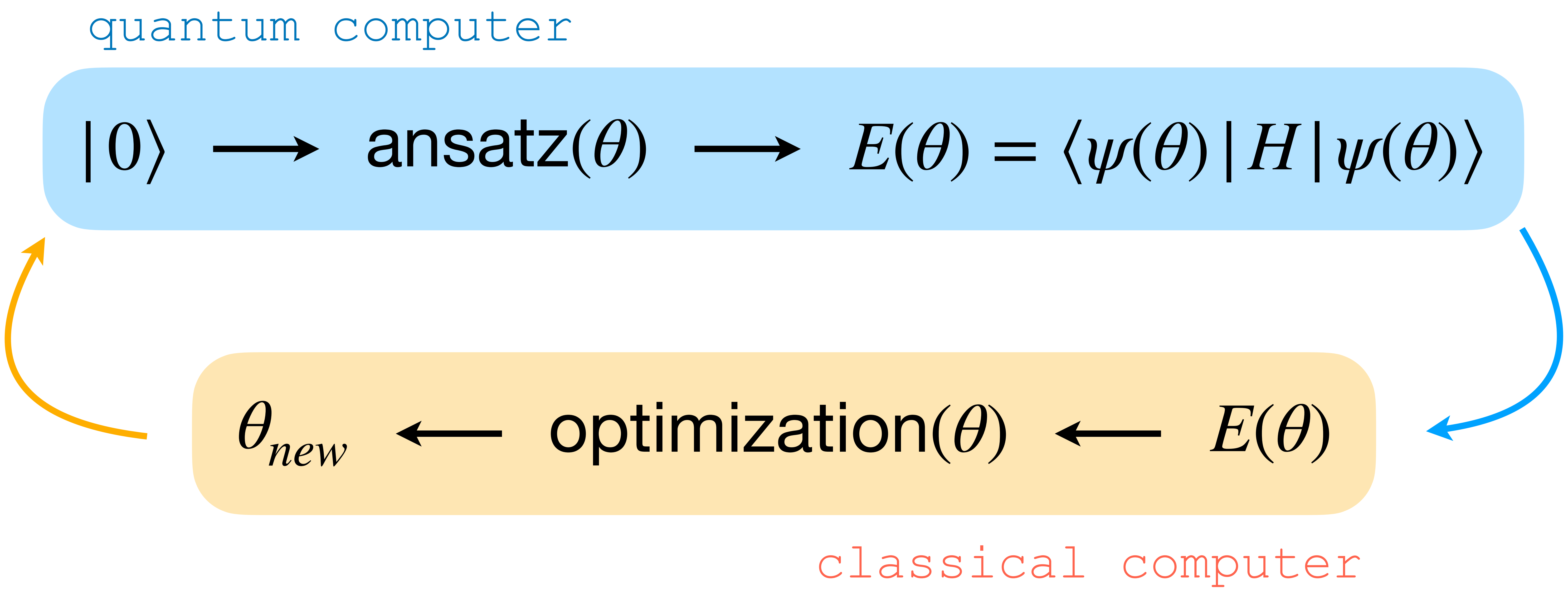
  - H : hamiltoniano che rappresenta il problema

- ***CPU:*** aggiorna iterativamente i parametri dell'ansatz quantistico θ

- ***QPU:*** calcola la funzione obiettivo

Algoritmi proposti (2/3)

Variational Quantum Eigensolver (VQE)



Algoritmi proposti (3/3)

Quantum Approximate Optimization Algorithm (QAOA)

- ***Algoritmo ibrido*** che opera attraverso una sequenza di layers (p) per approssimare la soluzione ottimale di un problema combinatorio

Algoritmi proposti (3/3)

Quantum Approximate Optimization Algorithm (QAOA)

- ***Algoritmo ibrido*** che opera attraverso una sequenza di layers (p) per approssimare la soluzione ottimale di un problema combinatorio

- $\hat{U}_C(\gamma) = e^{-i\gamma\hat{H}_C}$: operatore di costo che promuove l'esplorazione dello spazio delle soluzioni

Algoritmi proposti (3/3)

Quantum Approximate Optimization Algorithm (QAOA)

- ***Algoritmo ibrido*** che opera attraverso una sequenza di layers (p) per approssimare la soluzione ottimale di un problema combinatorio

- $\hat{U}_C(\gamma) = e^{-i\gamma\hat{H}_C}$: operatore di costo che promuove l'esplorazione dello spazio delle soluzioni

- $\hat{U}_M(\beta) = e^{-i\beta\hat{H}_M}$: operatore di mixing che codifica i vincoli e la funzione obiettivo

