| Framework | Domain | Mathematical Formulation | Python Sample | Explanation | Quadratic or Others? | Relevant Quantum/Classical Algorithm |
|---|---|---|---|---|---|---|
| Modern Portfolio Theory | Finance, Investment | \max \sum_{i} \sum_{j} iv[i] \cdot iv[j] \cdot \text{prices}[i] \cdot \text{prices}[j] \cdot \text{cov}[i,j] - \sum_{i} iv[i] \cdot \text{prices}[i] \cdot \text{mean}[i] , | set_objective(quicksum(quicksum(iv[i]*iv[j]*prices[i]*prices[j]*cov[i,j] for j in range(N)) for i in range(N)) - quicksum(iv[i]*prices[i]*mean[i] for i in range(N))) | iv[i]: Investment proportion (weight) of asset i, prices[i]: Price of asset i, cov[i,j]: Covariance between assets i and j, | Quadratic | Mean-Variance Optimization |
| Traveling Salesman Problem | Logistics, Graph Theory | \min \sum_{i} \sum_{j} d_{i,j} \cdot x_{i,j} , | set_objective(quicksum(quicksum(distance_matrix[i][j] * x[i][j] for j in range(N)) for i in range(N))) | d[i,j]: Distance between cities i and j, x[i,j]: Binary decision variable for visiting city i after city j, | Integer Linear | Held-Karp Algorithm (Classical), Quantum Approximate Optimization Algorithm (QAOA) |
| Economic Order Quantity | Manufacturing, Retail, CPG | \min \sqrt{\frac{2 \cdot D \cdot S}{H}} , | set_objective(math.sqrt(2 * D * S / H)) | D: Demand rate, S: Ordering cost, H: Holding cost per unit per year, | Non-quadratic | Quantum Approximate Optimization Algorithm (QAOA) |
| Vehicle Routing Problem | Logistics, Transportation | \min \sum_{i} \sum_{j} d_{i,j} \cdot x_{i,j} , | set_objective(quicksum(quicksum(distance_matrix[i][j] * x[i][j] for j in range(N)) for i in range(N))) | d[i,j]: Distance between customers i and j, x[i,j]: Binary decision variable for serving customer i after customer j, | Integer Linear | Quantum Approximate Optimization Algorithm (QAOA) |
| Knapsack Problem | Retail, CPG | \max \sum_{i} v_i \cdot x_i , | set_objective(quicksum(value[i] * x[i] for i in range(N))) | v[i]: Value of item i, x[i]: Binary decision variable for selecting item i, | Non-quadratic | Grover's Search Algorithm (Quantum) |
| Network Flow Problem | Telecommunication, Transport | \max \sum_{i} \sum_{j} f_{i,j} , | set_objective(quicksum(quicksum(flow[i][j] for j in range(N)) for i in range(N))) | f[i,j]: Flow from node i to node j, | Non-quadratic | Classical Network Flow Algorithms |
| Hospital Staff Scheduling | Healthcare | \min \sum_{i} \sum_{j} c_{i,j} \cdot x_{i,j} , | set_objective(quicksum(quicksum(cost_matrix[i][j] * x[i][j] for j in range(N)) for i in range(N))) | c[i,j]: Cost of assigning staff i to shift j, x[i,j]: Binary decision variable for assigning staff i to shift j, | Quadratic | Classical Scheduling Algorithms |
| Quadratic Assignment Problem | Optimization | \min \sum_{i} \sum_{j} a_{i,j} \cdot b_{\pi(i), \pi(j)} , | set_objective(quicksum(quicksum(a[i][j] * b[pi[i]][pi[j]] for j in range(N)) for i in range(N))) | a[i,j]: Cost of assigning item i to location j, b[pi[i], pi[j]]: Flow between locations pi(i) and pi(j), | Quadratic | Quantum Approximate Optimization Algorithm (QAOA) |
| Maximum Cut Problem | Combinatorial Optimization | \max \sum_{i} \sum_{j} w_{i,j} \cdot x_{i,j} , | set_objective(quicksum(quicksum(weight_matrix[i][j] * x[i][j] for j in range(N)) for i in range(N))) | w[i,j]: Weight of edge between nodes i and j, x[i,j]: Binary decision variable indicating if edge i,j is cut, | Quadratic | Quantum Approximate Optimization Algorithm (QAOA) |
| Set Cover Problem | Combinatorial Optimization | \min \sum_{i} c_i \cdot y_i , | set_objective(quicksum(cost[i] * y[i] for i in range(N))) | c[i]: Cost of selecting set i, y[i]: Binary decision variable indicating if set i is chosen, | Non-quadratic | Greedy Algorithms, Integer Linear Programming |
