PAPER • **OPEN ACCESS**

Quantum assisted genetic algorithm for multi-objective optimization

To cite this article: Rushit Kansara *et al* 2026 *Phys. Scr.* **101** 225103

Physica Scripta

OPEN ACCESS

RECEIVED
12 February 2026

REVISED
19 April 2026

ACCEPTED FOR PUBLICATION
11 May 2026

PUBLISHED
1 June 2026

PAPER

Quantum assisted genetic algorithm for multi-objective optimization

Rushit Kansara*, María Isabel Roldán Serrano and Martin Bähr

German Aerospace Center, Institute of Low-Carbon Industrial Processes, Weinbergstrasse 10, 03050 Cottbus, Germany

* Author to whom any correspondence should be addressed.

E-mail: rushit.kansara@dlr.de

Keywords: quantum annealing, genetic algorithms, multi-objective optimization

Abstract

Multi-objective optimization (MOO) remains a significant computational challenge, particularly for high-dimensional, non-convex problems which demand substantial computational effort from traditional evolutionary algorithms to obtain a Pareto front that is both well converged and uniformly diverse. To address this challenge, this work presents the quantum-assisted non-dominated sorting genetic algorithm (QANSGA), a novel hybrid heuristic that leverages principles of quantum annealing to enhance the core mechanisms of classical MOO genetic algorithm. The quantum component is realized via a D-Wave quantum annealer, which is used to implement quantum mutation operators and facilitate the injection of highly promising quantum solutions into the classical population pool. This hybrid mechanism fundamentally improves the population's diversity maintenance, the global search capability and accelerates its convergence dynamics toward the true Pareto-optimal set, achieving the desired global optimum Pareto front in fewer generations. To rigorously evaluate its performance, we benchmark QANSGA against leading classical genetic algorithm (GA) for MOO problems, which is non-dominated sorting genetic algorithm (NSGA-II) across a widely used suite of complex multi-objective benchmark problems.

The results provide compelling evidence for the efficacy of quantum-inspired heuristics in overcoming the efficiency bottlenecks of conventional MOO methods. The study concluded that QANSGA outperforms classical NSGA-II in terms of convergence and diversity by 33% and 42% respectively on an average when applied to 6 benchmark multi-objective problems. This work positions QANSGA as a valuable new hybrid algorithm, paving the way for novel approaches to tackle multi-objective, complex optimization challenges in areas like energy system design, resource allocation, and logistics.

Nomenclature

Latin symbols

A, B	Time dependent coefficient
C	Constraint
D	Design samples
f, h, g	functions
H	Hamiltonian
J	Coupling strength
l	Local biases
M	Total number of objectives
m	Total number of variables

P	Population
Q	QUBO coefficients
s	Spin
v	Continuous variable
W	Weight matrix
x	Design variable
Greek symbols	
α, β	Complex probability amplitude
ϕ	Decoding function
ψ	Quantum state
Subscripts and superscripts	
max	Maximum
min	Minimum
gen	Generation
var	Variable
K	Number of quantum solutions
k	variable index
N	Total number
Abbreviations	
GA	Genetic algorithm
GD	Generational distance
HV	Hyper volume
IGD	Inverted generational distance
MOO	Multi-objective optimization
MOQSOA	Multi-objective quantum-inspired seagull optimization algorithm
NDS	Non-dominated sorting
NSGA	Non-dominated sorting genetic algorithm
OSY	Osyczka
PSD	Positive semi definite
QA	Quantum annealing
QAGA	Quantum assisted genetic algorithm
QANSGA	Quantum assisted non-dominated sorting genetic algorithm
QEEA	Quantum enhanced evolutionary algorithm
QUBO	Quadratic unconstrained binary optimization
TNK	Tanaka
VQA	Variational quantum algorithm
ZDT	Zitzler Deb Thiele

1. Introduction

Conventional parameter optimization techniques aim to identify a single optimal solution by consolidating multiple objectives into a single scalar value, frequently through the use of weighted sums. While effective, when objectives are correlated, this approach fails in complex engineering and design problems where objectives are inherently conflicting; for example, maximizing performance while simultaneously minimizing cost or weight [1]. In these cases, no single design can be considered globally superior.

Multi-objective optimization (MOO), also referred to as Pareto optimization, tackles this issue by determining a set of optimal trade-off solutions, which together constitute the Pareto set. When these solutions are projected onto the objective space, they form the Pareto front. Providing this front facilitates designers and engineers to select a design that best fits their needs, with a clear understanding of the compromises involved.

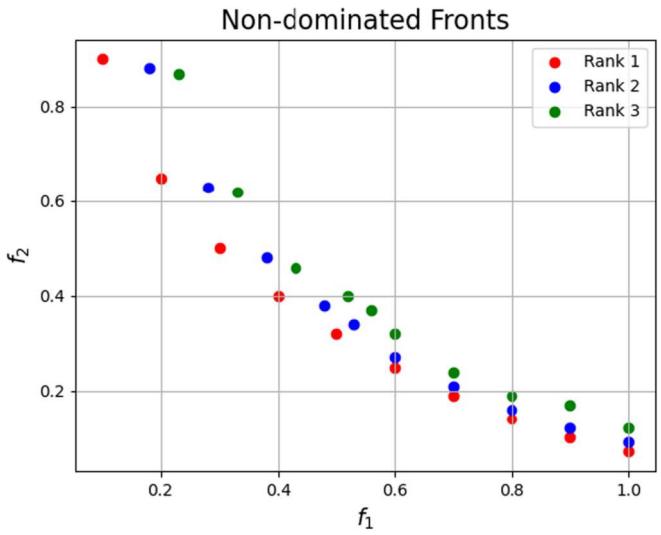

Figure 1. Non-dominated Pareto fronts by Ranks of dominance in minimizing two-objectives f_1 and f_2. The illustrative example is recreated from [1].

The optimization problems discussed here are typically defined by the goal of minimizing (or maximizing) p objective functions $f_i(\mathbf{x})$, while satisfying q inequality constraints $C_j(\mathbf{x})\leqslant 0$, where $\mathbf{x}$ represents the n design variables. The generic multi-objective problem is shown as:

$$\min_{\mathbf{x}} \quad (f_1(\mathbf{x}), f_2(\mathbf{x}),\ldots,f_m(\mathbf{x})) \tag{1}$$

$$\text{subject to} \quad C_j(\mathbf{x}) \leqslant 0, \quad j = 1, 2,\ldots,q \tag{2}$$

$$\mathbf{x} \in \mathbb{R}^n \tag{3}$$

1.1. Non-dominated sorting

A common and highly effective strategy for ranking candidate designs in MOO is non-dominated sorting (NDS) [2, 3]. The designs are referred to as solutions. The core principle of NDS is Pareto dominance: Design A is said to dominate Design B if A is better than B in at least one objective function and is not worse than B in all other objective functions.

The NDS process iteratively ranks the designs within a population. The first step searches and recognizes all feasible solutions that are not dominated by any other solution in the current set; these solutions are assigned Rank 1 and constitute the current non-dominated set. Conceptually, these designs are removed, and the NDS process is repeated on the remaining designs to determine Rank 2, and so forth as shown in figure 1. The set of Rank 1 designs is often referred to as a local Pareto front, which, as the optimization process progresses, must converge toward the true Pareto front. true Pareto front consists of the set of solutions which are non-dominated across the entire search space.

1.2. Performance and efficiency of the multi-objective optimization algorithm

The performance of a MOO algorithm can be assessed with different metrics. It requires a comprehensive evaluation of the quality of the resulting solution set. Unlike single-objective optimization, which mainly relies on a single fitness value, MOO algorithm is compared using criteria that quantify the convergence to true Pareto front and how effectively it covers and represent objective space.

Some of the criteria to assess the performance of a MOO algorithm are: execution time of the algorithms [4]; convergence to the bench-marking optimal Pareto front, indicated by the normalized generational distance (GD) [5]; normalized inverted generational distance (IGD) to quantify both the convergence and the diversity [6]; spread indicator introduced by [7]; spacing, which shows the uniformity of the solution set by measuring the standard deviation of the distances between consecutive solutions [8]; hypervolume (HV) [9]; total number of generations require to reach true Pareto front [10].

For the purposes of this discussion, we consider two criteria: the evaluation of the quality and convergence of the results. For each chosen MOO the GD, IGD and HV are calculated and compared for different number of generations. The comparison shows the performance of each multi-objective algorithm in terms of accuracy, diversity and computational efforts. The efficiency of a Pareto optimization algorithm is assessed based on the

computational effort required, particularly the minimum number of evaluations needed to achieve a high-quality approximation of the true Pareto front [1].

1.3. Research gap in multi-objective quantum algorithms

Quantum computing is rapidly gaining prominence across disciplines due to its capability for significant computational acceleration [11, 12], with development focusing primarily on two architectures: universal quantum computers and specialized quantum annealers. While universal systems offer high flexibility for diverse problems [13], their current hardware is limited to a few hundred qubits, restricting the size of optimization problems [14]. Conversely, quantum annealers currently boast over 5,000 qubits [15] and are specifically designed for combinatorial optimization, excelling at quadratic unconstrained binary optimization (QUBO) problems [16]. For annealers to tackle optimization problems with non-binary variables (integer or continuous), these must be binary encoded [17], creating a critical trade-off where solution accuracy is balanced against the required qubit count. This discretization strategy has been successfully applied to complex energy system optimization, demonstrated in applications like heat exchanger network synthesis, reformulation of unit commitment problems [18], and the tactical capacity problem for distributed electricity generation [19].

The application of quantum computation to MOO, which seeks the optimal set of non-dominated solutions known as the Pareto front, is explored across multiple quantum architectures due to the computational complexity of the task.Recent research on variational quantum algorithms (VQAs) has introduced techniques such as the variational quantum multi-objective optimization algorithm, which integrates all objective function Hamiltonians directly into the quantum circuit. This enables the generation of a quantum state that encodes Pareto-optimal solutions in superposition, subsequently using the hypervolume indicator to measure the quality of the resulting solution set [20]. Low-depth quantum approximate optimization algorithm (QAOA) has also been used to approximate optimal Pareto front in certain multi-objective weighted maximum cut problem [21]. In contrast, specialized quantum annealers have been used to formulate MOO problems, with comparative studies showing that quantum annealing (QA) can outperform classical and VQA methods on certain problems, often leading to an improved Pareto front [22]. Furthermore, quantum-enhanced evolutionary algorithms represent a critical hybrid approach, leveraging quantum principles to improve classical heuristics: this includes foundational work on the quantum-assisted genetic algorithm, which uses reverse QA as a novel source of mutation to find global optima where forward QA struggles [23], and advanced methods that frame selection as a binary quadratic minimization problem solved by a quantum annealer, demonstrating performance gains over classical selection operators [24]. Complementary Multi-Objective Quantum-inspired Seagull Optimization algorithm (MOQSOA), which integrate quantum particle behaviors like superposition and entanglement into classical swarm techniques, have also shown promising results in accelerating convergence and enhancing solution diversity in particular multi-objective environments, as seen in the MOQSOA [25]. While current quantum-assisted methods show promise through enhanced operators or specialized quantum-inspired mechanisms, a dedicated and comprehensive quantum-annealing solutions' integration into classical GA structure to get better performance remains unexplored. The aim of this paper is to bridge this gap by introducing the quantum-assisted non-dominated sorting genetic algorithm (QANSGA) and providing a performance comparison against existing GAs for MOO problems.

