# Optimisation Algorithms and Their Real-World Applications

Stuti Malik

March 2025

## Contents

**1 Introduction** **3**
1.1 Examples of Optimisation Applications . . . 3
1.2 Challenges and Trade-offs in Optimisation . . . 4

**2 What Are Optimisation Algorithms?** **5**
2.1 Challenges in Optimisation . . . 6
2.2 Recent Advancements in Optimisation Algorithms . . . 6

**3 Types of Optimisation Algorithms** **7**
3.1 Gradient-Based Optimisation . . . 7
3.2 Metaheuristic Algorithms . . . 7
3.3 Linear and Non-Linear Programming . . . 8
3.4 Dynamic Optimisation . . . 8
3.5 Convex Optimisation . . . 8
3.6 Quadratic Programming . . . 9
3.7 Stochastic Optimisation . . . 9
3.8 Network Optimisation . . . 9
3.9 Integer Programming . . . 9
3.10 Constraint Programming . . . 9
3.11 Robust Optimisation . . . 10
3.12 Global Optimisation . . . 10
3.13 Surrogate-Based Optimisation . . . 10
3.14 Other Advanced Optimisation Methods . . . 11

**4 Applications in Key Industries** **12**
4.1 Healthcare . . . 12
4.2 Marketing and Advertising . . . 12
4.3 Energy and Sustainability . . . 13
4.4 Transportation and Logistics . . . 13
4.5 Finance and Investment . . . 14

**5 Challenges and Future of Optimisation** **15**
5.1 Future Directions . . . 15

**6 Conclusion** **17**
6.1 Further Questions to Explore . . . . . 17

# 1 Introduction

Optimisation algorithms are essential tools for solving complex problems across various industries, including healthcare, finance, logistics, and artificial intelligence. These algorithms seek to identify the most efficient solutions to a given problem while satisfying a set of constraints. The goal is to find the optimal value of an objective function, which is typically a mathematical representation of the system's performance.

In mathematical terms, an optimisation problem is generally formulated as:

$$\min_{x \in R^n} f(x) \quad \text{subject to} \quad g_i(x) \leq 0, \quad h_j(x) = 0, \tag{1}$$

where $f(x)$ is the objective function to be minimised, $g_i(x)$ represents inequality constraints (such as limits on resources or performance), and $h_j(x)$ represents equality constraints (such as certain conditions that must hold true).

In practice, optimisation algorithms address critical trade-offs that exist in many real-world problems, such as cost versus efficiency, speed versus accuracy, and resource allocation under uncertainty. The complexity of real-world applications often involves making decisions based on a large set of variables and constraints. These decisions need to balance conflicting objectives to find the best possible outcome.

## 1.1 Examples of Optimisation Applications

- **Logistics and Supply Chain Optimisation**: In the logistics industry, optimisation algorithms are used to reduce transportation costs and minimise delivery times. For example, companies use *vehicle routing algorithms* to determine the most efficient routes for delivery trucks. The objective is to minimise the total distance travelled while satisfying constraints such as delivery windows and vehicle capacity. A well-known example is the *Travelling Salesman Problem* (TSP), where the goal is to find the shortest route that visits a set of cities exactly once and returns to the starting point.

- **Portfolio Optimisation in Finance**: Portfolio optimisation techniques in finance aim to allocate assets in a way that maximises returns while minimising risk. One common approach is the *Markowitz Efficient Frontier*, which balances the trade-off between risk (variance) and return in investment portfolios. The objective function in this case would be to minimise the portfolio's variance for a given level of expected return, subject to constraints such as the total investment sum and individual asset limits.

- **Healthcare Optimisation**: In healthcare, optimisation algorithms are used to create efficient treatment plans for patients. For instance, in radiation therapy, the goal is to deliver the maximum dose of radiation to cancerous cells while minimising damage to surrounding healthy tissue. This is a typical example of a constrained optimisation problem where the objective function is to maximise the treatment's effectiveness, subject to constraints that ensure patient safety and treatment feasibility.

- **Artificial Intelligence and Machine Learning**: In machine learning, optimisation plays a crucial role in model training. Algorithms like gradient descent are used to minimise the loss function (or error) by adjusting model parameters iteratively. For example, when training a neural network, the optimisation algorithm seeks to minimise the error between the predicted and actual values by updating the weights of the network. The constraints here may be in the form of model architecture, computational resources, or overfitting prevention techniques.