Algoritmi proposti (3/3)

Quantum Approximate Optimization Algorithm (QAOA)

- ***Algoritmo ibrido*** che opera attraverso una sequenza di layers (p) per approssimare la soluzione ottimale di un problema combinatorio

- $\hat{U}_C(\gamma) = e^{-i\gamma\hat{H}_C}$: operatore di costo che promuove l'esplorazione dello spazio delle soluzioni

- $\hat{U}_M(\beta) = e^{-i\beta\hat{H}_M}$: operatore di mixing che codifica i vincoli e la funzione obiettivo

$$|\psi_p(\gamma, \beta)\rangle = e^{-i\beta_p\hat{H}_M}e^{-i\gamma_p\hat{H}_C}\ldots e^{-i\beta_1\hat{H}_M}e^{-i\gamma_1\hat{H}_C}|s\rangle$$

Algoritmi proposti (3/3)

Quantum Approximate Optimization Algorithm (QAOA)

- ***Algoritmo ibrido*** che opera attraverso una sequenza di layers (p) per approssimare la soluzione ottimale di un problema combinatorio

- $\hat{U}_C(\gamma) = e^{-i\gamma\hat{H}_C}$: operatore di costo che promuove l'esplorazione dello spazio delle soluzioni

- $\hat{U}_M(\beta) = e^{-i\beta\hat{H}_M}$: operatore di mixing che codifica i vincoli e la funzione obiettivo

$$|\psi_p(\gamma,\beta)\rangle = e^{-i\beta_p\hat{H}_M}e^{-i\gamma_p\hat{H}_C}\ldots e^{-i\beta_1\hat{H}_M}e^{-i\gamma_1\hat{H}_C}|s\rangle$$

- ***CPU:*** ottimizza i parametri β e γ per migliorare progressivamente il risultato

Algoritmi proposti (3/3)

Quantum Approximate Optimization Algorithm (QAOA)

- ***Algoritmo ibrido*** che opera attraverso una sequenza di layers (p) per approssimare la soluzione ottimale di un problema combinatorio
- $\hat{U}_C(\gamma) = e^{-i\gamma \hat{H}_C}$: operatore di costo che promuove l'esplorazione dello spazio delle soluzioni
- $\hat{U}_M(\beta) = e^{-i\beta \hat{H}_M}$: operatore di mixing che codifica i vincoli e la funzione obiettivo

$$|\psi_p(\gamma, \beta)\rangle = e^{-i\beta_p \hat{H}_M} e^{-i\gamma_p \hat{H}_C} \dots e^{-i\beta_1 \hat{H}_M} e^{-i\gamma_1 \hat{H}_C} |s\rangle$$

- ***CPU:*** ottimizza i parametri β e γ per migliorare progressivamente il risultato
- ***QPU:*** calcola la funzione obiettivo

Algoritmi proposti (3/3)

Quantum Approximate Optimization Algorithm (QAOA)

- ***Algoritmo ibrido*** che opera attraverso una sequenza di layers (p) per approssimare la soluzione ottimale di un problema combinatorio

- $\hat{U}_C(\gamma) = e^{-i\gamma\hat{H}_C}$: operatore di costo che promuove l'esplorazione dello spazio delle soluzioni

- $\hat{U}_M(\beta) = e^{-i\beta\hat{H}_M}$: operatore di mixing che codifica i vincoli e la funzione obiettivo

$$|\psi_p(\gamma,\beta)\rangle = e^{-i\beta_p\hat{H}_M}e^{-i\gamma_p\hat{H}_C}\ldots e^{-i\beta_1\hat{H}_M}e^{-i\gamma_1\hat{H}_C}|s\rangle$$

- ***CPU:*** ottimizza i parametri β e γ per migliorare progressivamente il risultato

- ***QPU:*** calcola la funzione obiettivo

$$F_p(\gamma,\beta) = \langle\psi_p(\gamma,\beta)|\hat{H}_C|\psi_p(\gamma,\beta)\rangle$$

Algoritmi proposti (3/3)

Quantum Approximate Optimization Algorithm (QAOA)

`quantum computer`

$$|0\rangle \longrightarrow \text{ansatz}(\gamma, \beta) \longrightarrow F(\gamma, \beta) = \langle \psi(\gamma, \beta) | H | \psi(\gamma, \beta) \rangle$$

$$(\gamma_{new}, \beta_{new}) \longleftarrow \text{optimization}(\gamma, \beta) \longleftarrow F(\gamma, \beta)$$