The QANSGA is a sophisticated hybrid metaheuristic algorithm designed to overcome the challenges of classical GAs for MOO problems. It seamlessly integrates the high-performance NDS and crowding mechanisms of non-dominated sorting genetic algorithm (NSGA-II) with a tailored, periodic quantum mutation executed on commercial quantum annealer. QANSGA uses the classical NSGA-II with QA. To understand the mechanism of QANSGA, some fundamental understanding of QA is necessary, which will be discussed in next subsections.

1.4. Research objectives

This paper introduces QANSGA, a cutting-edge novel hybrid quantum–classical optimization algorithm designed to address complex multi-objective problems more efficiently. The core mechanism of QANSGA is its strategic integration of the advantage of QA, specifically its ability to quickly explore vast energy landscapes and find low-energy minima to boost the performance and exploration capabilities of the widely adopted classical method, the NSGA-II. This hybridization aims to accelerate the discovery and refinement of the Pareto front approximation. To rigorously assess its efficacy, the paper contributes to the existing pool of knowledge by performing a comprehensive performance evaluation- directly comparing QANSGA against its classical counterpart: NSGA-II. This comparative study is conducted across a suite of six diverse and challenging benchmark MOO problems, ensuring a robust validation of QANSGA's superiority in key metrics such as convergence and the diversity of the solutions found. The structure of this paper systematically presents the research, beginning with the definition of the benchmark multi-objective problems and followed by a detailed

explanation of the QANSGA's architecture and operation. Subsequent sections include the empirical performance comparison of QANSGA and classical GAs, a critical discussion of the results, and finally, a conclusion and future outlook on the role of QA in advancing MOO.

2. Methodology

As discussed in section 1, the performance of the MOO problems must be evaluated for two key criteria: convergence to the optimal Pareto front and the diversity within the population. Another important factor for evaluating MOO methods is the computational effort, but this has less priority in this study. Zitzler *et al* [26] have shown different features that can cause difficulties for evolutionary algorithms in solving multi-objective problems. For example, the convergence of multi-objective evolutionary optimization algorithms can be hindered by factors such as multimodality, deception, and isolated optima. A lack of diversity can further limit the ability to achieve a well-distributed non-dominated Pareto front. Some of the causes can be: convexity or non-convexity, discontinuity and non-uniformity [27]. To address these difficulties we have chosen six different multi-objective problems for benchmarking the optimization results. Four Zitzler Deb Thiele (ZDT) problems, OSY, and TNK functions are utilized as standard benchmark problems in this paper because they represent a diverse taxonomy of mathematical challenges that stress-test an optimization algorithm's ability to navigate complex Pareto landscapes. The ZDT functions focuses on structural difficulties such as non-convexity, high-dimensionality, and discontinuity, which specifically test an algorithm's global search efficiency and its ability to avoid being trapped in local optima [27]. In contrast, Osyczka (OSY) and Tanaka (TNK) problems introduce rigorous constraint-handling requirements, featuring oddly shaped feasible regions and disconnected optimal fronts that lie along constraint boundaries [28, 29]. Together, these problems provide a comprehensive framework to evaluate the two primary goals of multi-objective optimization: convergence and diversity, ensuring that a novel algorithm like QANSGA is robust enough for real-world engineering applications.

For reasons of better visualization and interpretation, we have restrict ourselves to only two objectives in the investigation of all the problems.

2.1. Benchmark multi-objective problems

Below, we briefly introduce the six widely used benchmark problems considered in this work; for more details we refer the reader to the original papers. All problems have two objective functions, but differ in features such as the number of design variables and constraints, the shape (convex, concave, mixed, discontinuous) or the multimodality (local, global) of the Pareto front, which are generally difficult for the algorithms to handle.

2.1.1. ZDT problems

The first four benchmark problems are based on Zitzler Deb Thiele's (ZDT) test suite introduced in [27]. The four unconstrained ZDT problems are constructed in a similar manner in which the second objective function consists itself of the first objective function f_1 and two subfunctions g, h, given in the following formulation:

$$\min_{\mathbf{x}} \quad (f_1(\mathbf{x}), f_2(\mathbf{x})) \tag{4}$$

with

$$f_2(\mathbf{x}) = g(\mathbf{x})h(f_1(\mathbf{x}), g(\mathbf{x})) \tag{5}$$

where $\mathbf{x} \in \mathbb{R}^m$ and m is the number of decision variables. The function f_1 only consists of the first decision variable x_1, the function g depends on the $m-1$ remaining variables and the function h is a composition of f_1 and g. Here, h defines the Pareto front shape, while g defines the distance (or convergence). As a result, the function f_2 is a product of g and h. Basically, the ZDT problems differ in these three functions g, h, f_2, in the function domain and in the number of variables m. The latter can be used to increase the problem complexity by varying the dimensionality of the design space, which is also shown later in results section to compare performance for higher dimension problems.

The first ZTD problem, called ZDT-1, is the most simple one (of the ZDT problem suite) and has a convex Pareto front. The ZDT-1 problem is given by

$$f_1(\mathbf{x}) = x_1, \quad g(\mathbf{x}) = 1 + \frac{9\sum_{i=2}^{m} x_i}{m-1}, \quad h(f_1(\mathbf{x}), g(\mathbf{x})) = 1 - \sqrt{\frac{f_1(\mathbf{x})}{g(\mathbf{x})}} \tag{6}$$

with $m = 30$ and $x_i \in [0, 1]$. The true Pareto front is formed for $x_1^* \in [0,\ 1]$ and $x_i^* = 0,\ i = 2,\ 3,\ldots,m$ which corresponds to $g(\mathbf{x}) = 1$.

The second problem ZDT-2 has a concave Pareto front, which can be seen as a counterpart to ZDT-1. The non-convex Pareto front shape can cause evolutionary algorithms to converge prematurely or have difficulty maintaining a diverse solution set across the entire front. The ZDT-2 problem is formulated as

$$f_1(\mathbf{x}) = x_1, \quad g(\mathbf{x}) = 1 + \frac{9\sum_{i=2}^{m} x_i}{m-1}, \quad h(f_1(\mathbf{x}), g(\mathbf{x})) = 1 - \left(\frac{f_1(\mathbf{x})}{g(\mathbf{x})}\right)^2 \tag{7}$$

with $m = 30$ and $x_i \in [0, 1]$. The true Pareto front is formed again $x_1^* \in [0,\ 1]$ and $x_i^* = 0,\ i = 2,\ 3,\ldots,m$, i.e. $g(\mathbf{x}) = 1$.

The problem ZDT-3 is characterized by a discontinuous convex Pareto front, meaning that the true Pareto front consists of many disconnected convex parts. In general, algorithms often have difficulty to move between these disconnected regions, leading to incomplete or biased results. The ZDT-3 problem reads

$$f_1(\mathbf{x}) = x_1, \quad g(\mathbf{x}) = 1 + \frac{9\sum_{i=2}^{m} x_i}{m-1} \quad h(f_1(\mathbf{x}), g(\mathbf{x})) = 1 - \sqrt{\frac{f_1(\mathbf{x})}{g(\mathbf{x})}} - \left(\frac{f_1(\mathbf{x})}{g(\mathbf{x})}\right)\sin(10\pi f_1(\mathbf{x})) \tag{8}$$

with $m = 30$ and $x_i \in [0, 1]$. The true Pareto front is formed with $g(\mathbf{x}) = 1$. The discontinuity in the true Pareto front is introduced by the periodic behavior of the sine function in h.

The fourth and last ZDT problem considered in this paper is ZDT-4. This problem is highly multimodal, including 21^{m-1} local Pareto fronts in addition to the global Pareto-optimal front. The multimodality makes it difficult for the algorithms to converge to the true convex Pareto front, so they can easily get trapped in a local Pareto front. The ZDT-4 problem is shown as:

$$f_1(\mathbf{x}) = x_1, \quad g(\mathbf{x}) = 10m - 9 + \sum_{i=2}^{m}(x_i^2 - 10\cos(4\pi x_i)), \quad h(f_1(\mathbf{x}), g(\mathbf{x})) = 1 - \sqrt{\frac{f_1(\mathbf{x})}{g(\mathbf{x})}} \tag{9}$$

with $m = 10, x_1 \in [0, 1]$ and $x_i \in [-5, 5], i = 2, \ldots, m$. The true Pareto front is formed again with $g(\mathbf{x}) = 1$.

2.1.2. OSY problem

The fifth problem considered for the benchmark is the Osyczka (OSY) problem according to [28], in which the MOO problem is characterized by two objectives, six design variables and six constraints. Basically, the difficulty of the problem is to find the entire spread of the true Pareto front along the varying active constraints. The OSY problem is presented as:

$$\min_{\mathbf{x}} \quad (f_1(\mathbf{x}), f_2(\mathbf{x})) \tag{10}$$

$$\text{subject to} \quad C_1(\mathbf{x}) \coloneqq x_1 + x_2 - 2 \geqslant 0, \tag{11}$$

$$C_2(\mathbf{x}) \coloneqq 6 - x_1 - x_2 \geqslant 0, \tag{12}$$

$$C_3(\mathbf{x}) \coloneqq 2 - x_2 + x_1 \geqslant 0, \tag{13}$$

$$C_4(\mathbf{x}) \coloneqq 2 - x_1 + 3x_2 \geqslant 0, \tag{14}$$

$$C_5(\mathbf{x}) \coloneqq 4 - (x_3 - 3)^2 - x_4 \geqslant 0, \tag{15}$$

$$C_6(\mathbf{x}) \coloneqq (x_5 - 3)^2 + x_6 - 4 \geqslant 0 \tag{16}$$

with

$$f_1(\mathbf{x}) = -[25(x_1 - 2)^2 + (x_2 - 2)^2 + (x_3 - 1)^2 + (x_4 - 4)^2 + (x_5 - 1)^2] \tag{17}$$

$$f_2(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2 \tag{18}$$

where, $x_1, x_2, x_6 \in [0, 10]$, $x_3, x_5 \in [1, 5]$, and $x_4 \in [0, 6]$. The true Pareto front are five connected sections in which several constraints are active. The entire optimal Pareto-region holds for $x_4^* = x_6^* = 0$. Table 1 shows the values of other variables in optimal Pareto-regions.

2.1.3. TNK problem

The sixth and last benchmark problem is based on Tanaka (TNK) proposed in [29]. The TNK problem has two objectives, two design variables and two constraints and results in a nonconvex, discontinuous Pareto front, with the difficulty of finding the diverse solutions on the Pareto front coincide on nonlinear constraints. The TNK problem is defined as:

$$\min_{\mathbf{x}} \quad (f_1(\mathbf{x}), f_2(\mathbf{x})) \tag{19}$$

$$\text{subject to} \quad C_1(\mathbf{x}) \coloneqq x_1^2 + x_2^2 - 1 - 0.1\cos\left(16\arctan\left(\frac{x_1}{x_2}\right)\right) \geqslant 0, \tag{20}$$

Table 1. Optimal values of the variables x_1, x_2, x_3 and x_5 in the optimal Pareto region of the OSY problem [28].

