Cost-Minimal Reliability Allocation and Robustness of an Edge–Cloud Network Using the Coati and Arithmetic Optimization Algorithms

Authors Names

Hatem Kareem Sulaiman [a,*]

Karrar Habeeb Hashim [b]

Publication date: 17 / 11 /2026

Keywords: Reliability allocation, Cost minimization, Cost–reliability frontier, Robustness, Coati Optimization Algorithm, Arithmetic Optimization Algorithm, Edge–cloud network, Exact reliability polynomial.

ABSTRACT

The cost of meeting a reliability requirement is rarely visible in weighted-objective allocation studies, and the robustness of the resulting design is almost never examined. This paper poses reliability allocation on a twelve-node heterogeneous edge–cloud network (QSDR-ECRN) as cost minimization under an explicit reliability constraint, evaluated through the exact 69-term reliability polynomial of the network, which equals 0.998100 at the as-deployed component reliabilities. Two recent algorithms that have not been applied to network reliability allocation, the Coati Optimization Algorithm (COA) and the Arithmetic Optimization Algorithm (AOA), were implemented in Python and compared over 30 independent runs for each of five reliability targets from 0.9990 to 0.99995, with identical budgets of about 12,000 fitness evaluations, and against a constrained gradient-based reference solution. The cost-minimal designs form a smooth frontier in which only the storage node (R_7) and the database node (R_{12}) are upgraded, as exact Birnbaum importance suggests. COA stayed close to the reference optimum, with a mean cost gap of 1.4% at the 0.9997 target and between 0.16% and 2.78% over all targets, whereas AOA settled consistently above it (12.6% at 0.9997, 3.4% to 16.0% overall); the difference was significant at every target (Wilcoxon rank-sum $p < 0.001$) and for every budget and population size tested. A design optimized precisely for 0.9997 never meets that requirement in practice, a design targeted at 0.9998 meets it in only 1.5% of draws, and a design targeted at 0.9999 meets it in 99.4% of draws for 36.7% more cost, according to a Monte Carlo robustness study where the failure probability of every component rises by up to 50%. These numbers provide designers with a measurable cost for robustness.

1. Introduction

It is rare to ask a network designer to create the most dependable design possible. A design that satisfies a certain reliability criterion at the lowest feasible cost is typically requested, followed by an inquiry into what the cost would be if the requirement were tightened. The formal form of this dilemma is reliability allocation [1-3], which is still a concern today due to the size, partial redundancy, and high cost of over-provisioning of current service chains [4, 5]. Reliability and cost are usually combined into a single weighted objective or maximized within a cost cap in metaheuristic allocation studies, such as the author's previous work [6–11]. These decisions are valid, but each weight setting only produces one point. As a result, the relationship between a target and its cost is concealed, and the final design is only evaluated based on its nominal component reliabilities. Although this is a common occurrence in service, nothing is mentioned about how it responds when parts are marginally subpar. The Quad-Stage Diagonal-Redundant Edge-Cloud Reliability Network (QSDR-ECRN), a twelve-node edge-cloud network, is examined in this study from three perspectives. First, allocation is posed as pure cost minimization with an explicit reliability constraint, which a constrained gradient method can solve reliably and which therefore supplies a true benchmark for the metaheuristics. Second, sweeping the target traces the cost–reliability frontier. Third, a Monte Carlo

[a,*]General Directorate of Education in Babylon, Ministry of Education, Iraq, E-Mail: hatim19861986@gmail.com (corresponding author)

[b]General Directorate of Education in Babylon, Ministry of Education, Iraq, E-Mail: teacher89karrar@gmail.com

experiment with inflated component failure probabilities explores what a target-optimized design actually achieves and how much more money can be spent on robustness. The optimizers compared are the Coati Optimization Algorithm (COA) [12] and the Arithmetic Optimization Algorithm (AOA) [13], two recent algorithms of very different construction that, to our knowledge, have not been reported on network reliability allocation. Their budgets are matched by the number of fitness evaluations rather than by iterations, since the two methods spend evaluations at different rates.

The contributions are as follows:

(i) a cost-minimization formulation with a hard reliability constraint, evaluated through the exact 69-term polynomial of the QSDR-ECRN, with a constrained gradient-based benchmark for judging metaheuristic results;

(ii) the cost–reliability frontier of the network for five targets, with the optimal allocation at each point and its relation to exact Birnbaum importance;

(iii) a robustness analysis under inflated component failure probabilities, which quantifies the extra cost needed to keep a reliability requirement (the price of robustness);

(iv) executable Python implementations of COA and AOA with flowcharts, per-algorithm tables, and an evaluation-matched comparison over 30 independent runs, five targets, three evaluation budgets and three population sizes.

2. Related Work

Numerous surveys have been conducted on dependability allocation under cost and reliability restrictions [1, 3, 14], and current evaluations emphasize that traditional weighting and apportionment techniques are based on assumptions that are rarely true in real-world scenarios [4]. Particle swarm optimization and comparisons of the two on complex networks, aerospace, and neural-network architectures [9, 10, 16, 17] followed genetic algorithms as one of the first metaheuristics used to solve the problem [15]. The goal necessitates a structure function, which is often obtained for networks using path sets and the sum-of-disjoint-products approach [18]. Component importance metrics, like Birnbaum's [19], show where improvements are beneficial. Reliability is increasingly maximized in conjunction with scheduling and resource allocation in edge and cloud computing, such as through hybrid multi-objective approaches for resilience [5]. The family of swarm algorithms used for such tasks has grown quickly. The Grey Wolf Optimizer [20] and the White Shark Optimizer [21] are among the better known, and the Puma Optimizer [22] and quantum-inspired variants [23, 24] have also been proposed. The Coati Optimization Algorithm [12] imitates two behaviours of coatis, hunting iguanas and escaping predators, and combines a population-guided phase with a shrinking local search. The Arithmetic Optimization Algorithm [13] is a mathematics-inspired method in which the four basic arithmetic operators generate candidates around the best solution under two adaptive coefficients. Both have been evaluated mostly on benchmark functions and engineering design problems. Because no search method can be expected to be best on every problem [25,26], an evidence-based comparison on a concrete, exactly evaluated reliability problem is a useful addition.

3. The QSDR-ECRN Network and Its Exact Reliability

The network is the one shown in Figure 1: a data source is joined to a service destination through twelve heterogeneous nodes arranged in four cascaded stages, with each node reaching only some

nodes of the next stage through primary and backup links. We refer to it as the Quad-Stage Diagonal-Redundant Edge–Cloud Reliability Network (QSDR-ECRN). The twelve components and their baseline reliabilities are given in Table 1 together with the general symbols R_i used throughout.