## 1.2 Challenges and Trade-offs in Optimisation

While optimisation techniques provide powerful tools for decision-making, the process often involves significant challenges. One of the main challenges is dealing with non-linearities and large, complex solution spaces that make finding an optimal solution computationally expensive. For example, the problem of *global optimisation* requires searching for the best solution in a potentially vast and highly irregular solution space. Moreover, *multi-objective optimisation* problems, where more than one objective must be optimised simultaneously, present further complexity. These issues require sophisticated algorithms, such as *genetic algorithms* and *simulated annealing*, which are capable of finding near-optimal solutions in large, complex solution spaces.

Another trade-off encountered in real-world applications is the balance between solution accuracy and computational efficiency. In some cases, achieving a perfectly optimal solution may not be feasible due to time or resource constraints. In these cases, *approximation algorithms* may be employed to find good, albeit suboptimal, solutions that can be computed within a reasonable timeframe.

## Conclusion

Optimisation algorithms play a crucial role in improving decision-making and driving innovation across various sectors. By balancing competing objectives and managing constraints, they enable more efficient and effective problem-solving. The increasing complexity of modern problems, combined with advancements in computational power, continues to drive the development of more sophisticated optimisation techniques. This article provides an overview of optimisation techniques, their classifications, and their real-world applications, demonstrating the pivotal role of optimisation in solving complex, multi-dimensional problems.

# 2 What Are Optimisation Algorithms?

Optimisation is the process of finding the best possible solution from a set of feasible options, given an objective function and constraints. Mathematically, a general optimisation problem is expressed as:

$$\min_{x \in S} f(x), \tag{2}$$

where $f(x)$ represents the objective function to be minimised (or maximised), and $S$ denotes the feasible solution space. The goal is to determine the value of $x$ that either minimises or maximises $f(x)$, subject to the constraints within $S$.
Optimisation problems can be classified into different types based on the nature of the objective function, constraints, and the problem structure. The main classifications include:

- **Linear vs. Non-Linear Optimisation:** In linear optimisation, both the objective function and constraints are linear functions. For example, in linear programming, the objective might be to maximise profit subject to resource constraints. Non-linear optimisation, on the other hand, involves at least one non-linear function, which makes the problem more complex to solve. An example is portfolio optimisation, where the objective is to maximise return while minimising risk, and both return and risk are non-linear functions.

- **Convex vs. Non-Convex Optimisation:** Convex optimisation problems guarantee a global optimum due to the shape of the objective function and constraints. This is useful in scenarios like resource allocation where a single optimal solution exists. Non-convex optimisation problems, however, may contain multiple local optima, making it challenging to find the global optimum. An example of a non-convex problem is training deep neural networks, where multiple local minima can exist in the loss function.

- **Single-Objective vs. Multi-Objective Optimisation:** In single-objective optimisation, the goal is to optimise a single objective, such as minimising cost or maximising efficiency. For instance, a manufacturing company may seek to minimise production costs. Multi-objective optimisation, in contrast, involves multiple, often conflicting objectives. A common example is the trade-off between cost and quality in product development, where improving quality might increase production costs, and the challenge is to find a balance between the two.

Each type of optimisation problem requires specific algorithms and techniques to solve efficiently. For example:

- **Linear Optimisation** is typically solved using the Simplex algorithm or Interior Point methods.

- **Non-Linear Optimisation** often employs gradient-based methods, such as Gradient Descent or Newton's Method. These methods rely on calculating the gradient of the objective function to iteratively adjust the solution.

- **Multi-Objective Optimisation** commonly uses algorithms like the Non-dominated Sorting Genetic Algorithm II (NSGA-II) for Pareto optimal solutions. In multi-objective problems, the goal is to find a set of solutions that balances all objectives without one being strictly better than the others in all respects.

These techniques are tailored to handle the characteristics and complexities of different optimisation problems, ensuring efficient problem-solving in various fields such as logistics, finance, and machine learning.