`classical computer`

Risoluzione del problema (1/6)

- ***Configurazione:***
  - Numero di asset: 8
  - Budget: 5
  - Rischio: 20%
  - Ripetizioni: 50

Risoluzione del problema (1/6)

- ***Configurazione:***
  - Numero di asset: 8
  - Budget: 5
  - Rischio: 20%
  - Ripetizioni: 50
- ***Metodologie utilizzate:***

Risoluzione del problema (1/6)

- ***Configurazione:***
  - Numero di asset: 8
  - Budget: 5
  - Rischio: 20%
  - Ripetizioni: 50

- ***Metodologie utilizzate:***
  - Simulazione <u>senza rumore</u> (**noiseless**)

Risoluzione del problema (1/6)

- ***Configurazione:***
  - Numero di asset: 8
  - Budget: 5
  - Rischio: 20%
  - Ripetizioni: 50

- ***Metodologie utilizzate:***
  - Simulazione <u>senza rumore</u> (**noiseless**)
  - Simulazione <u>con rumore</u> (**noisy**) per rappresentare hardware reale

Risoluzione del problema (2/6)

Configurazione dei dati

```python
dp = RandomDataProvider(
    # settings
).run()
stock_data = dp._data

mu = dp.get_period_return_mean_vector()
sigma = dp.get_period_return_covariance_matrix()

po = PortfolioOptimization(
    expected_returns=mu,
    covariances=sigma,
    risk_factor=risk_factor,
    budget=budget
)

qp = po.to_quadratic_program()
```

Risoluzione del problema (2/6)

Configurazione dei dati

1. Generati i dati dei rendimenti con la classe **RandomDataProvider**

```
dp = RandomDataProvider(
    # settings
).run()
stock_data = dp._data

mu = dp.get_period_return_mean_vector()
sigma = dp.get_period_return_covariance_matrix()

po = PortfolioOptimization(
    expected_returns=mu,
    covariances=sigma,
    risk_factor=risk_factor,
    budget=budget
)

qp = po.to_quadratic_program()
```

Risoluzione del problema (2/6)

Configurazione dei dati

1. Generati i dati dei rendimenti con la classe **RandomDataProvider**

2. Calcolati μ e Σ con i dati degli andamenti

```
dp = RandomDataProvider(
    # settings
).run()
stock_data = dp._data

mu = dp.get_period_return_mean_vector()
sigma = dp.get_period_return_covariance_matrix()

po = PortfolioOptimization(
    expected_returns=mu,
    covariances=sigma,
    risk_factor=risk_factor,
    budget=budget
)

qp = po.to_quadratic_program()
```

Risoluzione del problema (2/6)

Configurazione dei dati

1. Generati i dati dei rendimenti con la classe **RandomDataProvider**

2. Calcolati μ e Σ con i dati degli andamenti

3. Impostato il problema con la classe **PortfolioOptimization**

```
dp = RandomDataProvider(
    # settings
).run()
stock_data = dp._data

mu = dp.get_period_return_mean_vector()
sigma = dp.get_period_return_covariance_matrix()

po = PortfolioOptimization(
    expected_returns=mu,
    covariances=sigma,
    risk_factor=risk_factor,
    budget=budget
)

qp = po.to_quadratic_program()
```

Risoluzione del problema (2/6)

Configurazione dei dati

1. Generati i dati dei rendimenti con la classe **RandomDataProvider**
2. Calcolati μ e Σ con i dati degli andamenti
3. Impostato il problema con la classe **PortfolioOptimization**
4. Problema convertito in programma quadratico con **to_quadratic_program()**

```
dp = RandomDataProvider(
    # settings
).run()
stock_data = dp._data

mu = dp.get_period_return_mean_vector()
sigma = dp.get_period_return_covariance_matrix()

po = PortfolioOptimization(
    expected_returns=mu,
    covariances=sigma,
    risk_factor=risk_factor,
    budget=budget
)

qp = po.to_quadratic_program()
```

Risoluzione del problema (3/6)

Risoluzione classica

```
exact_mes = NumPyMinimumEigensolver()
exact_eigensolver = MinimumEigenOptimizer(exact_mes)
result_exact = exact_eigensolver.solve(qp)
```

Risoluzione del problema (3/6)

Risoluzione classica

1. Approccio basato su autovalori usando la classe **NumPyMinimumEigensolver**

```
exact_mes = NumPyMinimumEigensolver()
exact_eigensolver = MinimumEigenOptimizer(exact_mes)
result_exact = exact_eigensolver.solve(qp)
```