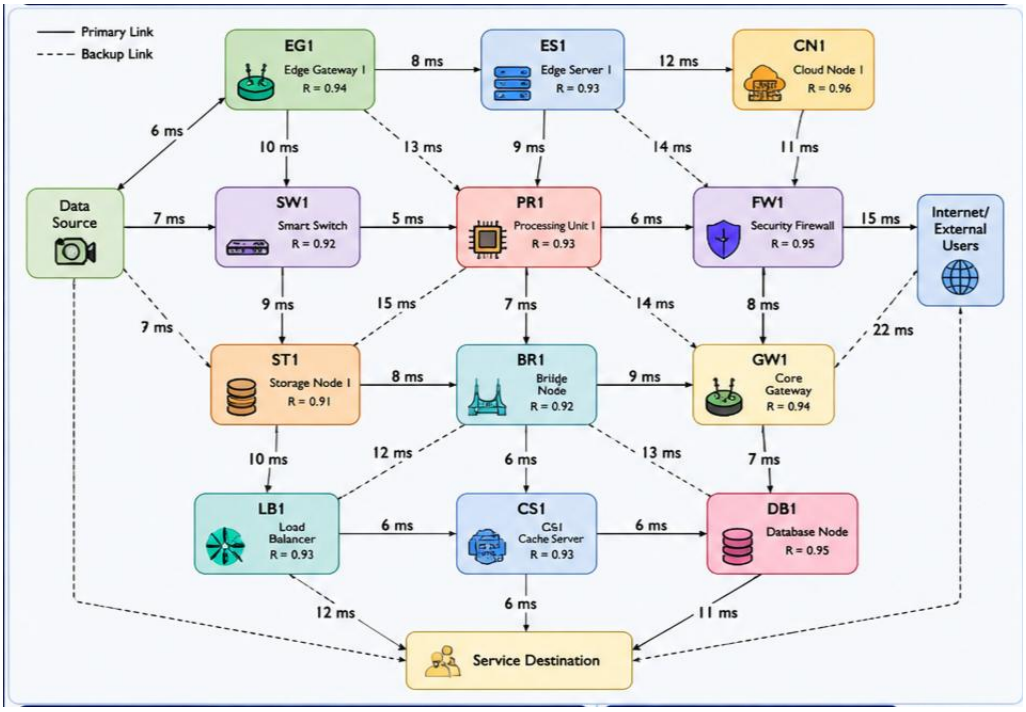


Figure 1. Topology of the QSDR-ECRN. Solid lines are primary links and dashed lines are backup links; link labels are latencies in milliseconds.

Table 1. Components of the QSDR-ECRN, general symbols and baseline reliabilities R_{i0}.

Symbol	Node ID	Device type	Stage	Baseline reliability
R_1	EG1	Edge Gateway	1 – Edge access	0.94
R_2	ES1	Edge Server	1 – Edge access	0.93
R_3	CN1	Cloud Node	1 – Edge access	0.96
R_4	SW1	Smart Switch	2 – Processing/security	0.92
R_5	PR1	Processing Unit	2 – Processing/security	0.93
R_6	FW1	Security Firewall	2 – Processing/security	0.95
R_7	ST1	Storage Node	3 – Routing/storage	0.91
R_8	BR1	Bridge Node	3 – Routing/storage	0.92
R_9	GW1	Core Gateway	3 – Routing/storage	0.94
R_{10}	LB1	Load Balancer	4 – Service delivery	0.93
R_{11}	CS1	Cache Server	4 – Service delivery	0.93
R_{12}	DB1	Database Node	4 – Service delivery	0.95

An exhaustive search of the directed graph that encodes all primary and backup links yields 61 simple source-to-destination routes. Many of them contain shorter routes as subsets, and when the routes are reduced by set inclusion only ten minimal path sets remain (Table 2). Evaluating the structure function on all 2^{12} = 4096 component states and applying Möbius inversion over the subset lattice gives the unique multilinear reliability polynomial, which has 69 non-zero terms with coefficients +1 or −1 and is listed in Appendix A. The same expression is obtained from a sum-of-disjoint-products derivation [18]. At the baseline reliabilities of Table 1 it gives

$$R_s(\mathbf{R}_0) = 0.998100, \tag{1}$$

and the corresponding unreliability is U_0 = 0.001900. With every component at R_{max} = 0.999 the system reliability would be 0.99999999, which fixes the largest target that can be demanded.

Table 2. Minimal path sets of the QSDR-ECRN (general symbols).

No.	Minimal path set	Size
1	{ R_7, R_{10} }	2
2	{ R_7, R_{11} }	2
3	{ R_4, R_8, R_{11} }	3
4	{ R_4, R_8, R_{12} }	3
5	{ R_7, R_8, R_{12} }	3
6	{ R_1, R_5, R_8, R_{11} }	4
7	{ R_1, R_5, R_8, R_{12} }	4
8	{ R_1, R_5, R_9, R_{12} }	4
9	{ R_4, R_5, R_9, R_{12} }	4
10	{ R_1, R_2, R_6, R_9, R_{12} }	5

The Birnbaum importance [19] of component j, $I_j = \partial R_s/\partial R_j$, is available in closed form because the polynomial is multilinear. Table 3 gives the exact values at the baseline. The storage node R_7 and the database node R_{12} dominate, and the cloud node R_3 has zero importance because every route through it is dominated by a shorter backup route.

Table 3. Exact Birnbaum importance of the QSDR-ECRN components at the baseline reliabilities.

Symbol	Node	Stage	Birnbaum I_j	Normalized (%)
R_7	ST1	3	0.018079	22.00
R_{12}	DB1	4	0.017189	20.92
R_8	BR1	3	0.010418	12.68
R_{11}	CS1	4	0.007648	9.31
R_9	GW1	3	0.007427	9.04
R_1	EG1	1	0.007424	9.04
R_4	SW1	2	0.006458	7.86
R_{10}	LB1	4	0.003550	4.32
R_5	PR1	2	0.002404	2.93
R_2	ES1	1	0.000792	0.96
R_6	FW1	2	0.000775	0.94
R_3	CN1	1	0.000000	0.00

4. Cost-Minimization Model, Frontier and Robustness

4.1 Constrained allocation problem

The decision vector is $r = (r_1, \dots, r_{12})$ with $R_{i0} \le r_i \le R_{\max}$ and $R_{\max} = 0.999$. Raising component i above its baseline costs

$$C_i(r_i) = a_i\left[\exp\left(b_i\,(r_i - R_{i0})\right) - 1\right], \tag{2}$$

with the coefficients of Table 4, a form that expresses diminishing returns [2, 8, 11]. The allocation problem for a target R_{min} is

$$\text{minimize } C(r) = \sum_{i=1}^{12} C_i\,(r_i) \quad \text{subject to} \quad R_s(r) \ge R_{\min},\ R_{i0} \le r_i \le R_{\max}. \tag{3}$$

For the metaheuristics the constraint is handled by a static penalty. With $U(r) = 1 - R_s(r)$, $U_{max} = 1 - R_{min}$ and $C_{max} = 29.3049$ (the cost of raising every component to R_{max}), the fitness minimized by both algorithms is

$$F(r) = C(r)/C_{\max} + \lambda \cdot \max(0,\ (U(r) - U_{\max})/U_{\max}), \quad \lambda = 20. \tag{4}$$

The violation is measured relative to the allowed unreliability, so that a design which misses the target by one per cent of U_{max} is penalized by 0.2, which exceeds any cost saving available near the optimum. Reliability is evaluated through the exact polynomial of Appendix A at every call.

Table 4. Cost-model coefficients and the cost of raising each component to R_{max} = 0.999.

Symbol	Node	Baseline R_{i0}	a_i	b_i	$C_i(R_{max})$
R_1	EG1	0.94	1.2	18	2.271
R_2	ES1	0.93	1	18	2.463
R_3	CN1	0.96	1.5	16	1.300
R_4	SW1	0.92	0.8	20	3.084
R_5	PR1	0.93	1.1	18	2.709
R_6	FW1	0.95	1.4	15	1.520
R_7	ST1	0.91	0.9	20	4.437
R_8	BR1	0.92	1	18	3.145
R_9	GW1	0.94	1.3	16	2.041
R_{10}	LB1	0.93	1	18	2.463
R_{11}	CS1	0.93	1	18	2.463
R_{12}	DB1	0.95	1.3	15	1.411

4.2 Robustness and the frontier

The cost-reliability frontier, or the lowest cost as a function of the target, can be found by solving problem (3) for a variety of objectives. A design r is regarded as a nominal specification for robustness, and the components that are actually constructed are presumed to be less dependable than those stated. Each component's failure probability is inflated by an independent factor κ_i drawn uniformly from [1, 1.5], so that $r'_i = 1 - \kappa_i(1 - r_i)$. This multiplicative form means that poorer components suffer larger absolute losses than the highly reliable ones, which is more realistic than a uniform shortfall. After evaluating the system dependability Rs(r') for 20,000 random draws, we note the chance that it still satisfies the 0.9997 criteria along with its mean and fifth percentile. A margin is provided by a design optimized for a target above 0.9997; the question is how much margin is required and how much it costs.