## 2.1 Challenges in Optimisation

Despite the power of optimisation algorithms, several challenges arise when dealing with real-world problems:

- **Scalability:** As the number of variables and constraints increases, the solution space becomes exponentially larger, making optimisation algorithms more computationally expensive and time-consuming.
- **Accuracy vs. Computation Time:** In many real-world applications, finding the exact optimal solution may not be feasible due to time or resource constraints. Approximation algorithms may be used to generate near-optimal solutions in a reasonable timeframe.
- **Constraints Handling:** Complex constraints, such as non-linear or dynamic constraints, require more sophisticated methods to incorporate them into the solution process.

## 2.2 Recent Advancements in Optimisation Algorithms

Optimisation techniques have evolved with advancements in computing power and the need to solve increasingly complex problems. Recent developments include:

- **Quantum Optimisation:** Using quantum computing to solve optimisation problems with potentially significant improvements in speed and efficiency, particularly in complex multi-objective and non-convex problems.
- **Machine Learning in Optimisation:** Integration of machine learning techniques, such as reinforcement learning, into optimisation to adaptively find optimal solutions in dynamic environments.
- **Metaheuristics and Evolutionary Algorithms:** Advances in algorithms like Genetic Algorithms, Simulated Annealing, and Particle Swarm Optimisation to handle large-scale and non-linear optimisation problems.

As optimisation problems continue to grow in complexity, these innovations help improve both the speed and quality of solutions.

# 3 Types of Optimisation Algorithms

Different optimisation techniques are suited to different problem domains. Below, we outline major categories of optimisation algorithms, along with relevant examples.

## 3.1 Gradient-Based Optimisation

Gradient-based methods rely on the derivative of the objective function to iteratively improve the solution. These methods are particularly useful for problems where the objective function is differentiable. Common techniques include:

- **Gradient Descent,** a widely used method in machine learning for minimising loss functions. It iteratively adjusts the parameters in the direction of the negative gradient to find a local minimum.
- **Newton's Method,** which uses second-order derivatives to improve convergence speed. It incorporates the Hessian matrix, making it more computationally expensive but more efficient in some cases.

For example, in training deep neural networks, *Stochastic Gradient Descent* (SGD) is employed to update the weights iteratively:

$$w_{t+1} = w_t - \eta \nabla f(w_t), \tag{3}$$

where $w_t$ is the weight at time $t$, $\eta$ is the learning rate, and $\nabla f(w_t)$ is the gradient of the objective function at $w_t$.

## 3.2 Metaheuristic Algorithms

Metaheuristics are approximate methods for solving complex, non-convex problems that may contain multiple local minima. These algorithms are often used when traditional methods struggle to find a global optimum. Examples include:

- **Genetic Algorithms,** inspired by the principles of natural selection, are often used in scheduling and game AI. They use populations of solutions that evolve over time through selection, crossover, and mutation.
- **Simulated Annealing,** based on the metal cooling process, allows for occasional moves to worse solutions in order to escape local minima and explore the solution space more thoroughly.
- **Particle Swarm Optimisation,** inspired by the social behaviour of birds and fish, is commonly used in robotics and supply chain management. It updates the position of particles in the solution space based on both their own experiences and the experiences of other particles.

## 3.3 Linear and Non-Linear Programming

Optimisation problems can also be classified based on the structure of the objective function and constraints. Two major types are:

- **Linear Programming (LP),** used for problems where the objective function and constraints are linear. A common application is resource allocation, where the Simplex Method is used to solve problems like production planning.
- **Non-Linear Programming (NLP),** in which at least one of the objective functions or constraints is non-linear. This is typically used in more complex scenarios, such as portfolio optimisation and risk assessment, where the relationships between variables are not linear.

## 3.4 Dynamic Optimisation

Dynamic optimisation addresses problems where decisions are made sequentially over time, often under uncertainty. It is widely used in areas such as reinforcement learning and optimal control. Key methods include:

- **Markov Decision Processes (MDPs),** used in reinforcement learning to model decision-making under uncertainty. In MDPs, the future state depends only on the current state and action, not on previous states.
- **Bellman's Equation,** a fundamental concept in dynamic programming and optimal control problems. It provides a recursive decomposition of the value function, which helps in finding the optimal policy. The Bellman Equation is expressed as:

$$V(s) = \max_a \left[ R(s,a) + \gamma \sum P(s'|s,a) V(s') \right], \tag{4}$$

  where $V(s)$ is the value function at state $s$, $R(s,a)$ is the reward for taking action $a$ in state $s$, $P(s'|s,a)$ is the probability of transitioning to state $s'$ given action $a$, and $\gamma$ is the discount factor.