Optimal Values				
x_1^*	x_2^*	x_3^*	x_5^*	Active constraints
5	1	(1,…,5)	5	2,4,6
5	1	(1,…,5)	1	2,4,6
(4.056, …,5)	$(x_1^* - 2)/3$	1	1	2,4,6
0	2	(1,…,3.732)	1	2,4,6
(0,…,1)	$2 - x_1^*$	1	1	2,4,6

$$C_2(\mathbf{x}) \coloneqq (x_1 - 0.5)^2 + (x_2 - 0.5)^2 \leqslant 0.5 \tag{21}$$

with

$$f_1(\mathbf{x}) = x_1, \quad f_2(\mathbf{x}) = x_2 \tag{22}$$

where, $x_1, x_2 \in [0, \pi]$. Naturally, C_1 is periodic and C_2 must also be fulfilled, not all solutions on the boundary of the first constraint are Pareto-optimal, so that this interaction results in a disconnected Pareto front. As a result, a significant challenge for any optimization algorithm is maintaining adequate diversity, that is, achieving a uniform and comprehensive spread of solutions across all segments of the non-dominated set.

2.2. Benchmark classical genetic algorithms

In this subsection, we briefly introduce the popular NSGA-II, which is used as a comparative method in this study. NSGA-II is a well-known and widely used evolutionary multi-objective optimization (EMOO) algorithm, recognized for its computational efficiency and elitist strategy [2]. It improves the original NSGA by incorporating three key features designed to enhance both convergence and diversity:

- Fast NDS: This process efficiently categorizes the population into different Pareto fronts (ranks) by assessing the dominance relationship between solutions in $O(MN^2)$ time, where M represents the number of objectives and N denotes the population size. This mechanism ensures that superior solutions are preferentially retained.
- Elitism: The algorithm merges the current population with the offspring population to form a combined pool, ensuring that all previously identified non-dominated solutions are retained and given the opportunity to compete in subsequent generations.
- Crowding distance metric: To preserve solution diversity and avoid premature convergence, NSGA-II employs the crowding distance operator. The crowding distance of a solution is defined as the perimeter of the cuboid formed by its nearest neighbors on the same Pareto front. Solutions with larger crowding distances are located in less crowded areas and are therefore prioritized during selection to ensure maximization of diversity and spread within the non-dominated set. This operator acts as a density estimator, enabling the algorithm to maintain a diverse set of trade-off solutions without the need for external parameters.

Although the NSGA-II is considered a highly efficient EMOO algorithm, it exhibits significant limitations of classical evolutionary algorithms, especially when scaling to large, complex optimization landscapes. Its main drawbacks include:

- Computational bottleneck for a large number of function evaluations (NFE): Although sorting is computationally efficient, the total NFE required to achieve satisfactory convergence can be prohibitively large, especially if the objective functions involve computationally expensive simulations.
- Difficulty with many-objective problems: The effectiveness of the crowding distance metric degrades significantly as the number of objectives M increases, leading to ineffective approximations of the Pareto front and a loss of diversity in higher dimensions (the curse of dimensionality) and therefore requiring high computational costs for accurate solutions.

- Lack of local search ability: NSGA-II primarily relies on genetic operators (crossover and mutation) which are effective for global exploration, i.e. moving towards promising regions within the entire search space and thus maintaining diversity, but struggle with local exploiting in finding precise optimal solutions, often leading to slow fine-tuning near the true Pareto front boundary.

The inherent limitations of the classical algorithms, in particular the high computational cost arising from larger number of function evaluations requiring large number of generations and the difficulties in achieving strong local search performance across complex, multimodal landscapes, form the basis for introducing the novel QANSGA. This hybrid algorithm is designed to utilize QAs rapid exploration capabilities to accelerate key components of the evolutionary process, thereby mitigating these classical deficiencies.

2.3. Implementation of QANSGA considering the fundamentals of quantum computing

Quantum computing leverages the fundamental, non-classical principles of quantum mechanics, particularly superposition and entanglement, to perform computations on an exponentially greater scale than classical models [30]. The fundamental unit of information is the qubit. Unlike a classical bit, a qubit can exist in a linear combination of its basis states ($|0\rangle$ and $|1\rangle$), known as superposition. This state is represented using Dirac notation as follows:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \quad \text{with} \quad |\alpha|^2 + |\beta|^2 = 1 \tag{23}$$

where, α and β are complex probability amplitudes. The normalization condition ensures that the total probability of all possible measurements sum to unity. This ability to explore multiple states simultaneously allows a system of N qubits to encode and explore 2^N states concurrently, forming the foundation for quantum speedup in search algorithms.

2.3.1. Quantum annealing

QA is an quantum computing optimization method designed to find the global minimum of a target objective function [31], primarily for discrete optimization problems. All problems submitted to a QA device must be formulated as a minimization problem on the Ising model Hamiltonian, which describes the energy of a system of spins $s_i \in \{-1, +1\}$, where $i = 1, \ldots, n$, which is shown as

$$H_{\text{Ising}}(\mathbf{s}) = \sum_i l_i s_i + \sum_{i<j} J_{ij} s_i s_j \tag{24}$$

where the coefficients l_i (local biases) and J_{ij} (coupling strengths) are the programmable parameters that encode the optimization problem. The forward QA process which is used in this paper is implemented by a time-dependent Hamiltonian

$$H(t) = A(t)H_{\text{initial}} + B(t)H_{\text{problem}} \tag{25}$$

The initial Hamiltonian H_{initial} serves as a starting point and drives the system from an initial, easily prepared ground state, uniform superposition, to the problem Hamiltonian H_{problem}, which encodes the target Ising function. The schedules $A(t)$ and $B(t)$ are designed so that the system starts at H_{initial} and follows H_{problem}, therefore $A(t)$ decreases as the annealing schedule progresses while $B(t)$ increases. The adiabatic theorem ensures that if this transformation occurs slowly enough, the system remains in the instantaneous ground state of $H(t)$, eventually reaching the minimum energy state of H_{problem}, which represents the optimal solution [31].

2.3.2. QUBO formulation, binary encoding and quadratic surrogate

To close the gap between continuous MOO problems and the QA hardware, the conversion of original continuous problem into unconstrained binary problem is necessary. The Ising model is equivalent to the QUBO problem, the native input format for D-Wave solvers. The QUBO formulation [32] minimizes a quadratic function of binary variables $x_i \in \{0, 1\}$ with $i = 1, \ldots, n$

$$\min S_{\text{QUBO}}(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} = \sum_i Q_{ii} x_i + \sum_{i<j} Q_{ij} x_i x_j \tag{26}$$

where the QUBO matrix $\mathbf{Q}$ is derived directly from the linear (diagonal entries) and quadratic (off-diagonal entries) coefficients, representing the relationships between these variables, of the scalarized objective function.

Each continuous variable v_k for $k = 1, \ldots, N$ in original problem is represented by a sequence of B binary variables, $\{x_{k,0}, x_{k,1}, \ldots, x_{k,B-1}\}$. N represents total number of continous variables. The index j represents the bit position, where $j = 0$ is the Least Significant Bit (LSB) and $j = B - 1$ is the Most Significant Bit (MSB). The transformation from the resulting binary solution to the real-valued design variable v_k is performed via the weights matrix $\mathbf{W} = \left(\frac{v_{k,\max} - v_{k,\min}}{2^B - 1}\right)$ and the lower-upper bounds $v_{k,\min}$ and $v_{k,\max}$:

$$v_k = v_{\min} + \left(\frac{v_{k,\max} - v_{k,\min}}{2^B - 1}\right)\sum_{j=0}^{B-1} x_{k,j}2^j \tag{27}$$

This discretization allows the continuous search space to be accurately represented on the digital quantum architecture.

In the QANSGA approach for the benchmark problems, the computationally expensive non-linear objective functions may be approximated by a quadratic surrogate model $\hat{f}(\mathbf{v})$ fitted to a preliminary set of sampled data points. The form of this surrogate model [33] reads as

$$\hat{f}(\mathbf{v}) \approx c + \mathbf{b}^{\mathrm{T}}\mathbf{v} + \frac{1}{2}\mathbf{v}^{\mathrm{T}}\mathbf{H}\mathbf{v} \tag{28}$$

with the scalar constant c, the vector $\mathbf{b}$, and the Hessian matrix $\mathbf{H}$, which for reasons of robustness is ensured to be positive semi-definite, is mathematically tractable for conversion into the QUBO matrix.

2.3.3. Embedding on quantum hardware

In QA, the optimization problem is encoded as a logical graph structurally defined by the QUBO matrix $\mathbf{Q}$, where the nodes are logical qubits (binary variables) and the edges are couplings (interaction between variables). In contrast, however, the physical D-Wave QA system has a fixed, sparse topology, e.g., the Chimera or Pegasus graph [15]. Embedding is the process of mapping the dense, problem-specific logical graph onto the sparse, fixed physical graph.

- The core embedding process (minor embedding): A logical qubit x_i is represented by one or more physical qubits q_l on the chip.

1. Physical topology constraint: The logical graph often requires connections (couplings Q_{ij}) that are not available between any two given physical qubits on the chip [15]. For example, the D-Wave Pegasus graph has a fixed structure, and if the QUBO requires a coupling between logical qubit 1 and logical qubit 10, but the corresponding physical qubits are far apart, they cannot be directly connected.
2. The chain mechanism: To overcome the sparsity constraint, a single logical qubit x_i is represented by a set of physically connected qubits (a chain or clique). These physical qubits are strongly coupled together ferromagnetically, meaning that they preferably have the same state, either all 0 or all 1 [15].

– Logical qubit: Represents a single variable x_i in the QUBO.

– Physical chain: A set of physical qubits $q_1, q_2, \ldots, q_l$ that represent x_i.

– Logical coupling: If logical qubit x_i needs to be coupled to x_j, the physical couplers between any two qubits in the x_i chain and the x_j chain are applied, provided that these physical connections exist.

3. Chain strength: The integrity of the chain is maintained by setting a very high chain strength (a negative ferromagnetic coupling J) between all adjacent qubits in the chain. This high bias forces the physical qubits in the chain to minimize their local energy by remaining in the same state throughout the annealing process. If the chain breaks (i.e., the qubits in the chain report different values, e.g., {0, 1, 0}), the resulting solution is invalid and the system is penalized by the high energy cost associated with the break [15].

- Fixed embedding: A fixed embedding approach pre-calculates the minor embedding once and reuses this specific mapping for all subsequent problem submissions, regardless of minor changes to the QUBO coefficients. This eliminates the computational overhead of running the embedding algorithm in each iteration of quantum calls. However, the embedding is optimized for a general QUBO structure, not for the specific coefficients of the current iteration [15].
- Dynamic embedding: A dynamic embedding approach calculates a new minor embedding each time a problem is submitted to the quantum machine. More precisely, the embedding algorithm is executed for each new QUBO submission to find the optimal mapping to the current physical topology. If faulty qubits are detected, the new embedding can bypass them, maximizing the use of the available processor. However, this significantly increases the overall solution time, which is impractical for iterative optimization in real-time or under strict time constraints [15].