5. The Two Optimization Algorithms

Both algorithms work on the continuous box $R_{i0} \le r_i \le R_{max}$, start from uniform random populations and replace a candidate only when the new one has a lower fitness. They spend fitness evaluations at different rates. A COA iteration costs 2.5n evaluations (one per agent in the hunting phase, one extra for each agent of the second half whose iguana falls to a random position, and one per agent in the escape phase), whereas an AOA iteration costs n. To keep the comparison fair the budget was therefore fixed in evaluations: COA runs 120 iterations and AOA 300 iterations, so that both use 12,040 evaluations including the initial population. Control parameters follow the original papers (Table 5).

5.1 Coati Optimization Algorithm (COA)

COA models two natural behaviours of coatis [12]. In the first phase (hunting and attacking iguanas) the population is split in two halves. The first half climbs toward the best solution, regarded as the iguana,

$$x_i^{new} = x_i + r\,(\text{Iguana} - I\,x_i), \tag{5}$$

where r is a uniform random vector and I is randomly 1 or 2. For the second half the iguana has fallen to a random position Iguana_G in the box, and the agent moves toward it if it is better than the agent and away from it otherwise,

$$x_i^{new} = x_i + r\,(\,\text{Iguana}_G - I\,x_i\,) \text{ if } F(\text{Iguana}_G) < F(x_i); \quad \text{otherwise } x_i + r\,(\,x_i - \text{Iguana}_G\,). \tag{6}$$

In the second phase (escaping from predators) each agent performs a local move within bounds that shrink with the iteration counter t,

$$x_i^{new} = x_i + (\,1 - 2r\,)\,(\,lb / t + r\,(\,ub / t - lb / t\,)\,). \tag{7}$$

In every phase the new position is clipped to the box and accepted only if it improves the fitness. The procedure is summarized in Figure 2.

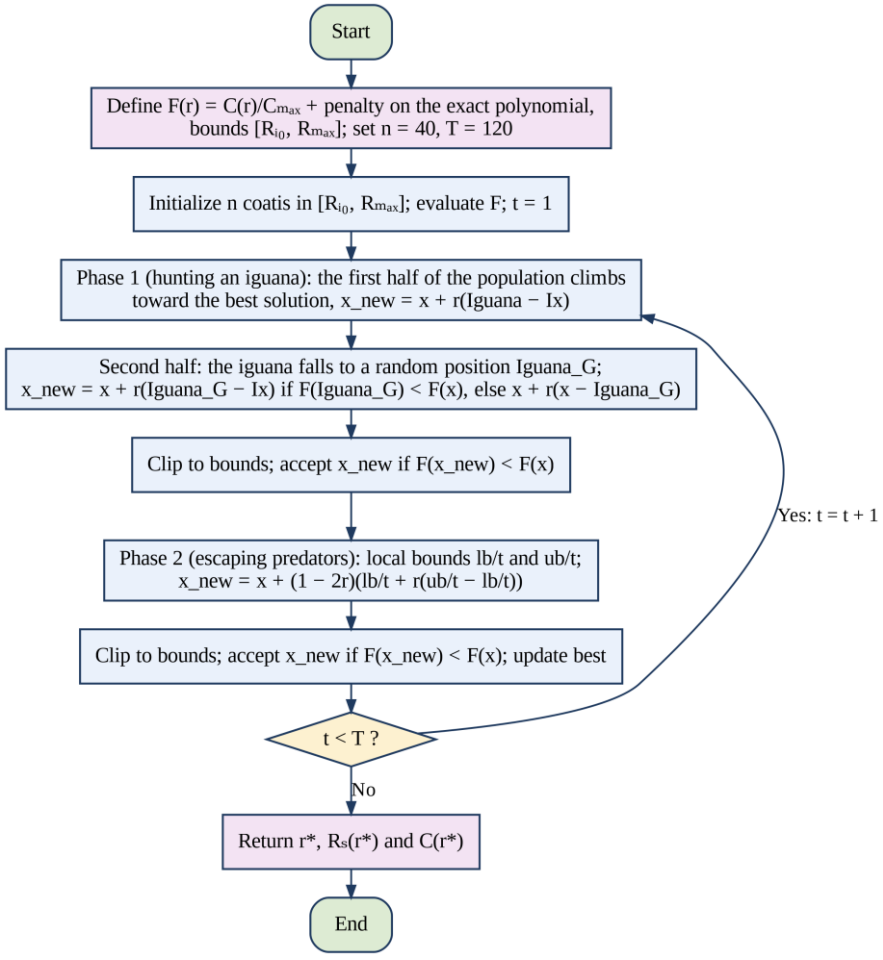


Figure 2. Flowchart of the Coati Optimization Algorithm (COA) as implemented in this study.

5.2 Arithmetic Optimization Algorithm (AOA)

AOA builds each candidate from the best solution found so far by means of the four arithmetic operators [13]. Two coefficients control the search: the accelerated function MOA = Min + t (Max − Min) / T, with Min = 0.2 and Max = 1, and the probability MOP = $1 - t^{1/\alpha} / T^{1/\alpha}$, with $\alpha = 5$. For each coordinate a random number r_1 is compared with MOA. If $r_1 >$ MOA the coordinate is explored by division or multiplication (chosen by a random number r_2),

$$x^{new} = x_{best} / (MOP + \varepsilon) \cdot S \quad (r_2 > 0.5), \quad x^{new} = x_{best} \cdot MOP \cdot S \quad (\text{otherwise}), \tag{8}$$

and otherwise it is exploited by subtraction or addition (chosen by r_3),

$$x^{new} = x_{best} - MOP \cdot S \quad (r_3 > 0.5), \quad x^{new} = x_{best} + MOP \cdot S \quad (\text{otherwise}), \tag{9}$$

where $S = (ub - lb)\,\mu + lb$ with $\mu = 0.499$ is a coordinate-wise scale. Because MOA grows and MOP shrinks over the run, exploration dominates early and exploitation later, and the exploitation step shrinks to zero at the end. New positions are clipped to the box and accepted greedily. The procedure is summarized in Figure 3.

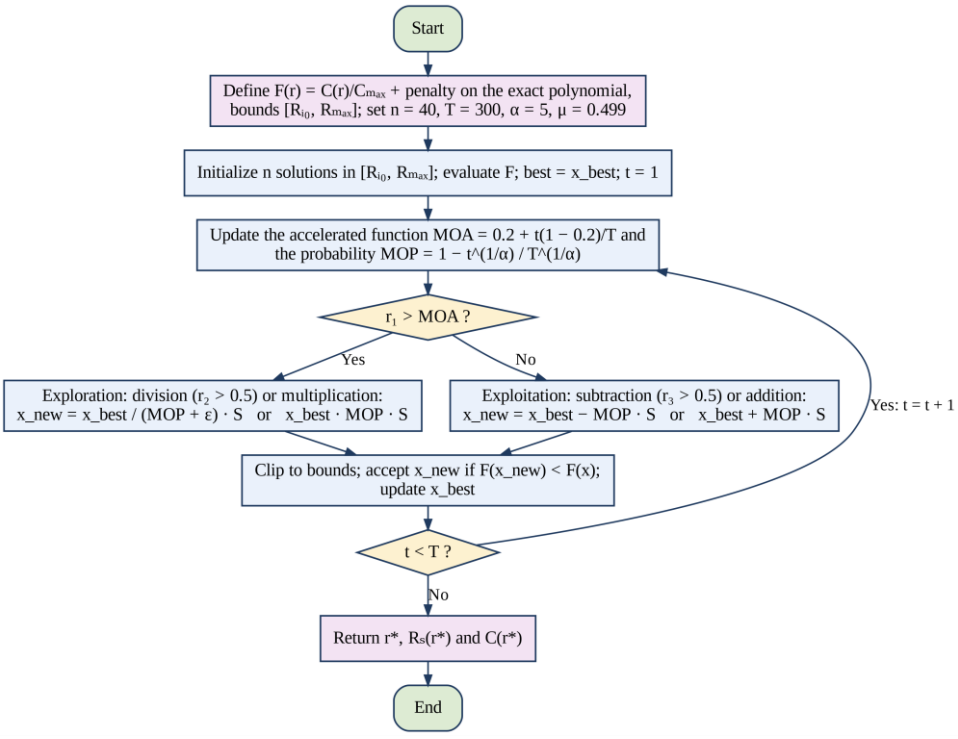


Figure 3. Flowchart of the Arithmetic Optimization Algorithm (AOA) as implemented in this study.