| Linear Regression | Machine Learning | \min \sum_{i} (y_i - \sum_{j} X_{i, j} \cdot \beta_j)^2 , | set_objective(quicksum((y[i] - quicksum(X[i][j] * beta[j] for j in range(M)))**2 for i in range(N))) | y[i]: Observed output value for sample i, X[i,j]: Feature value of sample i for feature j, beta[j]: Model coefficient for feature j, | Quadratic | Classical Least Squares Regression |
| Maximum Independent Set | Combinatorial Optimization | \max \sum_{i} x_i , | set_objective(quicksum(x[i] for i in range(N))) | x[i]: Binary decision variable indicating if node i is part of the independent set, | Non-quadratic | Quantum Approximate Optimization Algorithm (QAOA) |
| Facility Location Problem | Combinatorial Optimization | \min \sum_{i} \sum_{j} f_{i,j} \cdot x_i , | set_objective(quicksum(quicksum(facility_cost[i][j] * x[i] for j in range(N)) for i in range(N))) | f[i,j]: Cost of opening facility i with respect to demand center j, x[i]: Binary decision variable indicating if facility i is opened, | Quadratic | Quantum Approximate Optimization Algorithm (QAOA) |

| | | | | | | |
|---|---|---|---|---|---|---|
| Job Shop Scheduling | Manufacturing, Production | \min \sum_{i} \sum_{j} \sum_{k} c_{i,j,k} \cdot x_{i,j,k} , | set_objective(quicksum(quicksum(quicksum(cost_matrix[i][j][k] * x[i][j][k] for k in range(K)) for j in range(N)) for i in range(N))) | c[i,j,k]: Cost of assigning job i to machine j at time step k, x[i,j,k]: Binary decision variable for scheduling job i on machine j at time step k, | Quadratic | Quantum Approximate Optimization Algorithm (QAOA) |
| Bin Packing Problem | Combinatorial Optimization | \min \sum_{i} y_i , | set_objective(quicksum(y[i] for i in range(N))) | y[i]: Binary decision variable indicating if bin i is used, | Non-quadratic | Greedy Algorithms, Integer Linear Programming |
| Set Partitioning Problem | Combinatorial Optimization | \min \sum_{i} c_i \cdot y_i , | set_objective(quicksum(cost[i] * y[i] for i in range(N))) | c[i]: Cost of selecting set i, y[i]: Binary decision variable indicating if set i is chosen, | Non-quadratic | Integer Linear Programming |
| Minimum Vertex Cover | Combinatorial Optimization | \min \sum_{i} x_i , | set_objective(quicksum(x[i] for i in range(N))) | x[i]: Binary decision variable indicating if node i is part of the vertex cover, | Non-quadratic | Integer Linear Programming |
| Graph Coloring Problem | Combinatorial Optimization | \min \sum_{i} \sum_{j} c_{i,j} \cdot x_i \cdot x_j , | set_objective(quicksum(quicksum(coloring_cost[i][j] * x[i] * x[j] for j in range(N)) for i in range(N))) | c[i,j]: Cost of adjacent nodes i and j having the same color, x[i]: Binary decision variable indicating if node i is colored, | Quadratic | Quantum Approximate Optimization Algorithm (QAOA) |
| Capacitated Vehicle Routing | Logistics, Transportation | \min \sum_{i} \sum_{j} d_{i,j} \cdot x_{i,j} , | set_objective(quicksum(quicksum(distance_matrix[i][j] * x[i][j] for j in range(N)) for i in range(N))) | d[i,j]: Distance between customers i and j, x[i,j]: Binary decision variable for serving customer i after customer j, | Integer Linear | Quantum Approximate Optimization Algorithm (QAOA) |
| Quadratic Knapsack Problem | Combinatorial Optimization | \max \sum_{i} \sum_{j} v_{i,j} \cdot x_i \cdot x_j , | set_objective(quicksum(quicksum(value_matrix[i][j] * x[i] * x[j] for j in range(N)) for i in range(N))) | v[i,j]: Value of item i when combined with item j, x[i]: Binary decision variable for selecting item i, | Quadratic | Quantum Approximate Optimization Algorithm (QAOA) |
| Quadratic Assignment Problem | Optimization | \min \sum_{i} \sum_{j} a_{i,j} \cdot b_{\pi(i), \pi(j)} | set_objective(quicksum(quicksum(a[i][j] * b[pi[i]][pi[j]] for j in range(N)) for i in range(N))) | a[i,j]: Cost of assigning item i to location j, b[pi[i], pi[j]]: Flow between locations pi(i) and pi(j) | Quadratic | Quantum Approximate Optimization Algorithm (QAOA) |