2.3.4. QANSGA - quantum assisted non-dominated sorting genetic algorithm

The proposed novel QANSGA is described in algorithm 1. The algorithm begins with an offline surrogate modeling phase (lines 2–11). In this phase, a set of design samples is generated uniformly from the search space. In the case of objectives that are expensive to evaluate or highly nonlinear, a quadratic surrogate model is fitted based on the sampled data points. This surrogate model then provides a fast approximation of the objective function and reduces the need for repeated evaluations during the evolutionary process. Objectives that are nearly linear or inexpensive are evaluated directly without constructing a surrogate model.

Once the surrogate models are available, QANSGA proceeds with population initialization and generates an initial population of design vectors distributed across the feasible domain. The true objectives are evaluated for all individuals to obtain an initial approximation of the Pareto front (lines 13–14).

The main evolutionary loop (lines 15–33) iterates for a predefined number of generations, combining standard NSGA-II operations with quantum-assisted injection. Classical evolutionary operators, including crossover and polynomial mutation, generate offspring that are evaluated using the true objective functions. The parent and offspring populations are combined, and NDS is applied to identify Pareto fronts. To maintain diversity within each front, a crowding-distance metric is used, and the best individuals are selected to form the next generation.

At predefined intervals (lines 22–32), the algorithm performs quantum-assisted injection, which is referred to as quantum mutation (QM). It depends on the injection rate I_R, which determines after how many generations the QM has to be performed. In this phase, the multiple objectives are combined into a single scalar surrogate using a weighting scheme. The weighting scheme depends on the current generation n. Continuous design variables are encoded as binary variables to match the input requirements of QA hardware. The scalar surrogate is then converted into a QUBO problem, which is submitted to the quantum annealer. The annealer returns a set of low-energy binary solutions, for example 100 in the case of ZDT-1 problem, which are decoded back into continuous design vectors and evaluated using the true objectives. The best quantum-generated K solutions are then used to replace the worst-performing K individuals in the population, enabling the algorithm to explore promising regions of the search space that may be difficult to reach with classical operators alone.

Algorithm 1. QANSGA: quantum-assisted non-dominated sorting genetic algorithm

1: **Offline Surrogate Construction:**
2: Generate design samples $\mathcal{D} = \{\mathbf{v}^{(h)}\}_{h=1}^{N_{\text{train}}}$.
3: **for** each objective $f_i, i = 1, \ldots, M$ **do**
4: **if** f_i computationally expensive **then**
5: Fit quadratic surrogate

$$\hat{f}_i(\mathbf{v}) = c_i + \mathbf{b}_i^\top \mathbf{v} + \frac{1}{2}\mathbf{v}^\top \mathbf{H}_i \mathbf{v}.$$

6: Project Hessian: $\mathbf{H}_i \leftarrow \text{proj}_{\mathbb{S}_+}(\mathbf{H}_i)$.
7: **else**
8: $\hat{f}_i \leftarrow f_i$
9: **end if**
10 **end for**
11: **Initialization:**
12: Initialize population $P_0 = \{\mathbf{v}\}$.
13: Evaluate objective vectors $F(\mathbf{v}) = (f_1(\mathbf{v}), \ldots, f_M(\mathbf{v}))$.
14: **Main Evolution Loop:** 15: **for** $n = 1$ to N_{gen} **do**
16: Generate offspring Q_n via crossover + polynomial mutation.
17: Evaluate $F(\mathbf{v})$ for all $\mathbf{v} \in Q_n$.
18: $R_n = P_n \cup Q_n$ from parents and offspring solutions.
19: Extract Pareto fronts by NDS.
20: Form P_{n+1} using crowding-distance truncation.
21: **if** $n \bmod I_R = 0$ **then** $\rightarrow$*Quantum − Mutation*:
22: Sample scalarization weights $\boldsymbol{\alpha}_n \in \Delta^{M-1}$
23: Build scalar surrogate

$$\hat{g}(\mathbf{v}) = \sum_{i=1}^{M} \alpha_{n,i} \hat{f}_i(\mathbf{v}).$$

24: Encode each design variable by

$$v_k = v_{k,\min} + \left(\frac{v_{k,\max} - v_{k,\min}}{2^B - 1}\right) \sum_{j=0}^{B-1} x_{k,j} 2^j, \quad x_{k,j} \in \{0, 1\}.$$

25: Substitute encoding into $\hat{g}(\mathbf{v})$ to obtain QUBO: $S_{\text{QUBO}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{Q}\mathbf{x}$.

(Continued.)

Submit $\mathbf{Q}$ to annealer $\mathcal{A}$:

$$\{\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(K)}\} = \mathcal{A}(\mathbf{Q}).$$

Decode solutions $\mathbf{v}^{(k)} = \Phi(\mathbf{x}^{(k)})$.

Evaluate $F(\mathbf{v}^{(k)})$.

Compute scalar scores to rank candidates:

$$g(\mathbf{v}) = \boldsymbol{\alpha}_n^{\top} F(\mathbf{v})$$

Replace the K worst solutions in P_{n+1} with the K best quantum solutions.

end if

end for

return Non-dominated set P_N.

Specifically for constrained multi-objective problems such as OSY and TNK, feasibility is carefully maintained during the quantum-assisted mutation phase. Since the QUBO formulation is inherently unconstrained, constraint handling is incorporated through a penalty-based approach, where violations are added to the scalarized objective with appropriate weights to discourage infeasible solutions. However, due to the stochastic nature of quantum annealing, infeasible solutions may still be generated. Therefore, all decoded candidates are explicitly checked for feasibility after sampling. Feasible solutions are prioritized and ranked based on the scalarized objective, while infeasible ones are either penalized according to their violation magnitude or discarded if sufficient feasible solutions exist. During population update, only the best feasible quantum solutions are injected, replacing the worst-performing individuals, thereby ensuring that feasibility is preserved while still benefiting from quantum-assisted exploration.

3. Results and discussion

3.1. Pareto front and metrics comparison for different number of generations

Figure 2 shows the visual comparison of classical NSGA-II and QANSGA for ZDT-1 problem after different number of generations N_{gen}. Some of the algorithms parameters are kept constant here. The number of variables N_{var} is 10. Population size for both the algorithms is kept constant at 48. The quantum-injection rate I_R is 5, which means quantum-mutation (refer to 1) is performed every 5*th* generation. The number of classical solutions that are replaced with quantum solutions in quantum-mutation step, is kept constant to 5. The embedding on D-Wave Pegasus Graph is kept fixed in the beginning of the optimization.

Visual evolution of the solutions for both the algorithms is shown compared to the true Pareto front. As seen in figure 2, after 20 generations itself QANSGA starts to dominate NSGA-II. The QANSGA solutions progress faster towards true Pareto front compared to NSGA-II. After 20 generations, we can see the lack of spread and diversity in both set of solutions. The diversity and convergence of both set of solutions get better with increase in the number of generations. After 50 generations, QANSGA almost reaches true Pareto front, whereas NSGA-II still remains a little far from the true Pareto fronts. After 100 generations both set of solutions almost align and overlaps with true Pareto front after 150 generations. These four sub-figures show faster convergence-numbers of generations required to reach optimal Pareto front of QANSGA towards true Pareto front compared to classical NSGA-II.

Table 2 shows comparison metrics for both the algorithms considering performance and efficiency criteria. GD and IGD values are expected to be lower for better algorithm, and HV value is expected to be higher for better algorithm. GD value is 64.21% lower for QANSGA compared to NSGA-II, which shows large improvement in the results after 50 generations. IGD value is 68.11% lower for QANSGA, which again shows better convergence and diversity of solutions in hybrid algorithm. HV value for QANSGA is 8.61% higher than NSGA-II. In all 3 metrics of convergence and diversity of solutions QANSGA dominates. Computational time for both the algorithms is also compared. Classical NSGA-II shows 77% faster performance than QANSGA. QANSGA uses classical NSGA-II for 40 out of 50 generations here. Only for 10 generations quantum-annealing has been used due to current injection rate. Each quantum run takes around 200 ms to produce solutions. For less complicated problems such as ZDT-1, classical NSGA-II shows faster convergence in terms of computational time. In 3 out of 4 comparison metric QANSGA shows better performance except computational time. Usually the run-time of QANSGA is influenced by both classical evolutionary operations and quantum annealing overhead. The dominant additional cost arises from QUBO construction, embedding, and quantum sampling. In particular, embedding time can be significant when using dynamic embedding, as the mapping from logical QUBO variables to the physical quantum hardware must be recomputed for each submission. This overhead is mitigated through fixed embedding in this paper, where a single embedding with equal weights for

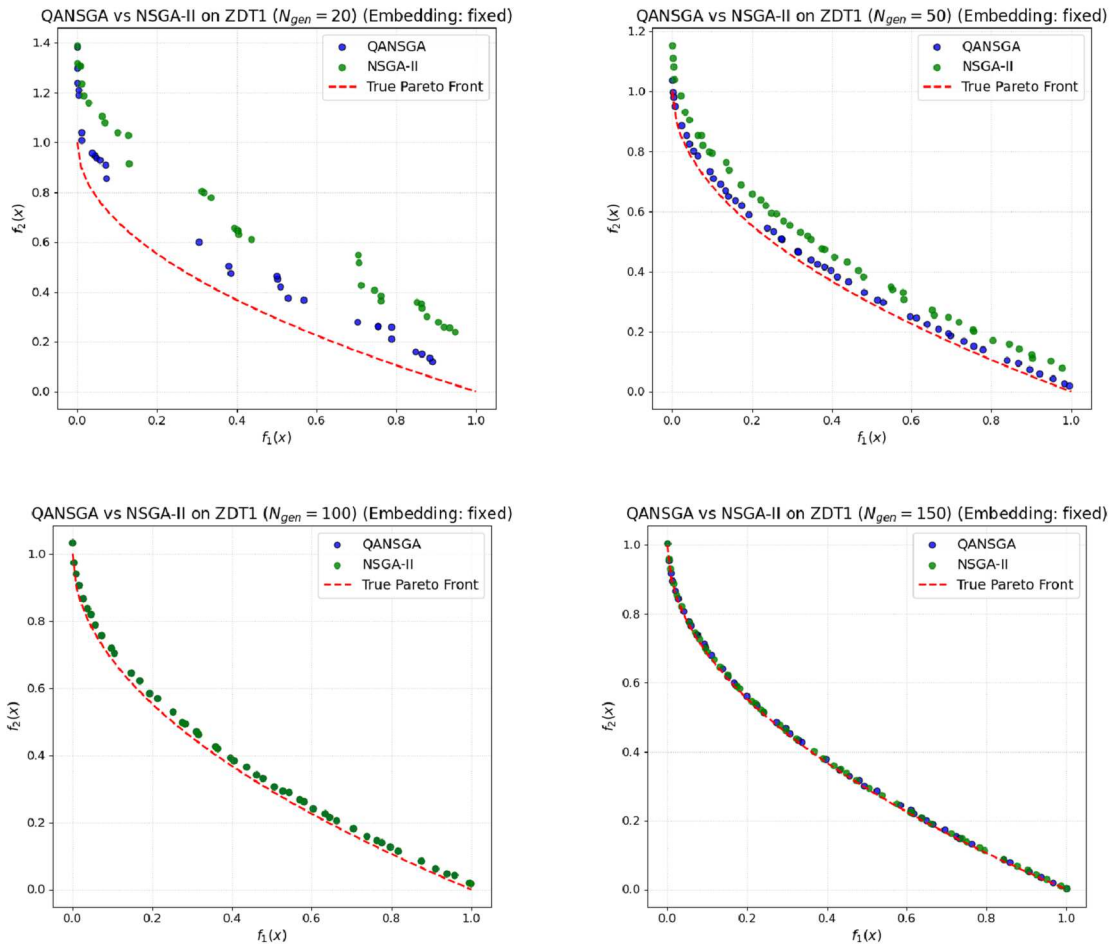

Figure 2. Pareto front comparison of classical NSGA-II and QANSGA with fixed embedding for ZDT-1 problem with 10 variables; injection rate = every 5 generations; 20 total generations (**top-left**), 50 total generations (**top-right**); 100 total generations (**bottom-left**); 150 total generations (**bottom-right**).