## 3.5 Convex Optimisation

Convex optimisation problems involve an objective function that is convex, and the feasible region is also convex. These types of problems guarantee a global optimum, making them attractive in practical scenarios. Common techniques for convex optimisation include:

- **Interior Point Methods,** used for large-scale convex problems such as linear and non-linear programming. These methods are efficient for solving convex optimisation problems with many variables.
- **Gradient Projection Methods,** employed when the problem involves constraints and is usually applied to constrained convex problems.

## 3.6 Quadratic Programming

Quadratic programming is a special case of non-linear programming in which the objective function is quadratic, and the constraints are linear. This is frequently used in portfolio optimisation and risk management, where risk is represented as the variance or covariance of asset returns.

## 3.7 Stochastic Optimisation

Stochastic optimisation deals with problems that have uncertainty or randomness. Techniques include:

- **Stochastic Gradient Descent (SGD),** an approximation of the gradient descent method that works by calculating the gradient using a random subset of the data, allowing for faster convergence in large datasets.
- **Monte Carlo Methods,** which use random sampling to solve problems that may be deterministic in nature but require estimation due to uncertainty.

## 3.8 Network Optimisation

Network optimisation focuses on solving problems related to routing, flow, and resource allocation in networks. Techniques include:

- **Shortest Path Algorithms,** such as Dijkstra's algorithm, which finds the shortest path between two nodes in a graph. This is widely used in routing for network optimisation.
- **Maximum Flow Problems,** where the goal is to find the maximum flow through a network, which can be applied to resource allocation in logistics networks.

## 3.9 Integer Programming

Integer programming is a special case of linear and non-linear programming where some or all of the decision variables are restricted to integer values. This is commonly used in problems like scheduling, logistics, and resource allocation.

- **Mixed-Integer Linear Programming (MILP),** where some variables are integers and others are continuous. This is widely used in operational research problems such as production planning, vehicle routing, and network design.
- **Binary Integer Programming,** where the decision variables are restricted to binary values (0 or 1). This is often used in problems like facility location and equipment selection.

## 3.10 Constraint Programming

Constraint programming focuses on solving combinatorial problems by specifying a set of constraints that the solution must satisfy. It is particularly useful for scheduling, timetabling, and other combinatorial optimisation problems.

- **Backtracking,** an algorithmic approach used to find solutions by systematically exploring the possible configurations while pruning infeasible solutions early. This is applied in solving puzzles and scheduling.
- **Local Search,** methods that start with an initial solution and iteratively explore neighbouring solutions to find the optimal one, while considering the constraints. This is applied to scheduling problems where finding the best solution is challenging.

## 3.11 Robust Optimisation

Robust optimisation deals with optimisation problems under uncertainty, focusing on finding solutions that are feasible and optimal in a worst-case scenario, where some parameters are subject to uncertainty or variability.

- **Robust Linear Programming,** which extends linear programming to handle uncertain parameters, ensuring that the solution is feasible for all possible values within specified bounds. This is applied in production scheduling when demand is uncertain.
- **Robust Portfolio Optimisation,** where the goal is to construct an investment portfolio that performs well under various uncertain market conditions. This technique is often used in financial investment strategies.

## 3.12 Global Optimisation

Global optimisation techniques are designed to find the global optimum of a function, especially when the function is non-convex and contains multiple local minima or maxima.

- **Branch and Bound,** an exact method used to solve global optimisation problems by systematically dividing the problem into smaller subproblems and bounding the solution space. This is often used in integer programming problems.
- **Interval Analysis,** used to find the global minimum or maximum of a function over a specified range by successively narrowing down the interval of possible solutions. This method is applied in real-world applications such as engineering design.

## 3.13 Surrogate-Based Optimisation

Surrogate-based optimisation uses surrogate models (approximations of the true objective function) to guide the search for the optimum, particularly in expensive optimisation tasks such as engineering design, where evaluating the objective function is computationally expensive.

- **Kriging (Gaussian Process Optimisation),** a popular surrogate model used in global optimisation problems where the objective function is expensive to evaluate. It is commonly used in material science for optimisation of designs.
- **Radial Basis Function (RBF),** another type of surrogate model often used in engineering design and complex simulation-based optimisation.