Risoluzione del problema (3/6)

Risoluzione classica

1. Approccio basato su autovalori usando la classe **NumPyMinimumEigensolver**

2. Utilizzo del wrapper **MinimumEigenOptimizer** per il supporto alla risoluzione dei problemi quadratici

```
exact_mes = NumPyMinimumEigensolver()
exact_eigensolver = MinimumEigenOptimizer(exact_mes)
result_exact = exact_eigensolver.solve(qp)
```

Risoluzione del problema (4/6)

Risoluzione con VQE

```python
ansatz = TwoLocal(
    num_qubits=assets,
    rotation_blocks="ry",
    entanglement_blocks="cz",
    reps=1,
    entanglement="full",
    insert_barriers=True
)

vqe_mes = SamplingVQE(sampler=Sampler(),
                      ansatz=ansatz,
                      optimizer=cobyla)

vqe = MinimumEigenOptimizer(vqe_mes, penalty)

result = vqe.solve(qp)
```

Risoluzione del problema (4/6)

Risoluzione con VQE

1. Generato ansatz con la classe **TwoLocal**

```
ansatz = TwoLocal(
    num_qubits=assets,
    rotation_blocks="ry",
    entanglement_blocks="cz",
    reps=1,
    entanglement="full",
    insert_barriers=True
)

vqe_mes = SamplingVQE(sampler=Sampler(),
                      ansatz=ansatz,
                      optimizer=cobyla)

vqe = MinimumEigenOptimizer(vqe_mes, penalty)

result = vqe.solve(qp)
```

Risoluzione del problema (4/6)

Risoluzione con VQE

1. Generato ansatz con la classe **TwoLocal**

2. Ansatz integrato nel metodo **SamplingVQE** con l'uso di un campionatore **Sampler()** e l'ottimizzatore **cobyla** per la minimizzazione dei parametri

```
ansatz = TwoLocal(
    num_qubits=assets,
    rotation_blocks="ry",
    entanglement_blocks="cz",
    reps=1,
    entanglement="full",
    insert_barriers=True
)

vqe_mes = SamplingVQE(sampler=Sampler(),
                      ansatz=ansatz,
                      optimizer=cobyla)

vqe = MinimumEigenOptimizer(vqe_mes, penalty)

result = vqe.solve(qp)
```

Risoluzione del problema (4/6)

Risoluzione con VQE

1. Generato ansatz con la classe **TwoLocal**

2. Ansatz integrato nel metodo **SamplingVQE** con l'uso di un campionatore **Sampler()** e l'ottimizzatore **cobyla** per la minimizzazione dei parametri

3. Utilizzo del wrapper **MinimumEigenOptimizer**

```
ansatz = TwoLocal(
    num_qubits=assets,
    rotation_blocks="ry",
    entanglement_blocks="cz",
    reps=1,
    entanglement="full",
    insert_barriers=True
)

vqe_mes = SamplingVQE(sampler=Sampler(),
                      ansatz=ansatz,
                      optimizer=cobyla)

vqe = MinimumEigenOptimizer(vqe_mes, penalty)

result = vqe.solve(qp)
```

Risoluzione del problema (5/6)

Risoluzione con QAOA

```
qaoa_mes = QAOA(sampler=Sampler(),
                optimizer=cobyla,
                reps=1)

qaoa = MinimumEigenOptimizer(qaoa_mes, penalty)

result = qaoa.solve(qp)
```

Risoluzione del problema (5/6)

Risoluzione con QAOA

1. Circuito generato con la classe **QAOA** utilizzando il campionatore **Sampler()** e l'ottimizzatore **cobyla**

```
qaoa_mes = QAOA(sampler=Sampler(),
                optimizer=cobyla,
                reps=1)

qaoa = MinimumEigenOptimizer(qaoa_mes, penalty)

result = qaoa.solve(qp)
```

Risoluzione del problema (5/6)

Risoluzione con QAOA

1. Circuito generato con la classe **QAOA** utilizzando il campionatore **Sampler()** e l'ottimizzatore **cobyla**

2. Utilizzo del wrapper **MinimumEigenOptimizer**

```
qaoa_mes = QAOA(sampler=Sampler(),
                optimizer=cobyla,
                reps=1)

qaoa = MinimumEigenOptimizer(qaoa_mes, penalty)

result = qaoa.solve(qp)
```

Risoluzione del problema (6/6)

Introduzione del rumore

```python
backend = GenericBackendV2(num_qubits=assets,
                           noise_info=True,
                           seed=seed,
                           calibrate_instructions=True)

# noise settings

noise_model = NoiseModel.from_backend(backend)

simulator = AerSimulator(noise_model=noise_model)
```