Table 2. GD, IGD,HV and computational time comparison of NSGA-II and QANSGA after 50 generations for ZDT-1 problem. Computational time of QANSGA includes classical NSGA-II part + QPU access time excluding embedding.

Algorithm	GD	IGD	HV	Computational time (s)
NSGA-II	0.064	0.065	0.76	0.52
QANSGA	0.023	0.02	0.83	2.29

all the objectives is reused across generations, significantly reducing latency at the expense of some flexibility. In this study the embedding time is not taken into account. The overall run-time of QANSGA is composed of classical evaluation time and QPU access time, with the latter depending on the number of quantum calls. QPU access time includes total of anneal, readout, sampling, QPU core usage and post-processing time excluding embedding.

Figure 3 shows the visual comparison of classical NSGA-II and QANSGA for ZDT-2 problem after different number of generations N_{gen}. ZDT-2 has a non-convex true Pareto front. Algorithm parameters and D-Wave minor embedding is kept same as ZDT-1. Visual evolution of ZDT-2 in figure 3 with increasing number of generations is very different compared to ZDT-1 in figure 6. Due to its non-convex nature, it makes difficult for classical algorithms to converge towards true Pareto front faster, which is evident in figure 7 after 10 and 20 generations. After 50 generations QANSGA already reaches true Pareto front, though the solution lacks in diversity yet - a gap can be seen in QANSGA solutions (blue dots). After 100 generations QANSGA fits well on true Pareto front, while classical NSGA-II is still away from the optimal solutions.

Table 3 shows comparison metrics for both the algorithms for ZDT-2 problem. GD value is 73.16% lower for QANSGA compared to NSGA-II, which shows large improvement in the results after 100 generations. IGD

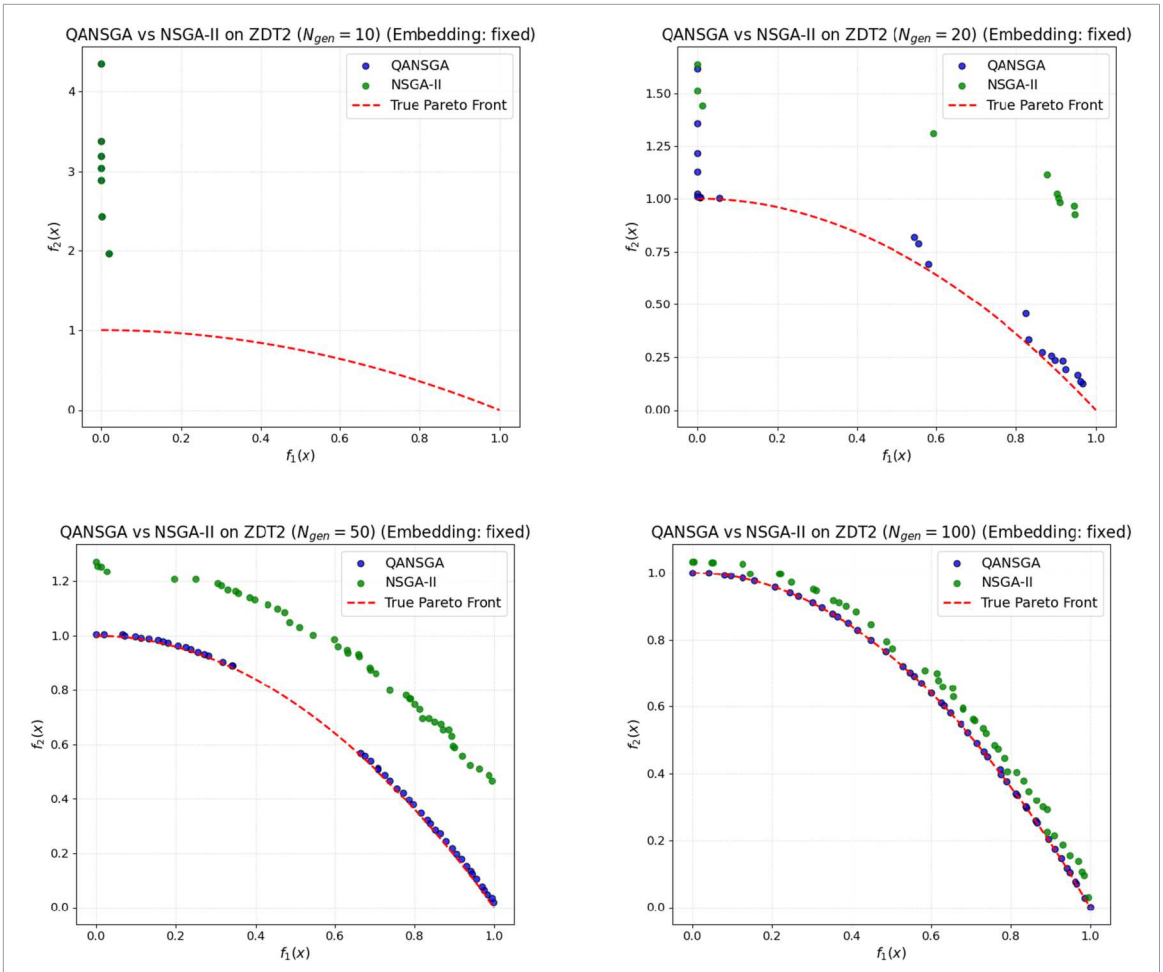

Figure 3. Pareto front comparison of classical NSGA-II and QANSGA with fixed embedding for ZDT-2 problem with 10 design variables; injection rate = every 5 generations; 10 total generations (**top-left**), 20 total generations (**top-right**), 50 total generations (**bottom-left**), 100 total generations (**bottom-right**).

Table 3. GD, IGD,HV and computational time comparison of NSGA-II and QANSGA after 100 generations for ZDT-2 problem. Computational time of QANSGA includes classical NSGA-II part + QPU access time excluding embedding.

Algorithm	GD	IGD	HV	Computational time (s)
NSGA-II	0.035	0.034	0.48	1.232
QANSGA	0.0093	0.004	0.53	2.782

value is 87.53% lower for QANSGA, which again shows better convergence and diversity of solutions in hybrid algorithm. HV value for QANSGA is 10.16% higher than NSGA-II. In all 3 metrics of convergence and diversity of solutions, QANSGA dominates. The computational time for both algorithms is also compared. Classical NSGA-II shows 55% faster performance than QANSGA, which uses classical NSGA-II for 80 out of 100 generations here. For 20 generations quantum-annealing has been used due to the current injection rate. Same as ZDT-1, 3 out of the 4 comparison metric QANSGA shows better performance except computational time considering the ZDT-2 problem.

ZDT-3 problem shows disconnected Pareto fronts due to its periodic nature. The evolution of the solutions for both algorithms is shown in figure 4. The parameters have been kept same as ZDT-1. From first quantum solutions injection after 10 generation we can see the difference between QANSGA and NSGA-II solutions. QANSGA solutions starts converging towards true Pareto front quickly. After 50 generations, QANSGA solutions reach true Pareto front whereas NSGA-II solutions are still not completely lying on the true front. Table 4 shows comparison metrics for both algorithms for the ZDT-3 problem. GD value is 20.23% lower for QANSGA compared to NSGA-II. IGD value is 56.63% lower for QANSGA, which again shows better convergence and

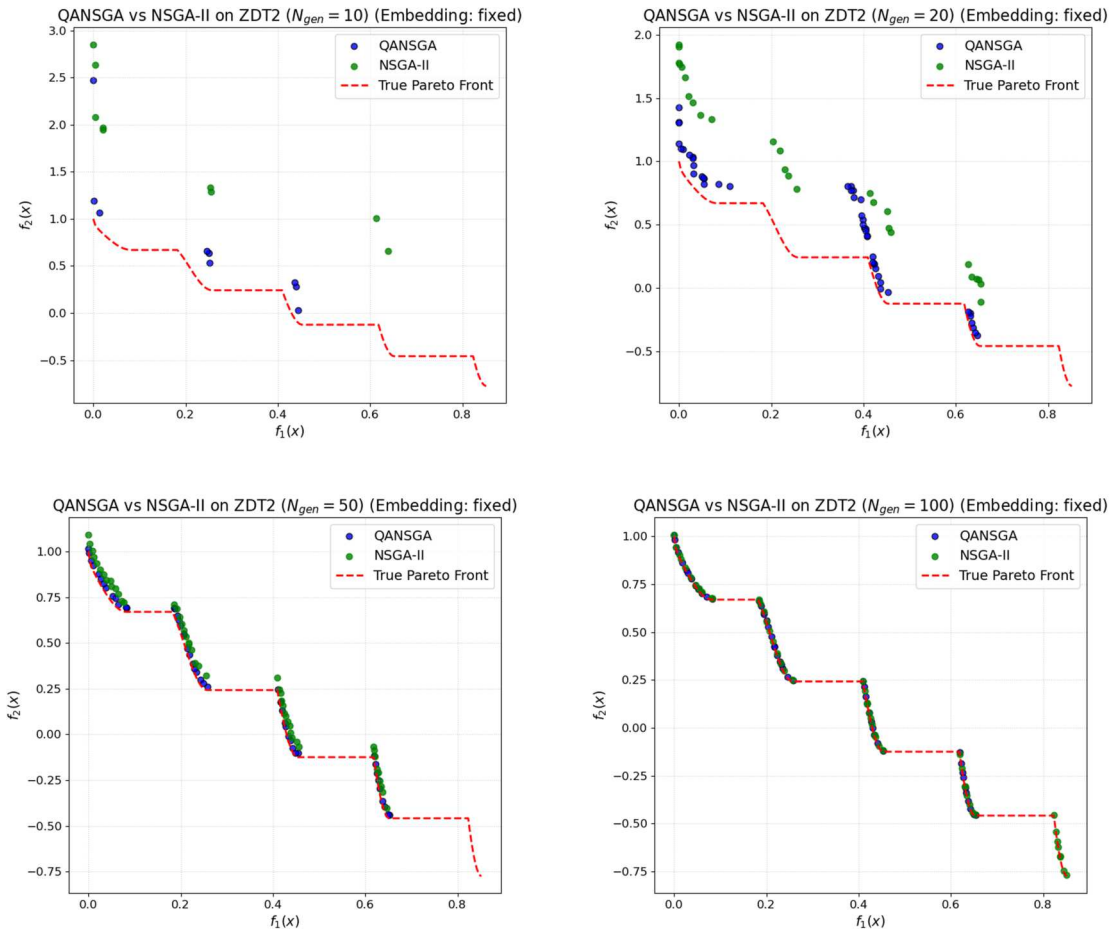

Figure 4. Pareto front comparison of classical NSGA-II and QANSGA with fixed embedding for ZDT-3 problem with 10 design variables; injection rate = every 5 generations; 10 total generations (**top-left**), 20 total generations (**top-right**), 50 total generations (**bottom-left**), 100 total generations (**bottom-right**).