Table 5. Parameter settings of the two algorithms (matched evaluation budget).

COA parameter	Value	AOA parameter	Value
Population size n	40	Population size n	40
Iterations T	120	Iterations T	300
I	1 or 2 (random)	Min, Max (MOA)	0.2, 1
Phase 2 bounds	lb/t, ub/t	α, μ	5, 0.499
Population split	Two halves	Operators	÷, ×, −, +
Replacement	Greedy	Replacement	Greedy
Evaluations per run	12,040	Evaluations per run	12,040

6. Materials and Methods

6.1 Materials

The study used graph-based route enumeration and Möbius inversion for the exact reliability polynomial, the cost model of Section 4, the COA and AOA algorithms of Section 5, and a Python 3 environment (NumPy, SciPy, Matplotlib) for the polynomial, the optimizers, the constrained reference solver, the statistics, the Monte Carlo experiment and the figures. All runs used a single CPU core.

6.2 Experimental design

Each algorithm was run 30 times with independent seeds for each of five reliability targets, R_{min} = 0.9990, 0.9995, 0.9997, 0.9999 and 0.99995, with population 40 and a budget of about 12,000 fitness evaluations. The target 0.9997 serves as the main case. Two sensitivity studies at this target changed the evaluation budget to about 4,000 and 24,000, and the population size to 20 and 80 at the main budget, each again with 30 runs per algorithm and with iteration counts adjusted so that both algorithms always receive the same number of evaluations. The benchmark for each target is the best of 40 sequential least-squares programming (SLSQP) solutions of problem (3) from random starting points; because the constrained problem is smooth, it provides a reliable reference, and the cost gap to it is reported as a percentage. A run is counted as feasible if $R_s \geq R_{min} - 10^{-9}$, and as near-optimal if its fitness lies within 0.5% of the reference fitness. Differences between algorithms were tested with the two-sided Wilcoxon rank-sum (Mann–Whitney U) test [26, 27] on the final fitness of the 30 runs, at the 5% level, with the effect size $r = |z|/\sqrt{N}$.

7. Results

7.1 Reference optimum and the cost–reliability frontier

The reference optimum for each objective is provided in Table 6. According to their Birnbaum relevance, only R7 and R12 are updated in each instance. The storage node R7 absorbs the extra dependability, rising from 0.9591 at 0.9995 to 0.9986 at 0.99995; the database node R12 is raised first and reaches the upper bound 0.999 once the target is 0.9995 or higher. The cost grows from 1.399 to 5.809 over this range, and the marginal price of reliability rises at the high end: halving the allowed unreliability from 0.9999 to 0.99995 costs 0.446 units, about 8% more. Figure 4 shows the frontier.

Table 6. Reference (SLSQP) cost-minimal allocation for each reliability target. All other components remain at baseline.

R_{min}	Minimal cost	r_7*	r_{12}*	Achieved R_s	Cost / C_{max}
0.9990	1.399	0.9388	0.9786	0.999000	0.0477
0.9995	2.912	0.9591	0.9990	0.999500	0.0994
0.9997	3.924	0.9766	0.9990	0.999700	0.1339
0.9999	5.363	0.9942	0.9990	0.999900	0.1830
0.99995	5.809	0.9986	0.9990	0.999950	0.1982

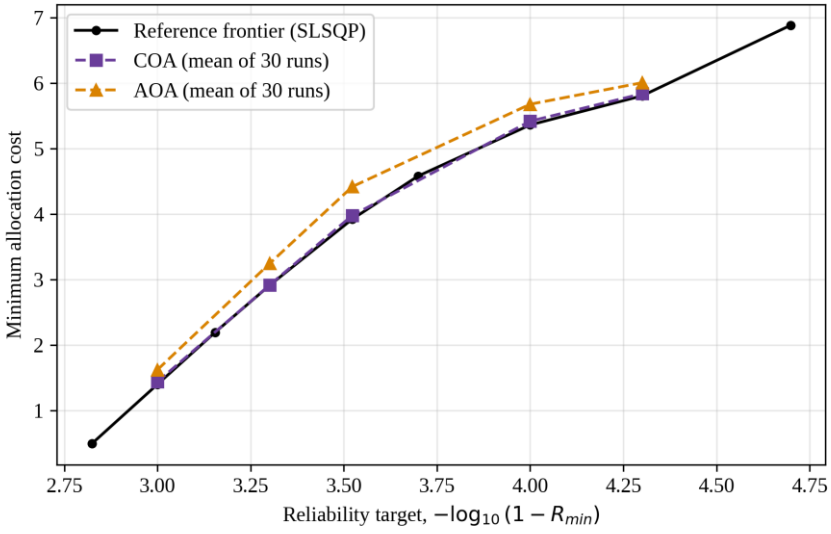


Figure 4. Cost–reliability frontier: reference minimum cost and the mean final cost of COA and AOA over 30 runs.

7.2 Results of the Coati Optimization Algorithm

Table 7 reports the COA allocation at the main target. Its best run raised R_7 to 0.9767 and R_{12} to 0.9990 and left the other ten components at baseline within 10^{-3}, at a cost of 3.924 against the reference 3.924 (gap 0.0%). Over the 30 runs the mean allocated reliabilities follow the same pattern, and the standard deviations are small apart from a modest spread in R_7 and R_{12}.

Table 7. Baseline, reference and COA allocation (best feasible run of 30; cost 3.924, R_s = 0.999700) at R_{min} = 0.9997, with the mean and standard deviation of each allocated reliability over 30 runs.

Symbol	Node	Baseline R_{i0}	Reference	COA r_i*	Δr	C_i	Mean (30)	s.d. (30)
R_1	EG1	0.94000	0.94000	0.94000	0.00000	0.000	0.9401	0.0002
R_2	ES1	0.93000	0.93000	0.93000	0.00000	0.000	0.9300	0.0001
R_3	CN1	0.96000	0.96000	0.96000	0.00000	0.000	0.9601	0.0002
R_4	SW1	0.92000	0.92000	0.92000	0.00000	0.000	0.9206	0.0026
R_5	PR1	0.93000	0.93000	0.93000	0.00000	0.000	0.9301	0.0002
R_6	FW1	0.95000	0.95000	0.95000	0.00000	0.000	0.9501	0.0004
R_7	ST1	0.91000	0.97665	0.97666	0.06666	2.514	0.9782	0.0047
R_8	BR1	0.92000	0.92000	0.92000	0.00000	0.000	0.9202	0.0005
R_9	GW1	0.94000	0.94000	0.94000	0.00000	0.000	0.9408	0.0028
R_{10}	LB1	0.93000	0.93000	0.93000	0.00000	0.000	0.9300	0.0002
R_{11}	CS1	0.93000	0.93000	0.93000	0.00000	0.000	0.9301	0.0003
R_{12}	DB1	0.95000	0.99900	0.99897	0.04897	1.410	0.9956	0.0095
Total						**3.924**		

7.3 Results of the Arithmetic Optimization Algorithm

The relevant AOA findings are shown in Table 8. With R7 at 0.9693 and R12 at 0.9990, the best AOA run is doable and structurally similar to the reference, but it also increased R9 to 0.9689, and its cost of 4.270 is 8.8% more than the reference. The averages reveal the source of the excess. AOA incorrectly balances the allocation between the two active components, causing components that should remain at baseline to hardly move. For example, R12's mean value is only 0.9585 instead of 0.999, and R7's mean is pushed to 0.9917 against the reference 0.9766 to make up the gap. This balance is expensive since the upgrading of R7 is the steeper of the two (b7 = 20 versus b12 = 15).