Risoluzione del problema (6/6)

Introduzione del rumore

1. Generato backend con la classe **GenericBackendV2**

```
backend = GenericBackendV2(num_qubits=assets,
                           noise_info=True,
                           seed=seed,
                           calibrate_instructions=True)

# noise settings

noise_model = NoiseModel.from_backend(backend)

simulator = AerSimulator(noise_model=noise_model)
```

Risoluzione del problema (6/6)

Introduzione del rumore

1. Generato backend con la classe **GenericBackendV2**

2. Generato modello di rumore con la classe **NoiseModel**

```
backend = GenericBackendV2(num_qubits=assets,
                           noise_info=True,
                           seed=seed,
                           calibrate_instructions=True)

# noise settings

noise_model = NoiseModel.from_backend(backend)

simulator = AerSimulator(noise_model=noise_model)
```

Risoluzione del problema (6/6)

Introduzione del rumore

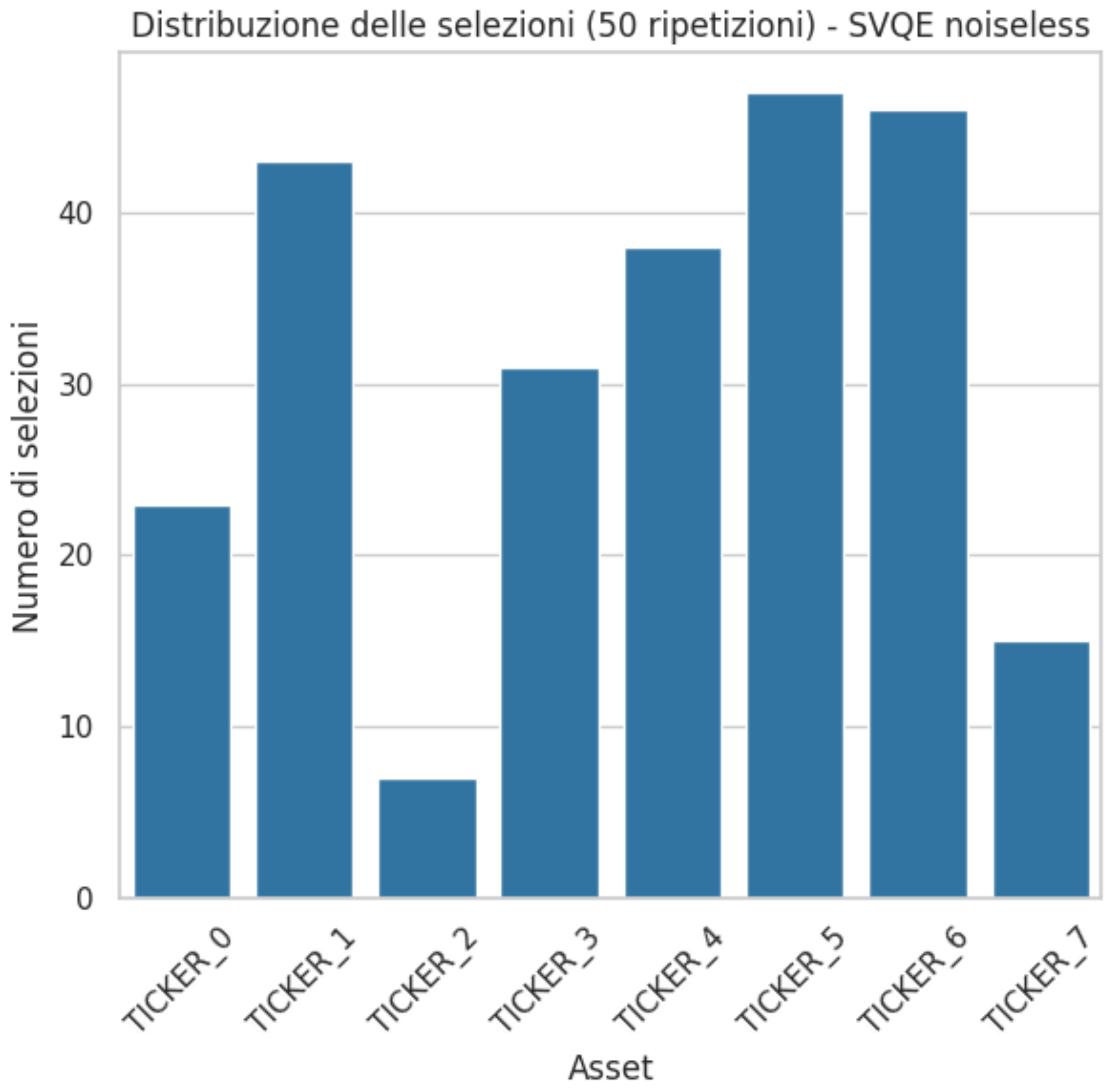
1. Generato backend con la classe **GenericBackendV2**