Table 4. GD, IGD,HV and computational time comparison of NSGA-II and QANSGA after 50 generations for ZDT-3 problem. Computational time of QANSGA includes classical NSGA-II part + QPU access time excluding embedding.

Algorithm	GD	IGD	HV	Computational time (s)
NSGA-II	0.087	0.020	0.93	0.51
QANSGA	0.069	0.008	0.96	2.13

diversity of solutions in hybrid algorithm. HV value for QANSGA is 4.29% higher than NSGA-II. Classical NSGA-II shows 76% faster performance than QANSGA.

ZDT-4 problem has a convex Pareto-optimal set. There exist many local Pareto-optimal solutions in this problem. Therefore, algorithms can easily get stuck in a local optimum. Figure 5 shows the visual evolution of QANSGA and NSGA-II solutions. We can see that it takes higher number of generations for both the algorithms to converge to true Pareto front compared to previous ZDT problems. QANSGA gets stuck in local optima after 500 generations as seen in figure 5. It needs more generations to get out of the local optima and converge towards the true Pareto front. This happens due to quadratic surrogate used in QANSGA. As D-Wave quantum computer can only solve QUBO, we have converted the actual problem into quadratic approximation as shown in algorithm 1. Which misses complex functionalities of original ZDT-4 and requires more number of classical NSGA-II generations to converge.

Table 5 shows the metrics comparison for both algorithms. Here, it is evident that classical NSGA-II performs better than QANSGA in all the metrics.

Figure 6 shows the visual evolution of NSGA-II and QANSGA solutions after certain number of generations for the OSY problem. As explained in section 2.1.2, OSY problem consists of 6 variables and 6 constraints.

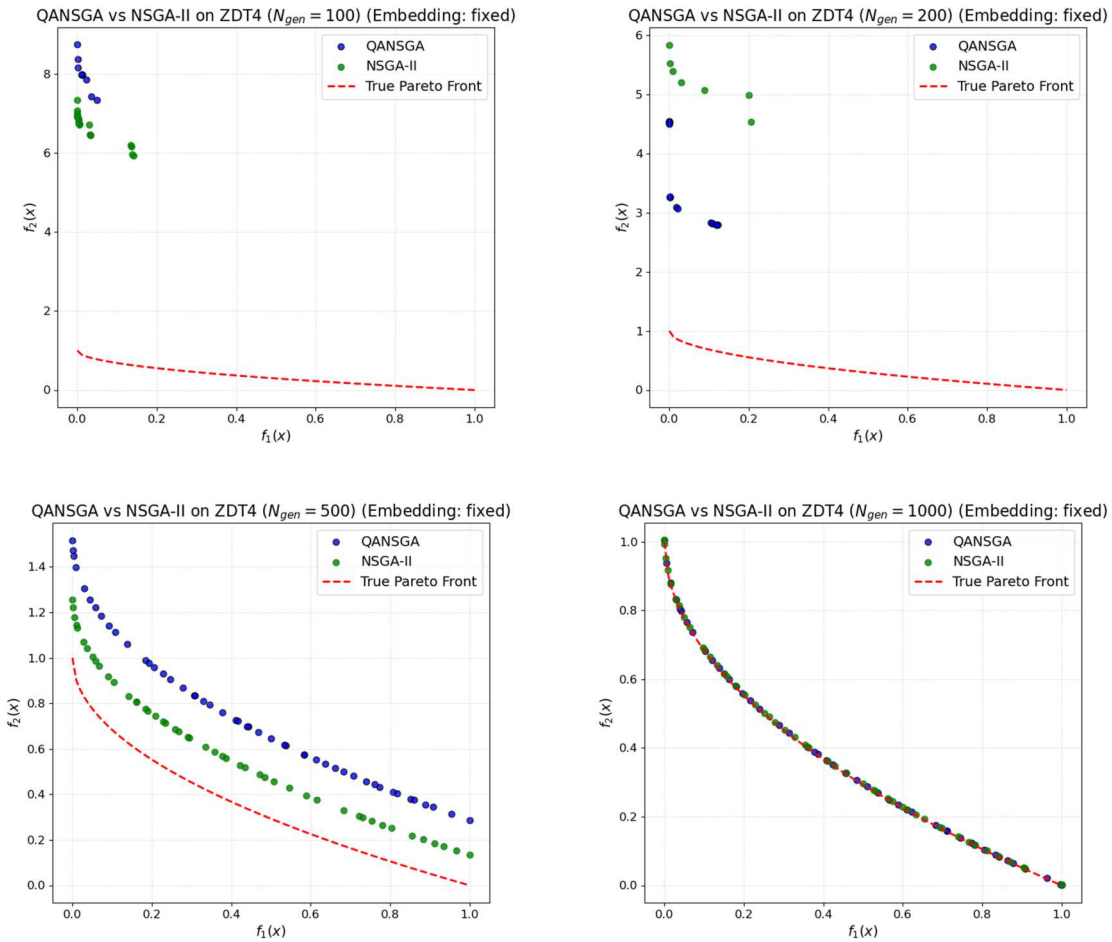

Figure 5. Pareto front comparison of classical NSGA-II and QANSGA with fixed embedding for ZDT-4 problem with 10 design variables; injection rate = every 5 generations; 100 total generations (**top-left**), 200 total generations (**top-right**), 500 total generations (**bottom-left**), 1000 total generations (**bottom-right**).

Table 5. GD, IGD,HV and computational time comparison of NSGA-II and QANSGA after 500 generations for ZDT-4 problem. Computational time of QANSGA includes classical NSGA-II part + QPU access time excluding embedding.

Algorithm	GD	IGD	HV	Computational time (s)
NSGA-II	0.13	0.126	0.67	9.68
QANSGA	0.26	0.269	0.49	11.75

OSY problem needs only 6 design variables for classical NSGA-II and QANSGA. All the other algorithmic parameters have been kept same as ZDT functions. During all the evolution process, QANSGA solutions dominate NSGA-II ones as shown in figure 6. After 200 generations, QANSGA solutions reach true Pareto front, while classical NSGA-II solutions (green dots) could not completely overlap true Pareto front. Table 6 shows the metrics comparison for NSGA-II and QANSGA for the OSY problem. QANSGA shows better performance in GD, IGD and HV. Same as for all ZDT functions, NSGA-II shows better performance in computational time.

Figure 7 shows the visual evolution of NSGA-II and QANSGA solutions after certain number of generations for TNK problem. As explained in section 2.1.3, TNK problem consists of 2 variables and 2 constraints. The Pareto-optimal set of TNK is disconnected. Since the Pareto-optimal solutions lie on a nonlinear constraint surface, an optimization algorithm may have difficulty in finding a good spread of solutions across all of the discontinuous Pareto-optimal sets. The TNK problem needs only 2 design variables for classical NSGA-II and QANSGA. All the other algorithmic parameters have been kept the same as ZDT functions except the quantum injection rate, which is kept as 2. During all the evolution process QANSGA solutions dominate NSGA-II ones as shown in figure 7. Table 7 shows the metrics comparison for NSGA-II and QANSGA for the

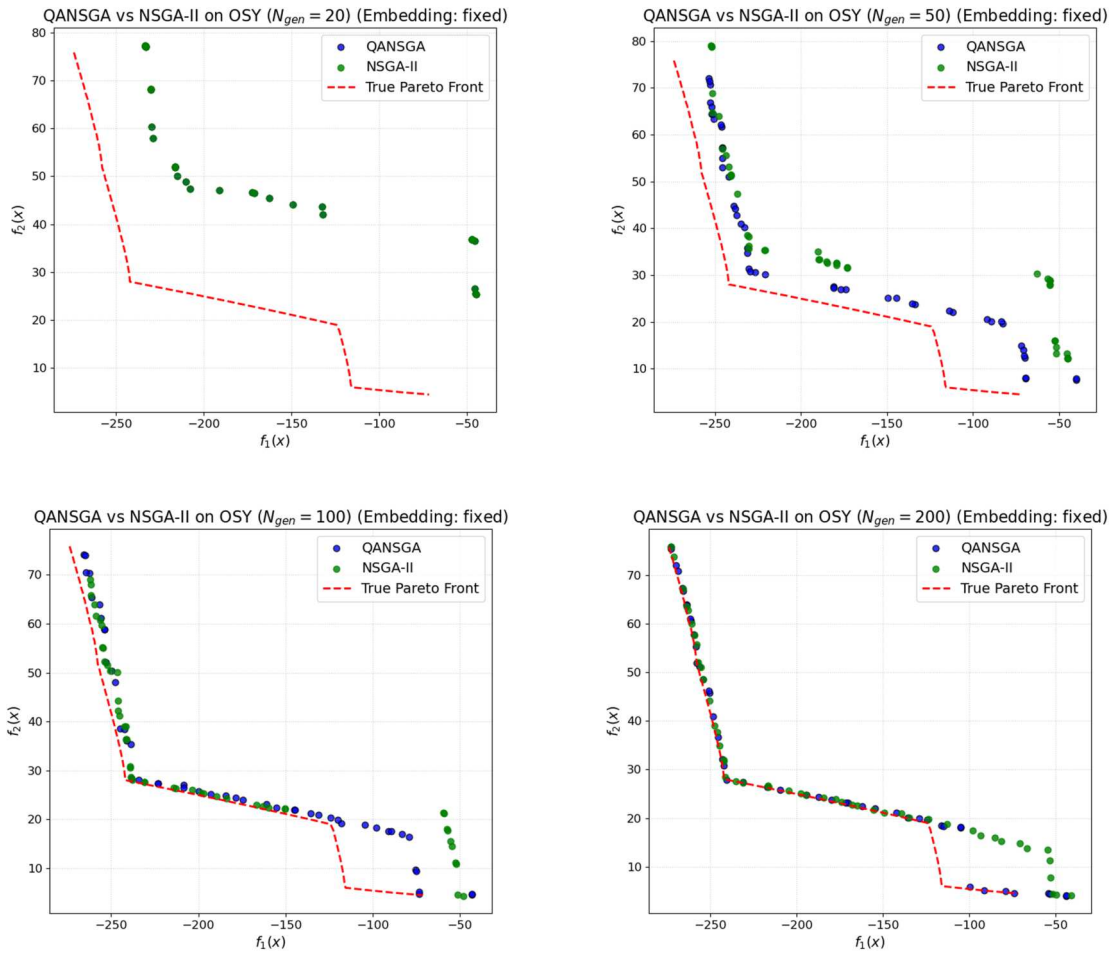

Figure 6. Pareto front comparison of classical NSGA-II and QANSGA with fixed embedding for OSY problem with 6 design variables; injection rate = every 5 generations; 20 total generations (**top-left**), 50 total generations (**top-right**), 100 total generations (**bottom-left**), 200 total generations (**bottom-right**).