Table 8. Baseline, reference and AOA allocation (best feasible run of 30; cost 4.270, R_s = 0.999705) at R_{min} = 0.9997, with the mean and standard deviation of each allocated reliability over 30 runs.

Symbol	Node	Baseline R_{i0}	Reference	AOA r_i*	Δr	C_i	Mean (30)	s.d. (30)
R_1	EG1	0.94000	0.94000	0.94065	0.00065	0.014	0.9412	0.0044
R_2	ES1	0.93000	0.93000	0.93000	0.00000	0.000	0.9305	0.0008
R_3	CN1	0.96000	0.96000	0.96000	0.00000	0.000	0.9602	0.0005
R_4	SW1	0.92000	0.92000	0.92064	0.00064	0.010	0.9219	0.0056
R_5	PR1	0.93000	0.93000	0.93000	0.00000	0.000	0.9303	0.0007
R_6	FW1	0.95000	0.95000	0.95065	0.00065	0.014	0.9503	0.0005
R_7	ST1	0.91000	0.97665	0.96928	0.05928	2.046	0.9917	0.0171
R_8	BR1	0.92000	0.92000	0.92064	0.00064	0.012	0.9209	0.0025
R_9	GW1	0.94000	0.94000	0.96889	0.02889	0.764	0.9446	0.0125
R_{10}	LB1	0.93000	0.93000	0.93000	0.00000	0.000	0.9308	0.0009
R_{11}	CS1	0.93000	0.93000	0.93000	0.00000	0.000	0.9315	0.0034
R_{12}	DB1	0.95000	0.99900	0.99900	0.04900	1.411	0.9585	0.0170
Total						**4.270**		

7.4 Comparison at the main target

Table 9 compares the 30 runs of each algorithm. COA is better on every cost measure: its mean cost is 3.980 against 4.420 for AOA, a mean gap to the reference of 1.4% against 12.6%. All 30 runs of both algorithms were feasible, so the difference is one of cost, not of constraint violation. 19 COA runs came within 0.5% of the reference fitness, whereas no AOA run did. The rank-sum test confirms the difference ($U = 5$, $p < 0.001$, $r = 0.85$). AOA is much faster per run, because it makes one vectorized call per iteration, and Figure 5 shows that its mean curve flattens early. COA needed about 85 of its 120 iterations to come within 0.5% of the reference fitness, in the 19 runs that did so.

Table 9. Summary statistics of 30 independent runs at R_{min} = 0.9997 (population 40, about 12,000 evaluations).

Metric	COA	AOA
Reference optimum cost (SLSQP)	3.9240	3.9240
Best feasible cost	3.9243	4.2704
Mean cost (30 runs)	3.9799	4.4195
Median cost	3.9338	4.4070
Worst cost	4.3915	4.8337
Standard deviation of cost	0.1059	0.0905
Cost gap, best run (%)	0.01	8.83
Cost gap, mean (%)	1.43	12.63
Feasible runs (of 30)	30	30
Mean fitness F (s.d.)	0.1358 (0.0036)	0.1508 (0.0031)
Runs within 0.5% of reference fitness	19	0
Iterations T	120	300
Mean CPU time per run (s)	0.31	0.06
Fitness evaluations per run	12,040	12,040
Wilcoxon rank-sum test (two-sided)	U = 5	p < 0.001

7.5 Convergence behaviour

Figure 5 plots the best-so-far fitness against the number of fitness evaluations, which is the fair horizontal axis for two methods with different iteration costs. COA decreases steadily and is still improving slowly when the budget ends, whereas AOA drops quickly during the first thousand evaluations and then stays on a plateau well above the reference level. The final-cost distributions in Figure 6 show the same difference, and Figure 7 compares the allocations of the reference, COA and AOA best runs.

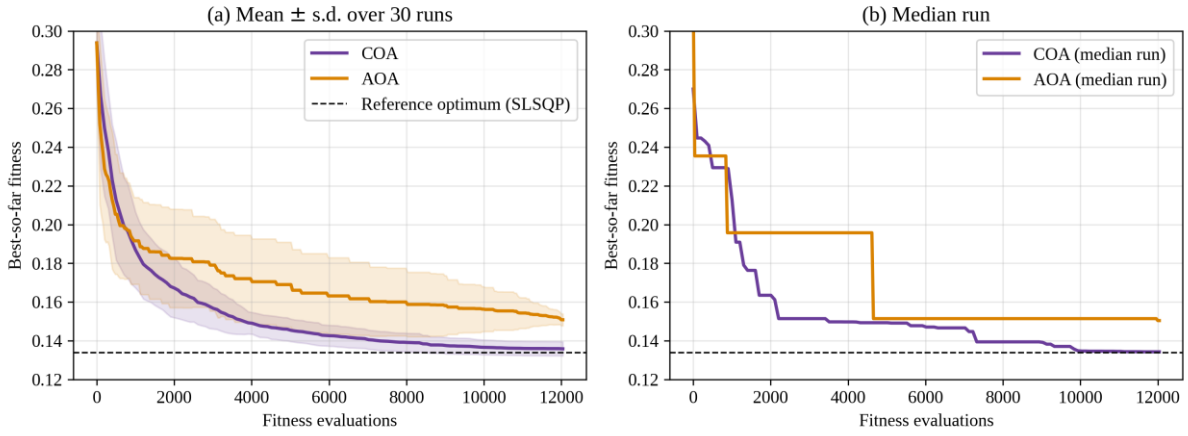


Figure 5. Best-so-far fitness against fitness evaluations at R_{min} = 0.9997: (a) mean ± standard deviation over 30 runs; (b) median run of each algorithm.

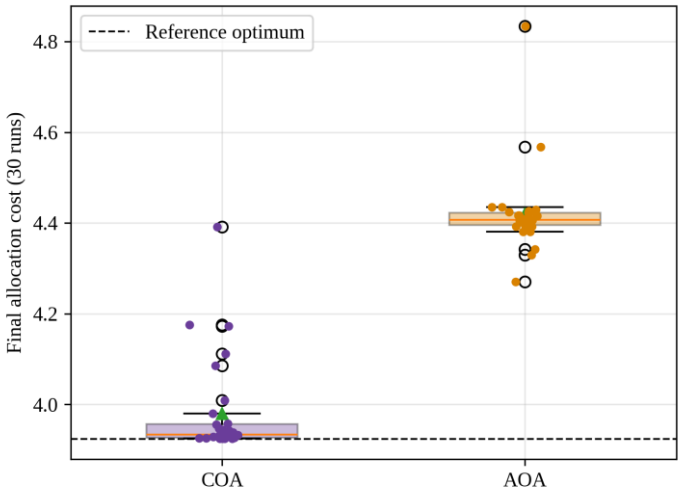


Figure 6. Final allocation cost over 30 runs at R_{min} = 0.9997 (diamonds are means).