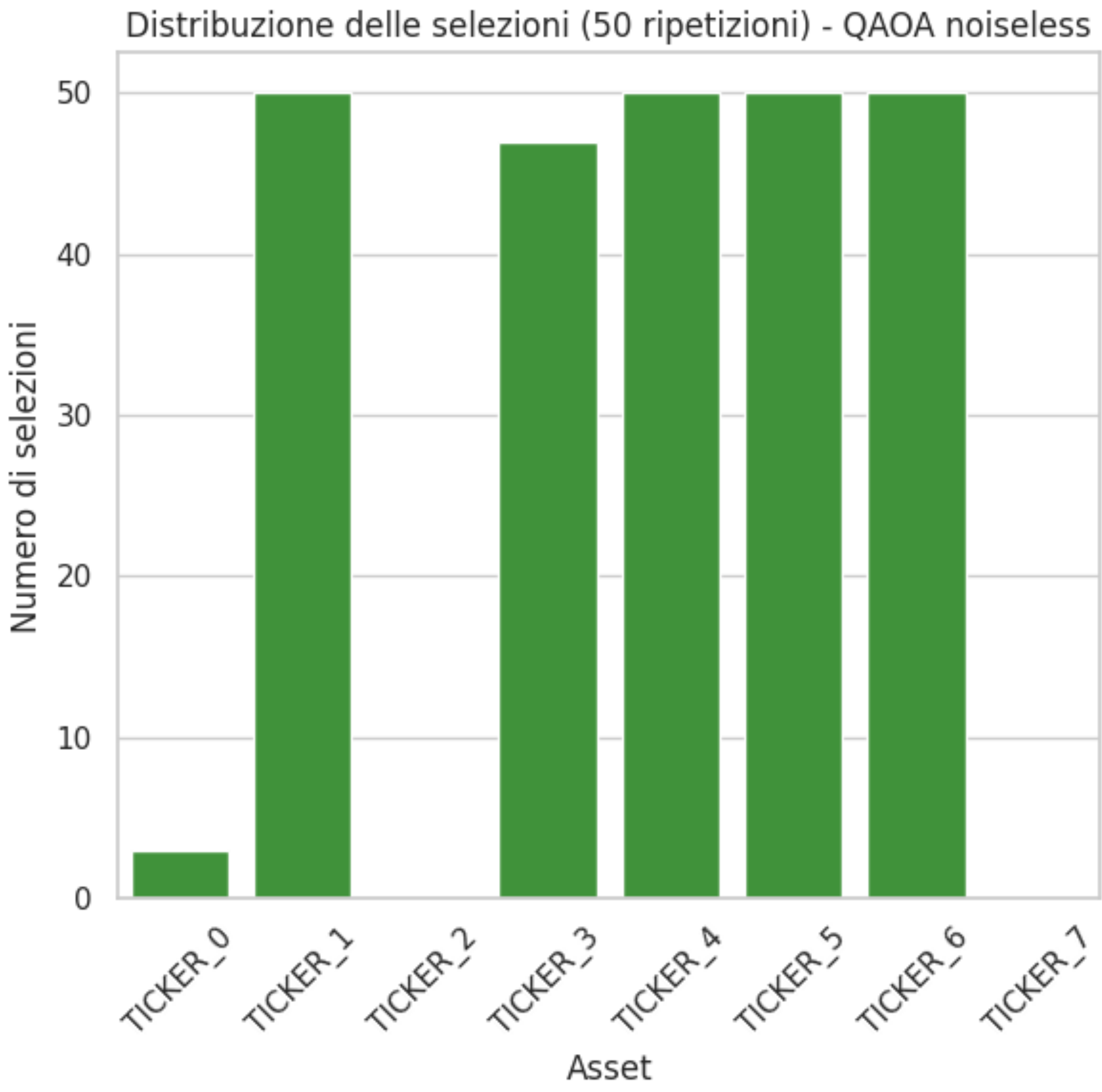
2. Generato modello di rumore con la classe **NoiseModel**