Table 6. GD, IGD,HV and computational time comparison of NSGA-II and QANSGA after 200 generations for OSY problem. Computational time of QANSGA includes classical NSGA-II part + QPU access time excluding embedding.

Algorithm	GD	IGD	HV	Computational time (s)
NSGA-II	10.36	6.97	10966.75	1.66
QANSGA	4.91	5.71	11505.68	3.55

TNK problem. QANSGA shows better performance in GD, IGD and HV. Same as all ZDT functions and OSY function, NSGA-II shows better performance in computational time for TNK problem as well.

The next section discusses about the influence of the number of variables on the computational time of NSGA-II and QANSGA. So far the number of variables were kept constant for each individual problem and only the number of generations were changed.

3.2. Effect of increasing number of variables on computational time and quality of the Pareto front

Figure 8 shows the computational time comparison between NSGA-II and QANSGA for different number of variables and different number of generations for ZDT-1 problem.

The overall pattern of all the graphs from figure 8 shows that the NSGA-II computational time increases faster than QANSGA with the increase performs faster compared to QANSGA for most number of variables. But the time difference between both the algorithms decreases with the increase in the number of variables. For 80 number of variables, after 100 generations QANSGA computational time is lesser than NSGA-II. For 200 generations this happens after 30 variables. QANSGA shows faster performance just after 30 variables when the

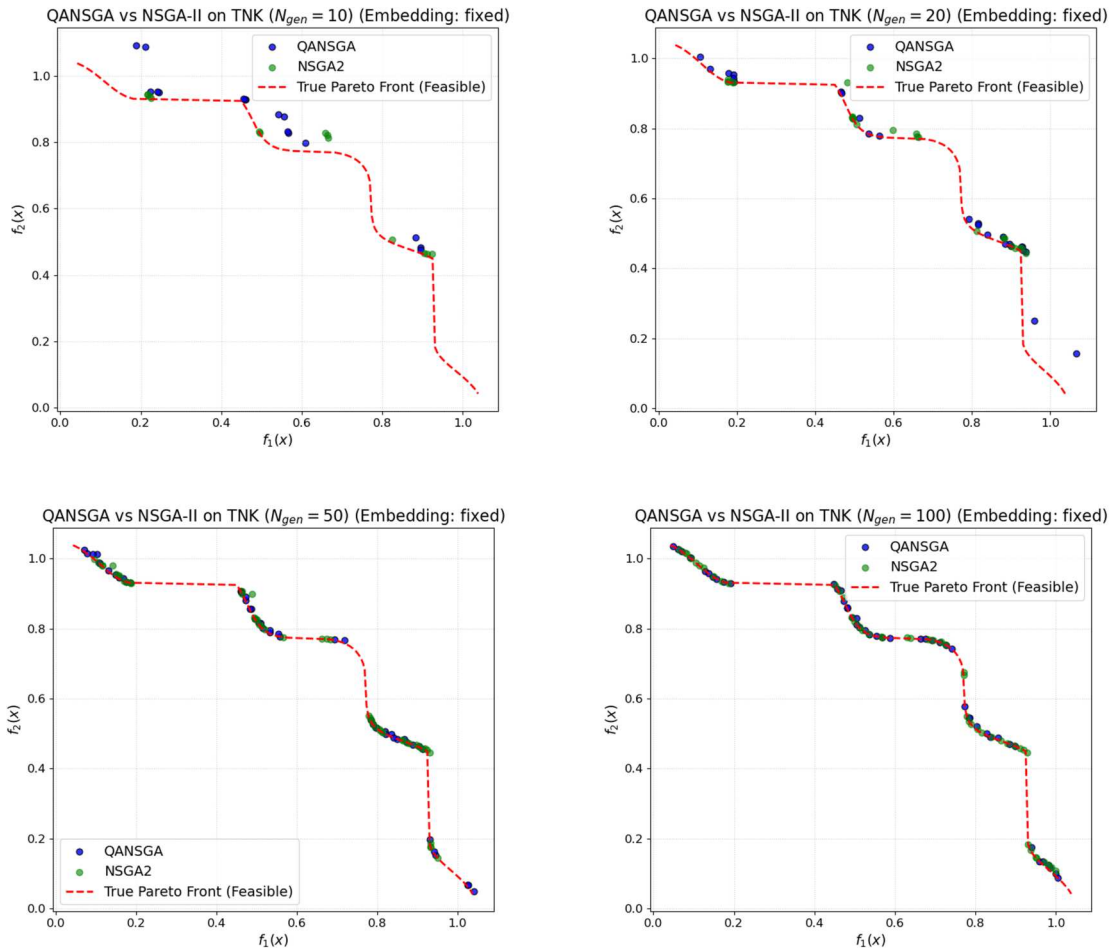

Figure 7. Pareto front comparison of classical NSGA-II and QANSGA with fixed embedding for TNK problem with 2 design variables; injection rate = every 2 generations; 10 total generations (**top-left**), 20 total generations (**top-right**), 50 total generations (**bottom-left**), 100 total generations (**bottom-right**).

Table 7. GD, IGD,HV and computational time comparison of NSGA-II and QANSGA after 50 generations for TNK problem. Computational time of QANSGA includes classical NSGA-II part + QPU access time excluding embedding.

Algorithm	GD	IGD	HV	Computational time (s)
NSGA-II	0.027	0.0049	0.49	1.66
QANSGA	0.014	0.0047	0.51	3.58

number of generations is increased to 200. And for further increase in number of variables the gap between both computational times increases, because QANSGA scales linearly where as NSGA-II scales with higher degree of polynomial. For generation 500, 1000 and 2000 QANSGA performs better for almost all number of variables, which shows that as the problem size increases, NSGA-II struggles to scale linearly. After 500, 1000 and 2000 generations for 80 variables ZDT-1 problem QANSGA shows 29%, 44% and 63% faster convergence time respectively compared to NSGA-II. QPU on D-Wave has physical limits of qubits embedding, that is why the highest number of variables have been restricted to 80.

Figure 9 shows the Pareto front comparison for NSGA-II and QANSGA solutions after 1000 (**Left**) and 5000 (**Right**) generations for ZDT-4 problem. Here, this problem consists of 30 variables, which is comparatively larger problem. Figure 5 showed the comparison for ZDT-4 generations for only 10 variables and up to 1000 generations. From figure 5 we could see that NSGA-II performed better in all the metrics compared to QANSGA. But when we increased the number of variables to 30, QANSGA started dominating for larger number of generations. We can see in figure 9 that even-though after 1000 generation NSGA-II is closer to true Pareto front, QANSGA shows better diversity and spread visually. As we increase the number of generations to

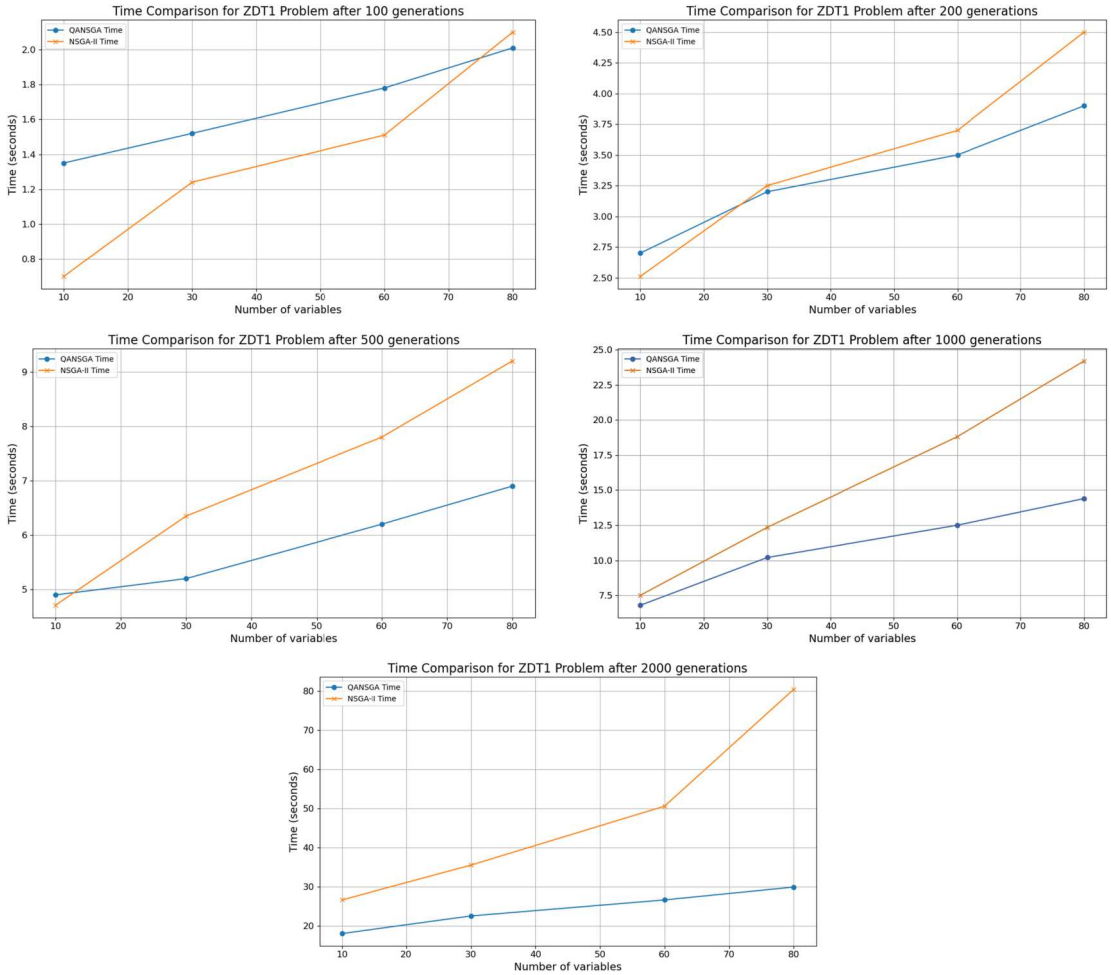

Figure 8. Computational time comparison for classical NSGA-II and QANSGA with different number of variables for ZDT-1 Problem.