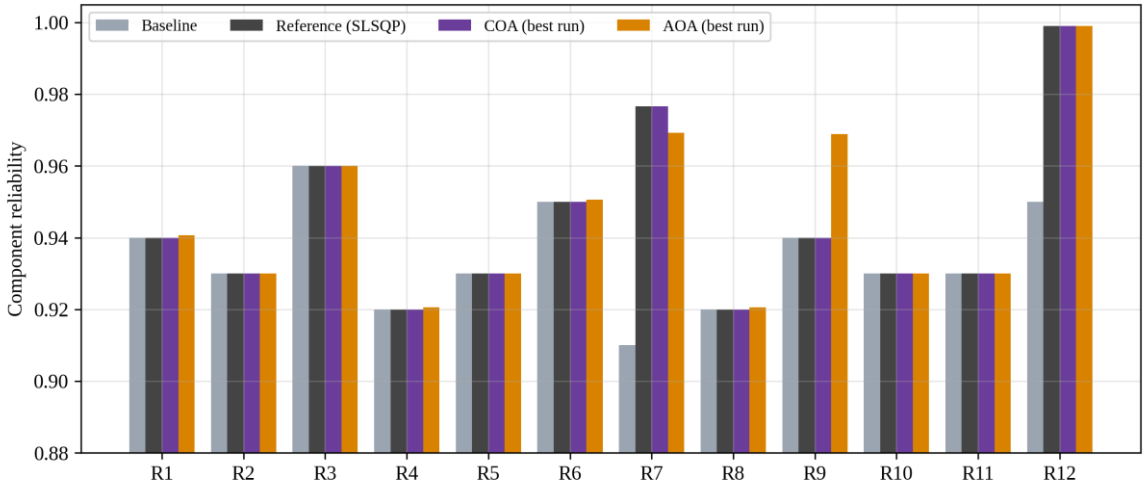


Figure 7. Component reliabilities of the baseline, the reference optimum and the best runs of COA and AOA at R_{min} = 0.9997.

7.6 Results across reliability targets

Table 10 repeats the comparison for five targets. COA is better than AOA at all of them ($p < 0.001$ in each case). Its mean cost gap to the reference stays between 0.16% and 2.78%, and is smallest at 0.9995 where 28 of its 30 runs are near-optimal. The AOA gap lies between 3.44% and 15.96% and is largest at the loosest target 0.9990, where one AOA run also ended below the reliability requirement, so that only 29 of 30 AOA runs were feasible.

Table 10. Comparison of COA and AOA for five reliability targets (30 runs each, n = 40, about 12,000 evaluations). Gaps are relative to the SLSQP reference cost.

R_{min}	Reference cost	COA mean cost (gap %)	AOA mean cost (gap %)	Best-run gap % C / A	Near-opt. runs C / A	p
0.9990	1.399	1.438 (2.78)	1.622 (15.96)	0.11 / 3.60	3 / 0	< 0.001
0.9995	2.912	2.917 (0.16)	3.245 (11.44)	0.01 / 1.34	28 / 0	< 0.001
0.9997	3.924	3.980 (1.43)	4.420 (12.63)	0.01 / 8.83	19 / 0	< 0.001
0.9999	5.363	5.420 (1.06)	5.679 (5.90)	0.09 / 2.85	5 / 0	< 0.001
0.99995	5.809	5.840 (0.54)	6.008 (3.44)	0.17 / 0.43	9 / 2	< 0.001

7.7 Evaluation budget and population size

The response to the evaluation budget and population size for Rmin = 0.9997 is displayed in Table 11. A higher budget obviously helps COA: 27 out of 30 runs become almost optimal, and its mean cost drops from 4.367 at roughly 4,000 evaluations to 3.980 at 12,000 and 3.935 at 24,000. As its early plateau in Figure 5 indicates, AOA never approaches the near-optimal range and only slightly improves (4.635, 4.420, and 4.411). Because it permits more iterations, a smaller population (n = 20) is better for COA at a fixed budget, whereas a bigger one (n = 80) is somewhat worse for both. The conclusion is independent of the specific budget or population size selected because COA is significantly better in all configurations ($p < 0.001$).

Table 11. Sensitivity to evaluation budget and population size at $R_{min} = 0.9997$ (30 runs each; both algorithms receive the same number of evaluations).

Budget (evaluations)	COA T	AOA T	COA mean cost (s.d.)	AOA mean cost (s.d.)	p
4,000	40	99	4.367 (0.164)	4.635 (0.272)	< 0.001
12,000	120	300	3.980 (0.106)	4.420 (0.091)	< 0.001
24,000	240	599	3.935 (0.030)	4.411 (0.029)	< 0.001
12,000, n = 20	240	601	3.950 (0.076)	4.412 (0.113)	< 0.001
12,000, n = 80	60	150	4.036 (0.137)	4.453 (0.161)	< 0.001

7.8 Robustness and the price of robustness

What a cost-optimal design achieves when components are subpar is the question posed in the last experiment. The mean system reliability of the design optimized precisely for 0.9997 falls to 0.999382 when each failure probability may increase by up to 50%, and none of the 20,000 draws satisfy the condition of 0.9997 (Table 12). The structural explanation is that any degradation drives a cost-minimal design below the constraint boundary. The required margin is created by designing for a higher aim, although it is a significant margin. Only 1.5% of draws for a design optimized for 0.9998 still satisfy the criteria, compared to 37.8% with a design optimized for 0.99985. At the design target

of 0.9999, at an additional cost of 36.7%, the probability hits 99.4%, and at 0.99995, at an additional cost of 48.0%, it achieves 100%. The jump between 0.99985 and 0.9999 shows that the probability rises steeply once the design target passes a threshold, a property that a designer can exploit by choosing the smallest target beyond it.

Table 12. Price of robustness: reference-optimal designs for several design targets evaluated against the requirement $R_s \geq 0.9997$ when each component failure probability is multiplied by a random factor in [1, 1.5] (20,000 draws).

Design target	Cost	Extra cost vs 0.9997	P($R_s \geq 0.9997$)	Mean R_s	5th percentile
0.9997	3.924	0.0%	0.0%	0.999382	0.999223
0.99975	4.238	8.0%	0.0%	0.999483	0.999351
0.9998	4.580	16.7%	1.5%	0.999583	0.999478
0.99985	4.954	26.3%	37.8%	0.999683	0.999604
0.9999	5.363	36.7%	99.4%	0.999784	0.999729
0.99995	5.809	48.0%	100.0%	0.999884	0.999849

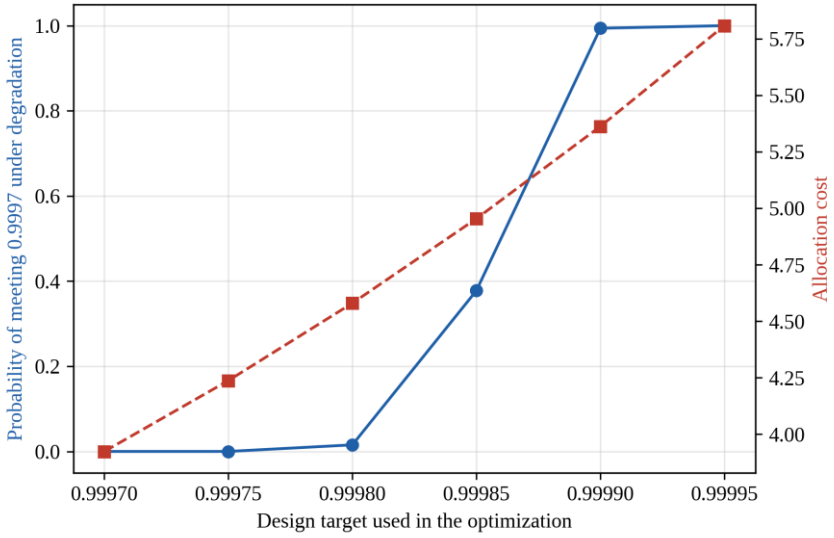


Figure 8. Probability of meeting 0.9997 under inflated failure probabilities (left axis) and allocation cost (right axis) against the design target used in the optimization.