## 3.14 Other Advanced Optimisation Methods

In addition to the main categories above, several advanced and niche methods are also used in particular contexts:

### Evolutionary Algorithms

Evolutionary algorithms, like Genetic Algorithms, are often used for optimization in highly complex, poorly understood problem spaces. These methods include:

- **Differential Evolution (DE),** an evolutionary algorithm that optimises a problem by iterating over generations and adjusting population members using random variations and recombination.
- **Evolution Strategies (ES),** similar to Genetic Algorithms but focuses more on the mutation process and adapts the variance of mutation distribution over time.

### Ant Colony Optimisation (ACO)

Inspired by the foraging behaviour of ants, ACO algorithms are particularly useful for solving discrete optimisation problems, such as routing and scheduling.

- **Ant Colony System (ACS),** an adaptation of the basic ACO algorithm, which uses pheromones deposited by ants to guide future search steps, typically used in combinatorial optimisation problems.

### Simultaneous Perturbation Stochastic Approximation (SPSA)

SPSA is a method used for optimisation in cases where the objective function is noisy or expensive to evaluate. It is particularly suitable for high-dimensional spaces where other gradient-based methods might struggle.

# 4 Applications in Key Industries

Optimisation algorithms drive efficiency and innovation across multiple fields. Below are some key industries where optimisation techniques have significant applications.

## 4.1 Healthcare

In healthcare, optimisation algorithms are used to enhance patient outcomes and streamline operations. Key applications include:

- **Radiation Therapy Planning,** which optimises dosage distribution to target tumours while minimising damage to healthy tissues.
- **Hospital Scheduling,** where linear programming is employed to allocate beds and schedule staff efficiently, ensuring optimal resource utilisation. The objective function in such problems can be formulated as:

$$\text{Minimise} \quad Z = \sum_{i=1}^{n} c_i x_i \tag{5}$$

  where $c_i$ represents the cost of assigning resource $i$ and $x_i$ is the decision variable indicating whether resource $i$ is allocated.

## 4.2 Marketing and Advertising

In marketing and advertising, optimisation techniques are critical for improving campaign performance and customer targeting. Examples include:

- **A/B Testing,** where Bayesian optimisation is used to determine the most effective marketing strategies based on data-driven insights. The Bayesian approach allows for continual learning from the data, where the posterior distribution is updated as new information becomes available:

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)} \tag{6}$$

  where $P(\theta|D)$ is the posterior probability of the parameter $\theta$ given data $D$, $P(D|\theta)$ is the likelihood, and $P(\theta)$ is the prior distribution.

- **Customer Segmentation,** in which clustering algorithms optimise personalised recommendations, enhancing customer engagement and conversion rates. K-means clustering is one such algorithm, which seeks to minimise the within-cluster sum of squares:

$$J = \sum_{i=1}^{k} \sum_{j=1}^{n_i} \|x_j - \mu_i\|^2 \tag{7}$$

  where $\mu_i$ is the centroid of cluster $i$, and $n_i$ is the number of points in cluster $i$.

- **Marketing Mix Modelling (MMM),** which uses regression-based methods to allocate budgets across different advertising channels, maximising return on investment. The objective is to maximise a return function $R$ that models the relationship between expenditure and returns:

$$R = \alpha + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n \tag{8}$$

where $X_i$ represents the advertising expenditure in channel $i$, and $\beta_i$ are the coefficients indicating the effectiveness of each channel.

## 4.3 Energy and Sustainability

Optimisation plays a pivotal role in managing energy resources and promoting sustainability. Examples include:

- **Smart Grid Optimisation,** where dynamic programming techniques are used to minimise energy wastage and balance supply with demand. The dynamic programming equation can be expressed as:

$$V_t(s_t) = \max_a \left( r_t(s_t, a) + \gamma \sum_{s'} P(s'|s_t, a) V_{t+1}(s') \right) \tag{9}$$

where $V_t(s_t)$ is the value of the state $s_t$ at time $t$, $r_t(s_t, a)$ is the reward, $P(s'|s_t, a)$ is the transition probability, and $\gamma$ is the discount factor.