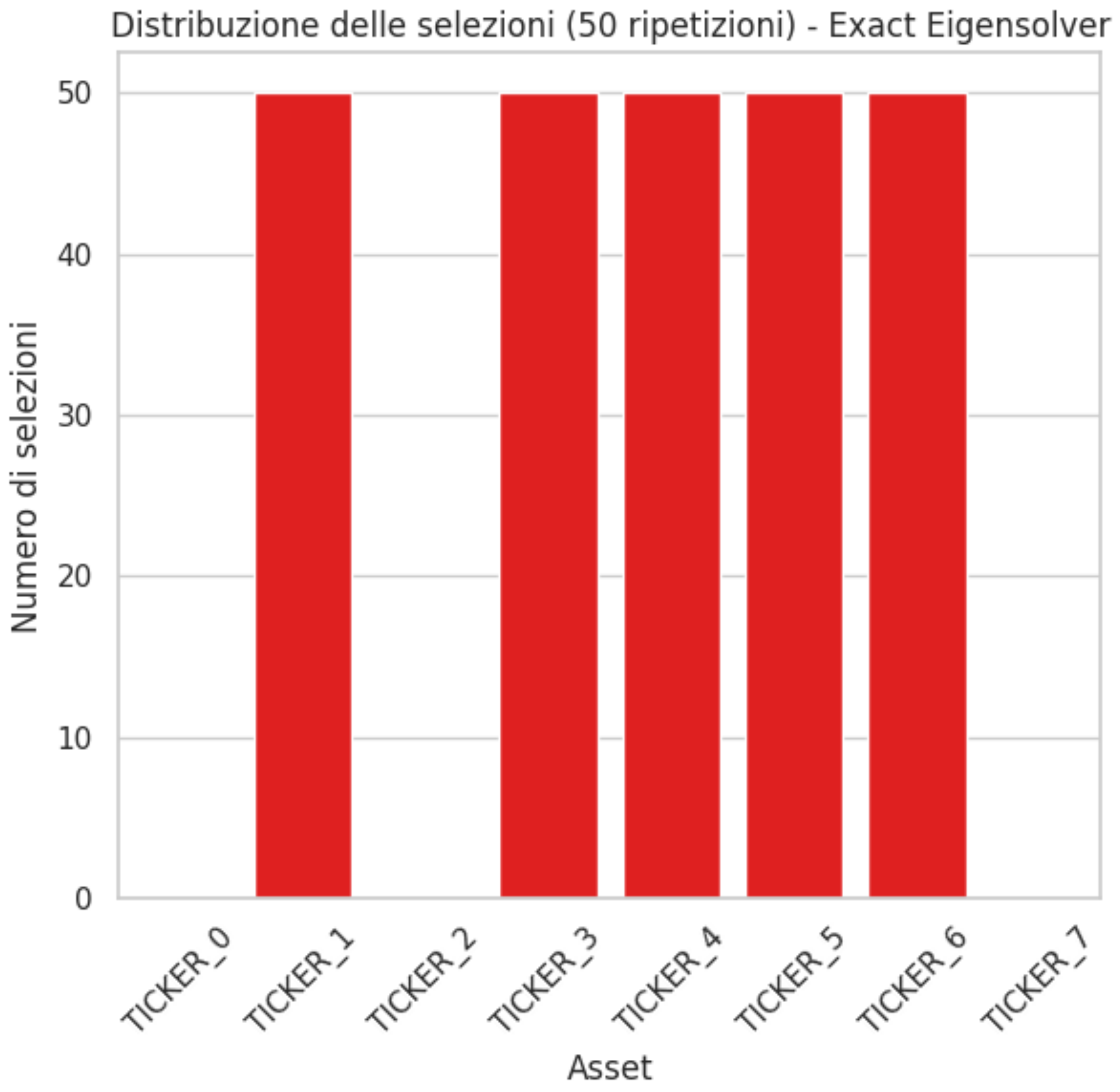
3. Generato simulatore con rumore con la classe **AerSimulator**

```
backend = GenericBackendV2(num_qubits=assets,
                           noise_info=True,
                           seed=seed,
                           calibrate_instructions=True)

# noise settings

noise_model = NoiseModel.from_backend(backend)

simulator = AerSimulator(noise_model=noise_model)
```

Risultati senza rumore (noiseless)