Table 13 checks whether the metaheuristic designs behave like the reference under the same degradation. For the targets 0.9997, 0.9999 and 0.99995 the best-run designs of COA are practically identical to the reference in cost and in robustness. The AOA designs are somewhat costlier (4.270 against 3.924 at 0.9997, for instance) and consequently slightly more robust where the requirement is borderline (99.9% against 99.5% at 0.9999). The additional robustness of AOA is not a free benefit, since it is bought by the same unnecessary extra spending that caused its cost gap, and it is far more cheaply obtained by raising the design target.

Table 13. Robustness of the best-run designs of COA and AOA, compared with the reference design, for three design targets (same degradation model as Table 12).

Design target	Algorithm	Cost	$P(R_s \geq 0.9997)$	Mean R_s	5th percentile
0.9997	Reference	3.924	0.0%	0.999382	0.999223
	COA	3.924	0.0%	0.999382	0.999223
	AOA	4.270	0.0%	0.999393	0.999234
0.9999	Reference	5.363	99.4%	0.999784	0.999729
	COA	5.368	99.5%	0.999785	0.999730
	AOA	5.516	99.9%	0.999792	0.999741
0.99995	Reference	5.809	100.0%	0.999884	0.999849
	COA	5.819	100.0%	0.999884	0.999848
	AOA	5.833	100.0%	0.999886	0.999850

8. Discussion

The issue is the subject of the first observation. The structure of the optimum is simple to understand when allocation is presented as limited cost minimization: the database node gets upgraded and saturates first, followed by the storage node, while all other components remain at baseline. Because R12 is both significant and affordable to improve, this ordering corresponds to the precise Birnbaum importance of Table 3 and the cost coefficients of Table 4. A single weighted-objective run cannot show that each new "nine" of reliability costs more than the previous one since the border in Figure 4 is convex. The algorithms are the subject of the second observation. The manner COA is constructed may account for its good performance. The population gradually focuses on the two active variables with greedy acceptance as its second phase makes small local moves within shrinking bounds, first exploring a good region and then refining it. Its first phase draws every agent toward the best solution or toward a random point. Because each candidate is created from the single best solution by adding or removing the same scaled quantity MOP·S in each coordinate, AOA behaves differently. Therefore, a candidate that improves one coordinate is likely to upset multiple others, and such candidates are rejected under greedy acceptance, which would account for the early plateau. The results relate to the default parameters of both approaches; adjusted or hybrid forms of AOA may perform better. This reading is an interpretation of Figure 5 and Table 11 that we did not test directly. The third point is useful. The robustness study demonstrates that when component reliabilities are uncertain, the nominal optimum is not a safe design and that a significant margin is required under the failure-rate inflation assumed here: the probability of meeting the requirement remains below 40% up to a design target of 0.99985 and is nearly certain at 0.9999. Therefore, rather than just setting the design target to the need, designers should approach it as a choice variable of its own, selected from the frontier and the robustness curve. There are a few restrictions. The cost coefficients are illustrative values, link failures and common-cause failures are not modeled, and component failures are expected to occur independently. Real degradation may be connected with the degradation model, which is a straightforward multiplicative inflation of failure probabilities. Conclusions concerning the algorithms should not be applied to different topologies without additional testing, as the comparison only looks at one network and two algorithms with default parameters.

9. Conclusions

This paper formulated reliability allocation on the QSDR-ECRN edge–cloud network as cost minimization under an explicit reliability constraint, evaluated through its exact 69-term reliability polynomial. A constrained reference solution gave the cost–reliability frontier: only the database and storage nodes need to be upgraded, and the cost rises from 1.399 at a target of 0.9990 to 5.809 at 0.99995. Under matched evaluation budgets, the Coati Optimization Algorithm stayed within 0.2% to 2.8% of the reference cost on average, while the Arithmetic Optimization Algorithm settled 3.4% to 16.0% above it and did not improve appreciably with larger budgets; the difference was significant for all five targets, three budgets and three population sizes. A Monte Carlo study showed that a design optimized exactly for a target fails to meet it as soon as component failure probabilities rise, and that a design target near 0.9999 is needed for the 0.9997 requirement to be met in almost every case, at 36.7% additional cost. Future work may include multi-objective optimizers that return the whole frontier in one run, correlated and common-cause failures, time-varying component reliabilities, larger topologies, and comparisons with the algorithms used in earlier studies of this network [20, 21].

Appendix A. Exact Reliability Polynomial of QSDR-ECRN

The exact system reliability under independent component failures is the sum of the 69 signed monomials in Table A1 (all coefficients are +1 or −1; monomials are ordered by decreasing degree). At the baseline reliabilities it equals [0.998100], and it is used unmodified inside the fitness function of Section .

Table A1. The 69 signed monomials of the exact reliability polynomial $R_s(R_1,\ldots,R_{12})$.

No.	Term	No.	Term	No.	Term
1	$+ R_1 \cdot R_2 \cdot R_5 \cdot R_6 \cdot R_7 \cdot R_8 \cdot R_9 \cdot R_{10} \cdot R_{11} \cdot R_{12}$	24	$- R_1 \cdot R_2 \cdot R_6 \cdot R_7 \cdot R_9 \cdot R_{11} \cdot R_{12}$	47	$- R_1 \cdot R_4 \cdot R_5 \cdot R_9 \cdot R_{12}$
2	$- R_1 \cdot R_2 \cdot R_4 \cdot R_5 \cdot R_6 \cdot R_7 \cdot R_8 \cdot R_9 \cdot R_{12}$	25	$- R_1 \cdot R_4 \cdot R_5 \cdot R_7 \cdot R_8 \cdot R_{11} \cdot R_{12}$	48	$- R_1 \cdot R_5 \cdot R_7 \cdot R_8 \cdot R_{11}$
3	$- R_1 \cdot R_2 \cdot R_5 \cdot R_6 \cdot R_7 \cdot R_8 \cdot R_9 \cdot R_{10} \cdot R_{12}$	26	$+ R_1 \cdot R_4 \cdot R_5 \cdot R_7 \cdot R_9 \cdot R_{10} \cdot R_{12}$	49	$- R_1 \cdot R_5 \cdot R_7 \cdot R_8 \cdot R_{12}$
4	$- R_1 \cdot R_2 \cdot R_5 \cdot R_6 \cdot R_7 \cdot R_8 \cdot R_9 \cdot R_{11} \cdot R_{12}$	27	$+ R_1 \cdot R_4 \cdot R_5 \cdot R_7 \cdot R_9 \cdot R_{11} \cdot R_{12}$	50	$- R_1 \cdot R_5 \cdot R_8 \cdot R_9 \cdot R_{12}$
5	$- R_1 \cdot R_2 \cdot R_5 \cdot R_6 \cdot R_7 \cdot R_9 \cdot R_{10} \cdot R_{11} \cdot R_{12}$	28	$+ R_1 \cdot R_5 \cdot R_7 \cdot R_8 \cdot R_9 \cdot R_{10} \cdot R_{12}$	51	$- R_1 \cdot R_5 \cdot R_8 \cdot R_{11} \cdot R_{12}$
6	$- R_1 \cdot R_2 \cdot R_6 \cdot R_7 \cdot R_8 \cdot R_9 \cdot R_{10} \cdot R_{11} \cdot R_{12}$	29	$+ R_1 \cdot R_5 \cdot R_7 \cdot R_8 \cdot R_9 \cdot R_{11} \cdot R_{12}$	52	$- R_4 \cdot R_5 \cdot R_8 \cdot R_9 \cdot R_{12}$
7	$+ R_1 \cdot R_4 \cdot R_5 \cdot R_7 \cdot R_8 \cdot R_9 \cdot R_{10} \cdot R_{11} \cdot R_{12}$	30	$+ R_1 \cdot R_5 \cdot R_7 \cdot R_9 \cdot R_{10} \cdot R_{11} \cdot R_{12}$	53	$+ R_4 \cdot R_7 \cdot R_8 \cdot R_{11} \cdot R_{12}$
8	$+ R_1 \cdot R_2 \cdot R_4 \cdot R_5 \cdot R_6 \cdot R_8 \cdot R_9 \cdot R_{12}$	31	$+ R_4 \cdot R_5 \cdot R_7 \cdot R_8 \cdot R_9 \cdot R_{10} \cdot R_{12}$	54	$+ R_7 \cdot R_8 \cdot R_{10} \cdot R_{11} \cdot R_{12}$
9	$+ R_1 \cdot R_2 \cdot R_4 \cdot R_6 \cdot R_7 \cdot R_8 \cdot R_9 \cdot R_{12}$	32	$+ R_4 \cdot R_5 \cdot R_7 \cdot R_8 \cdot R_9 \cdot R_{11} \cdot R_{12}$	55	$+ R_1 \cdot R_5 \cdot R_8 \cdot R_{11}$
10	$+ R_1 \cdot R_2 \cdot R_5 \cdot R_6 \cdot R_7 \cdot R_8 \cdot R_9 \cdot R_{12}$	33	$+ R_4 \cdot R_5 \cdot R_7 \cdot R_9 \cdot R_{10} \cdot R_{11} \cdot R_{12}$	56	$+ R_1 \cdot R_5 \cdot R_8 \cdot R_{12}$
11	$+ R_1 \cdot R_2 \cdot R_5 \cdot R_6 \cdot R_7 \cdot R_9 \cdot R_{10} \cdot R_{12}$	34	$- R_1 \cdot R_2 \cdot R_5 \cdot R_6 \cdot R_9 \cdot R_{12}$	57	$+ R_1 \cdot R_5 \cdot R_9 \cdot R_{12}$