- **Renewable Energy Distribution,** where metaheuristic algorithms help balance power supply and demand, optimising the distribution of renewable energy sources to minimise costs and environmental impact.

## 4.4 Transportation and Logistics

In transportation and logistics, optimisation algorithms are used to improve the efficiency of operations and reduce costs. Key applications include:

- **Route Optimisation,** which is used by companies like FedEx and UPS to design fuel-efficient delivery routes, reducing operational costs and environmental impact. The problem can be modelled as a Travelling Salesman Problem (TSP), aiming to minimise the total distance $D$ by finding the shortest path:

$$D = \sum_{i=1}^{n-1} \|x_i - x_{i+1}\| \tag{10}$$

where $x_i$ is the location of delivery point $i$, and $n$ is the total number of points.

- **Supply Chain Management,** where mixed-integer programming (MIP) is employed to ensure cost-efficient logistics, optimising inventory, distribution, and scheduling. The general form of an MIP problem can be written as:

$$\text{Minimise} \quad Z = \sum_{i=1}^{n} c_i x_i \quad \text{subject to} \quad Ax \leq b \tag{11}$$

where $c_i$ represents costs, $x_i$ are decision variables, $A$ is a matrix of constraints, and $b$ is the constraint vector.

## 4.5 Finance and Investment

Optimisation algorithms are widely used in finance and investment to maximise returns and manage risks. Notable applications include:

- **Portfolio Optimisation,** where Modern Portfolio Theory (MPT) is used to find the optimal risk-return trade-off by diversifying assets. The optimisation problem is typically formulated as:

$$\text{Minimise} \quad \sigma_p^2 = \sum_{i=1}^{n} w_i^2 \sigma_i^2 + 2 \sum_{i<j} w_i w_j \sigma_{ij} \tag{12}$$

  where $\sigma_p^2$ is the portfolio variance, $w_i$ is the weight of asset $i$, $\sigma_i^2$ is the variance of asset $i$, and $\sigma_{ij}$ is the covariance between assets $i$ and $j$.

- **High-Frequency Trading,** where reinforcement learning is applied to optimise algorithmic trading strategies, allowing for rapid decision-making and execution.

# 5 Challenges and Future of Optimisation

Despite the power of optimisation algorithms, several challenges remain in their application and development. These challenges include:

- **Local Minima,** where non-convex problems can trap solutions in suboptimal points, making it difficult to find the global optimum. One common approach to overcome this is using techniques like simulated annealing or genetic algorithms, which allow for exploration of the solution space. For example, simulated annealing is based on the concept of probabilistic transitions to escape local minima, where the temperature parameter controls the likelihood of accepting worse solutions. The temperature is gradually decreased according to the schedule:

  $$T(t) = T_0 \cdot \alpha^t$$

  where $T_0$ is the initial temperature, $\alpha$ is a constant factor (typically close to 1), and $t$ is the current iteration.

- **Computational Complexity,** particularly in high-dimensional problems, where the number of variables increases exponentially. To address this, advanced techniques such as parallelisation, decomposition methods, or approximation algorithms are often required to reduce computational demands. For instance, the curse of dimensionality can make exact solutions computationally expensive, motivating the use of heuristic methods or approximation algorithms such as *simulated annealing*, *genetic algorithms*, and *ant colony optimisation*.

- **Interpretability,** which is crucial in critical applications like healthcare and finance. In these fields, stakeholders need to understand how decisions are made to ensure trust in the optimisation model. For example, in medical diagnostics, the interpretability of machine learning models like decision trees or linear models can aid in understanding how conclusions were reached. This has led to the development of interpretable machine learning models and optimisation algorithms that prioritise transparency. The challenge is balancing model complexity and explainability, as more complex models may yield better results but at the cost of interpretability.

## 5.1 Future Directions

The future of optimisation is promising, with several emerging directions likely to shape its evolution:

- **Quantum Computing,** which holds the potential to revolutionise combinatorial optimisation problems, such as the Travelling Salesman Problem (TSP) and scheduling problems. Quantum algorithms like Grover's and Shor's could provide exponential speedups for certain classes of problems. Specifically, Grover's algorithm offers a quadratic speedup for unstructured search problems, and Shor's algorithm can factor large numbers exponentially faster than the best-known classical algorithms. These advances could significantly reduce the time required to solve problems such as the TSP, which is NP-hard.