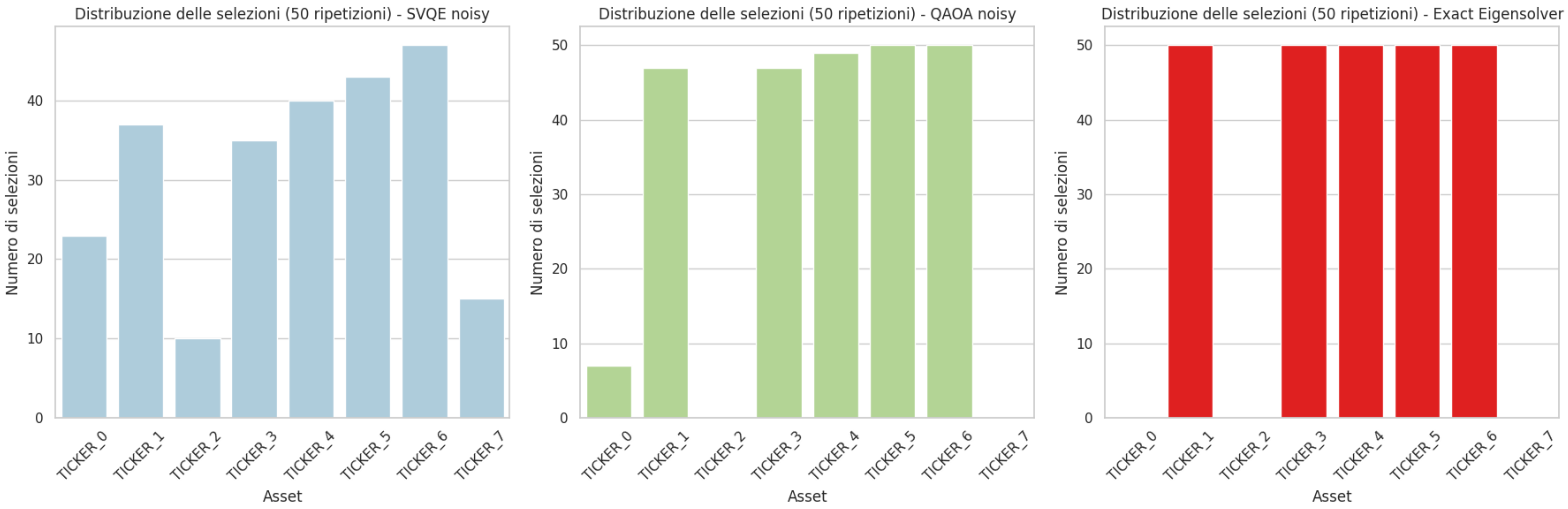






- Il QAOA tende a concentrarsi su configurazioni ristrette
- Il VQE esplora con maggiore diversificazione

Risultati con rumore (noisy)



- Diminuzione della precisione per VQE e QAOA
- Incremento dell'incertezza nelle distribuzioni degli asset selezionati

Conclusioni

- L'approccio quantistico offre un ***potenziale significativo*** per problemi complessi come l'ottimizzazione del portafoglio

Conclusioni

- L'approccio quantistico offre un ***potenziale significativo*** per problemi complessi come l'ottimizzazione del portafoglio
- ***Limiti attuali della tecnologia quantistica:***

Conclusioni

- L'approccio quantistico offre un ***potenziale significativo*** per problemi complessi come l'ottimizzazione del portafoglio

- ***Limiti attuali della tecnologia quantistica:***

    - Problemi su larga scala, come il portafoglio con migliaia di qubit, rimangono **impraticabili**

Conclusioni

- L'approccio quantistico offre un ***potenziale significativo*** per problemi complessi come l'ottimizzazione del portafoglio
- ***Limiti attuali della tecnologia quantistica:***
  - Problemi su larga scala, come il portafoglio con migliaia di qubit, rimangono **impraticabili**
  - Il rumore **compromette la qualità** delle soluzioni proposte, rendendo inefficace l'approccio quantistico

Q&A

Quantum Portfolio Optimization

Corso di Quantum Computing (a.a. 2024/25)

Merenda Saverio Mattia

UNIVERSITÀ DI PARMA

Bibliografia

1. *Land, Ailsa H and Doig, Alison G (2010). An automatic method for solving discrete programming problems, Springer.*

2. *Blekos, Kostas et al., (2024). "A review on quantum approximate optimization algorithm and its variants", Physics Reports, Vol. 1068, pp. 1–66.*

3. *Buonaiuto, Giuseppe et al., (2023). "Best practices for portfolio optimization by quantum computing, experimented on real quantum devices", Scientific Reports, Vol. 13 No. 1, p. 19434.*

4. *Qiskit (2024). Portfolio Optimization using Qiskit Finance, qiskit-community.github.io/qiskit-finance/tutorials/portfolio-optimization*