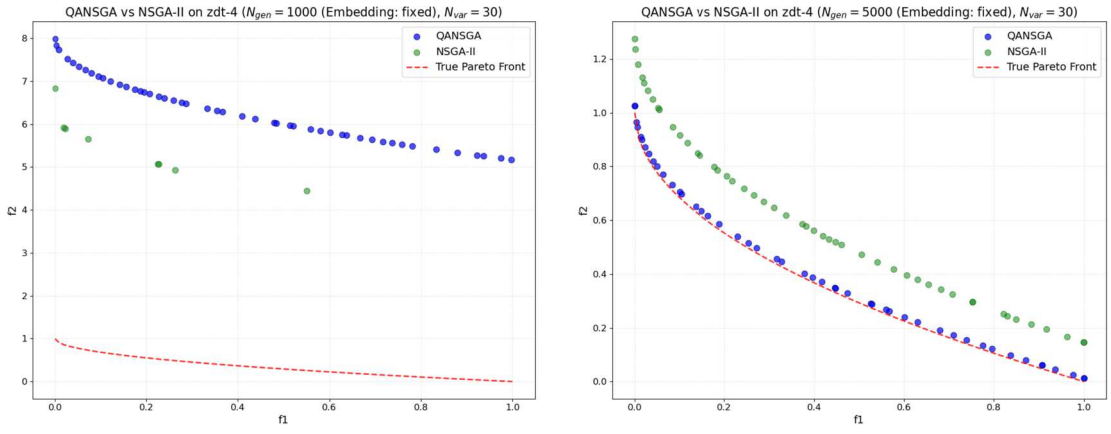

Figure 9. Pareto front comparison after 1000 and 5000 generations for QANSGA and NSGA-II solutions for ZDT-4 problem with 30 variables.

5000, QANSGA dominates and reaches true Pareto front. NSGA-II has still not converged to true Pareto front after 5000 generations.

Table 8 shows the metric comparison between NSGA-II and QANSGA. QANSGA is 87%, 89% and 27% better performance in GD, IGD and HV respectively. Even in computational time QANSGA is 21% faster than NSGA-Ii after 5000 generations. This shows the faster and efficient performance of QANSGA over NSGA-II for larger problems.

Table 8. GD, IGD,HV and computational time comparison of NSGA-II and QANSGA after 5000 generations for ZDT-4 problem. Computational time of QANSGA includes classical NSGA-II part + QPU access time excluding embedding.

Algorithm	GD	IGD	HV	Computational time (s)
NSGA-II	0.14	0.13	0.66	53
QANSGA	0.017	0.014	0.84	42

4. Conclusion and outlook

This paper presented novel quantum assisted GA, named as Quantum Assisted Non-dominated Sorting algorithm (QANSGA) to optimize MOO problems. This paper explores the performance of QANSGA for 6 different benchmark MOO problems such as ZDT, OSY and TNK problems. The comparison of QANSGA results with classical NSGA-II showed that QANSGA performs better for metrics such as GD, IGD and HV for majority of the problems for all problem sizes. QANSGA lacks in computational effort compared to NSGA-II for smaller problems. As the problem size increases in terms of number of variables, QANSGA outperforms NSGA-II even in computational effort, which shows robustness of QANSGA. The visual comparison of both the algorithms also showed better convergence and diversity of QANSGA compared to NSGA-II. The metric comparison showed that QANSGA outperforms classical NSGA-II in terms of convergence and diversity by 33% and 42% respectively on an average. The drawback of QANSGA is the physical limit of the state-of-the-art quantum annealers. Due to physical limit of the number of qubits, we had to restrict the problem size maximum up to 80 variables. Classical NSGA-II can be used even for larger problems. This limitation requires the QANSGA to be used strategically to use quantum resources efficiently. This is shown in our further research and use of QANSGA in [34], which also shows the use of QANSGA in real optimization problem of energy system optimization.

Building upon the robust performance of QANSGA, future research will focus on overcoming current hardware constraints and expanding the algorithm's versatility. As QA technology matures and qubit counts increase, we aim to scale QANSGA to high-dimensional problems exceeding the current 80-variable limit, potentially unlocking a definitive quantum advantage for massive-scale industrial optimizations. There has already been research advancement in mitigating the hardware constraints in gate-based and quantum annealing-based quantum computers [35, 36]. Furthermore, investigating hybrid quantum–classical architectures could allow for a more strategic distribution of computational loads, using quantum resources specifically for the most complex non-convex regions of the search space. Future iterations will also explore the integration of adaptive penalty functions to better handle highly constrained multi-objective landscapes. Beyond the benchmark optimizations explored in current work, the application of QANSGA to diverse fields such as industrial design, logistics, and drug discovery remains a promising frontier for demonstrating the algorithm's generalizability and efficiency.

Acknowledgments

The authors gratefully acknowledge the Jülich Supercomputing Centre (https://www.fz-juelich.de/ias/jsc) for funding this project by providing computing time on the D-Wave Advantage™ System JUPSI through the Jülich UNified Infrastructure for Quantum computing (JUNIQ).

Conflict of interest

The authors declare no conflicts of interest.

Data availability statement

The data cannot be made publicly available upon publication because no suitable repository exists for hosting data in this field of study. The data that support the findings of this study are available upon reasonable request from the authors.

Funding

This paper was funded internally by German Aerospace Center.

Author contributions

Rushit Kansara 0000-0001-9819-0321
Conceptualization (lead), Data curation (lead), Formal analysis (lead), Methodology (lead), Software (lead), Validation (lead), Visualization (lead), Writing – original draft (lead)

María Isabel Roldán Serrano 0000-0002-0663-6048
Project administration (lead), Supervision (supporting), Writing – review & editing (supporting)

Martin Bähr
Conceptualization (supporting), Investigation (supporting), Methodology (supporting), Software (supporting), Supervision (supporting)

References

[1] Chase N, Rademacher M, Goodman E, Averill R and Sidhu R 2021 *A Benchmark Study of Multi-Objective Optimization Methods* (HEEDS MDO)
[2] Deb K, Pratap A, Agarwal S and Meyarivan T 2002 A fast and elitist multiobjective genetic algorithm: Nsga-ii *IEEE Trans. Evol. Comput.* **6** 182–97
[3] Kukkonen S and Deb K 2006 A fast and effective method for pruning of non-dominated solutions in many-objective problems *Parallel Problem Solving From Nature - ppsn ix* ed T P Runarsson *et al* (Springer) pp 553–62
[4] Blank J and Deb K 2020 A running performance metric and termination criterion for evaluating evolutionary multi- and many-objective optimization algorithms *IEEE Congress on Evolutionary Computation (CEC)* pp 1–8
[5] van Veldhuizen D A 1998 Evolutionary Computation and Convergence to a Pareto Front, https://api.semanticscholar.org/CorpusID:18585705
[6] Hamdy M, Nguyen A-T and Hensen J L M 2016 A performance comparison of multi-objective optimization algorithms for solving nearly-zero-energy-building design problems *Energy Build.* **121** 57–71
[7] Kalyanmoy D 2001 *Multi-objective Optimization Using Evolutionary Algorithms* (John Wiley & Sons, Inc.)
[8] Schott J R 1995 Fault tolerant design using single and multicriteria genetic algorithm optimization, www.hdl.handle.net/1721.1/11582
[9] Guerreiro A P, Fonseca C M and Paquete L 2021 The hypervolume indicator: computational problems and algorithms *ACM Comput. Surv.* **54** 1–42
[10] Zitzler E, Thiele L, Laumanns M, Fonseca C M and da Fonseca V G 2003 Performance assessment of multiobjective optimizers: an analysis and review *IEEE Trans. Evol. Comput.* **7** 117–32
[11] Grover L K 1997 Quantum mechanics helps in searching for a needle in a haystack *Phys. Rev. Lett.* **79** 325–8
[12] Shor P W 1999 Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer *SIAM J. Comput.* **26** 1484–509
[13] Nielsen M A and Chuang I L 2010 *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press)
[14] Madsen L S *et al* 2022 Quantum computational advantage with a programmable photonic processor *Nature* **606** 75–81
[15] D-Wave Quantum Inc 2018 D-wave system documentation, https://docs.dwavesys.com/docs/latest/index.html
[16] Hauke P, Katzgraber H G, Lechner W, Nishimori H and Oliver W D 2020 Perspectives of quantum annealing: methods and implementations *Rep. Prog. Phys.* **83** 054401
[17] Zhao Z, Fan L and Han Z 2022 Hybrid quantum Benders' decomposition for mixed-integer linear programming *IEEE Wireless Communications and Networking Conference* (IEEE) pp 2536–40
[18] Ajagekar A and You F 2019 Quantum computing for energy systems optimization: challenges and opportunities *Energy* **179** 76–89
[19] Van der Linde S G, van der Schoot W and Phillipson F 2023 Efficient quantum solution for the constrained tactical capacity problem for distributed electricity generation *Innovations for Community Services* **1876** 203–21
[20] Ekstrom L, Wang H and Schmitt S 2025 Variational quantum multiobjective optimization *Phys. Rev. Res.* **7** 023141
[21] Kotil A, Pelofske E, Riedmüller S, Egger D J, Eidenbenz S, Koch T and Woerner S 2025 Quantum approximate multi-objective optimization *Nature Computational Science* **5** 1168–77
[22] King A D 2025 Multi-objective Optimization by Quantum Annealing arXiv:2511.01762
[23] King J, Mohseni M, Bernoudy W, Fréchette A, Sadeghi H, Isakov S V, Neven H and Amin M H 2019 Quantum-assisted Genetic Algorithm arXiv:1907.00707
[24] Dollen D V, Yarkoni S, Weimer D, Neukart F and Bäck T 2022 Quantum-enhanced Selection Operators for Evolutionary Algorithms arXiv:2206.10743
[25] Wang Y, Wang W, Ahmad I and Tag-Eldin E 2022 Multi-objective quantum-inspired seagull optimization algorithm *Electronics* **11** 1834
[26] Deb K 1999 Multi-objective genetic algorithms: problem difficulties and construction of test problems *Evol. Comput.* **7** 205–30
[27] Zitzler E, Deb K and Thiele L 2000 Comparison of multiobjective evolutionary algorithms: empirical results *Evolutionary Computation* **8** 173–95
[28] Osyczka A and Kundu S 1995 A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm *Struct. Optim.* **10** 94–9

[29] Tanaka M, Watanabe H, Furukawa Y and Tanino T 1995 Ga-based decision support system for multicriteria optimization *IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century* 2, 1556–61
[30] Nielsen M A and Chuang I L 2010 *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press)
[31] Farhi E, Goldstone J, Gutmann S and Sipser M 2000 Quantum Computation by Adiabatic Evolution arXiv:quant-ph/0001106
[32] Kochenberger G, Hao J-K, Glover F, Lewis M, Lü Z, Wang H and Wang Y 2014 The unconstrained binary quadratic programming problem: a survey *Journal of Combinatorial Optimization* **28** 58–81
[33] Jones D R, Schonlau M and Welch W J 1998 Efficient global optimization of expensive black-box functions *J. Global Optim.* **13** 455–92
[34] Kansara R, Kyriakidis L and Serrano M I R 2025 Robust quantum-assisted discrete design of industrial smart energy utility systems with long-term operational uncertainties: a case study of a food and cosmetic industry in germany *Energies* **18** 4258
[35] Cao C and Eisert J 2026 Measurement-driven quantum advantages in shallow circuits *Phys. Rev. Lett.* **136** 080601
[36] Cattelan M, Bennett J, Yarkoni S and Lechner W 2025 Scalable Embedding of Parity Constraints in Quantum Annealing Hardware *Phys. Rev. A* **111** 012345