- **AI-Driven Optimisation,** where machine learning techniques are combined with optimisation algorithms to adapt to dynamic environments. For example, reinforcement learning can be used to optimise decisions in environments that change over time, such as supply chain management or automated trading systems. Reinforcement learning models, such as Q-learning, can dynamically adjust policies based on evolving states, optimising long-term performance over time:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[ r + \gamma \max_{a} Q(s',a) - Q(s,a) \right]$$

  where $Q(s,a)$ is the action-value function, $\alpha$ is the learning rate, $r$ is the immediate reward, $\gamma$ is the discount factor, and $s'$ is the next state.

- **Hybrid Models,** which combine heuristics (e.g., genetic algorithms) with gradient-based methods (e.g., gradient descent) to achieve better performance. These models can efficiently navigate large, complex solution spaces while maintaining the accuracy of optimisation, particularly in non-convex problems. Hybrid approaches aim to combine the global search capabilities of heuristics with the local refinement abilities of gradient-based methods, making them well-suited for complex real-world optimisation problems.

# 6 Conclusion

Optimisation algorithms are essential for driving technological advancements and enhancing decision-making across various industries, including healthcare, marketing, energy, logistics, and finance. By enabling more efficient resource allocation, improved forecasting, and better strategic decisions, optimisation techniques have revolutionised problem-solving capabilities. As computing power continues to grow, emerging methods such as hybrid optimisation and AI-driven approaches will provide even greater efficiency and effectiveness. These advancements are expected to further accelerate innovation and productivity, leading to more intelligent, adaptive, and sustainable systems across diverse sectors.

For instance, in portfolio optimisation, the objective is to maximise return $R$ for a given level of risk $\sigma$. This can be represented by the following optimisation problem:

$$\text{Maximise} \quad R = \sum_{i=1}^{n} w_i r_i$$

subject to:

$$\text{Minimise} \quad \sigma = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j \text{Cov}(r_i, r_j)}$$

where $w_i$ are the weights of assets, $r_i$ are the returns, and $\text{Cov}(r_i, r_j)$ is the covariance between the returns of assets $i$ and $j$.

## 6.1 Further Questions to Explore

The field of optimisation continues to evolve, raising several intriguing questions that warrant further exploration:

- **How can optimisation algorithms be adapted for real-time decision-making?**
  In industries such as finance and logistics, real-time decision-making is critical. Optimisation algorithms can be adapted for real-time use by incorporating fast, approximate methods. For example, heuristics, metaheuristics, or approximation algorithms can provide quick solutions by sacrificing some accuracy for speed. Additionally, parallel computing techniques, where computations are divided across multiple processors, can speed up the process significantly. Real-time adaptation could also include techniques like reinforcement learning, where an agent learns optimal decisions through trial and error in dynamic environments, such as traffic flow optimisation or supply chain management.

- **What are the ethical considerations in applying optimisation to sensitive domains like hiring and healthcare?**
  Optimisation algorithms, if not designed carefully, can introduce biases in sensitive domains. In hiring, for instance, an algorithm trained on biased data might favour candidates from certain demographics over others, even if it doesn't intend to do so. In healthcare, algorithms could unintentionally lead to unequal access to treatment based on race or socio-economic status. To mitigate such risks, it is essential to

integrate fairness constraints into the optimisation process. One approach is to enforce demographic parity or equal treatment for different groups, ensuring that the algorithm's decisions do not discriminate. Moreover, transparency in the decision-making process, such as providing explanations for how conclusions are drawn, is crucial for building trust in sensitive applications.

- **How will quantum computing impact traditional optimisation techniques?** Quantum computing holds the potential to significantly impact optimisation, particularly in solving complex combinatorial problems. Traditional optimisation techniques often struggle with problems that involve a large number of variables, such as the Travelling Salesman Problem (TSP). Quantum computing can provide exponential speed-ups for certain types of optimisation problems. For instance, Grover's algorithm allows for faster searching of unsorted data, while Shor's algorithm provides efficient factorisation of large numbers. These advancements could make it possible to tackle previously intractable optimisation problems. However, quantum computing is still in the early stages, and understanding how to integrate quantum algorithms with existing optimisation methods will require further research. This will likely lead to hybrid models, where classical and quantum techniques are combined for better performance.