No.	Term	No.	Term	No.	Term
12	$+ R_1 \cdot R_2 \cdot R_5 \cdot R_6 \cdot R_7 \cdot R_9 \cdot R_{11} \cdot R_{12}$	35	$+ R_1 \cdot R_4 \cdot R_5 \cdot R_7 \cdot R_8 \cdot R_{11}$	58	$+ R_4 \cdot R_5 \cdot R_9 \cdot R_{12}$
13	$+ R_1 \cdot R_2 \cdot R_6 \cdot R_7 \cdot R_8 \cdot R_9 \cdot R_{10} \cdot R_{12}$	36	$+ R_1 \cdot R_4 \cdot R_5 \cdot R_7 \cdot R_8 \cdot R_{12}$	59	$- R_4 \cdot R_7 \cdot R_8 \cdot R_{11}$
14	$+ R_1 \cdot R_2 \cdot R_6 \cdot R_7 \cdot R_8 \cdot R_9 \cdot R_{11} \cdot R_{12}$	37	$+ R_1 \cdot R_4 \cdot R_5 \cdot R_8 \cdot R_9 \cdot R_{12}$	60	$- R_4 \cdot R_7 \cdot R_8 \cdot R_{12}$
15	$+ R_1 \cdot R_2 \cdot R_6 \cdot R_7 \cdot R_9 \cdot R_{10} \cdot R_{11} \cdot R_{12}$	38	$+ R_1 \cdot R_4 \cdot R_5 \cdot R_8 \cdot R_{11} \cdot R_{12}$	61	$- R_4 \cdot R_8 \cdot R_{11} \cdot R_{12}$
16	$- R_1 \cdot R_4 \cdot R_5 \cdot R_7 \cdot R_8 \cdot R_9 \cdot R_{10} \cdot R_{12}$	39	$+ R_1 \cdot R_5 \cdot R_7 \cdot R_8 \cdot R_{11} \cdot R_{12}$	62	$- R_7 \cdot R_8 \cdot R_{10} \cdot R_{12}$
17	$- R_1 \cdot R_4 \cdot R_5 \cdot R_7 \cdot R_8 \cdot R_9 \cdot R_{11} \cdot R_{12}$	40	$- R_1 \cdot R_5 \cdot R_7 \cdot R_9 \cdot R_{10} \cdot R_{12}$	63	$- R_7 \cdot R_8 \cdot R_{11} \cdot R_{12}$
18	$- R_1 \cdot R_4 \cdot R_5 \cdot R_7 \cdot R_9 \cdot R_{10} \cdot R_{11} \cdot R_{12}$	41	$- R_1 \cdot R_5 \cdot R_7 \cdot R_9 \cdot R_{11} \cdot R_{12}$	64	$+ R_4 \cdot R_8 \cdot R_{11}$
19	$- R_1 \cdot R_5 \cdot R_7 \cdot R_8 \cdot R_9 \cdot R_{10} \cdot R_{11} \cdot R_{12}$	42	$- R_4 \cdot R_5 \cdot R_7 \cdot R_9 \cdot R_{10} \cdot R_{12}$	65	$+ R_4 \cdot R_8 \cdot R_{12}$
20	$- R_4 \cdot R_5 \cdot R_7 \cdot R_8 \cdot R_9 \cdot R_{10} \cdot R_{11} \cdot R_{12}$	43	$- R_4 \cdot R_5 \cdot R_7 \cdot R_9 \cdot R_{11} \cdot R_{12}$	66	$+ R_7 \cdot R_8 \cdot R_{12}$
21	$- R_1 \cdot R_2 \cdot R_4 \cdot R_6 \cdot R_8 \cdot R_9 \cdot R_{12}$	44	$+ R_1 \cdot R_2 \cdot R_6 \cdot R_9 \cdot R_{12}$	67	$- R_7 \cdot R_{10} \cdot R_{11}$
22	$- R_1 \cdot R_2 \cdot R_6 \cdot R_7 \cdot R_8 \cdot R_9 \cdot R_{12}$	45	$- R_1 \cdot R_4 \cdot R_5 \cdot R_8 \cdot R_{11}$	68	$+ R_7 \cdot R_{10}$
23	$- R_1 \cdot R_2 \cdot R_6 \cdot R_7 \cdot R_9 \cdot R_{10} \cdot R_{12}$	46	$- R_1 \cdot R_4 \cdot R_5 \cdot R_8 \cdot R_{12}$	69	$+ R_7 \cdot R_{11}$

References

[1] W. Kuo, V. R. Prasad, An annotated overview of system-reliability optimization, IEEE Transactions on Reliability, 49 (2) (2000), 176–187.
[2] B. S. Dhillon, Design Reliability: Fundamentals and Applications, 1st ed., CRC Press, (1999).
[3] R. B. Patil, S. Kyeong, M. Pecht, R. A. Gujar, S. Mane, Assessment of reliability allocation methods for electronic systems: A systematic and bibliometric analysis, Stats, 8 (1) (2025), 11.
[4] M. I. Khaleel, Enhancing the resilience of error-prone computing environments using a hybrid multi-objective optimization algorithm for edge-centric cloud computing systems, Neural Computing and Applications, 36 (18) (2024), 10733–10760.
[5] H. K. Sulaiman, F. H. Ali, Z. A. H. Hassan, Computational models for allocation and optimization of reliability for ROSS network, AIP Conference Proceedings, 2591 (1) (2023), 050016.
[6] S. A. Abed, H. K. Sulaiman, Z. A. H. Hassan, Reliability allocation and optimization for (ROSS) of a spacecraft by using genetic algorithm, Journal of Physics: Conference Series, 1294 (3) (2019), 032013.
[7] H. K. Sulaiman, et al., A model for allocation and optimization of reliability complex network using particle swarm optimization and genetic algorithms, International Journal of Mathematics and Computer Science, 18 (2) (2023), 225–231.
[9] H. K. Sulaiman, A. M. Hassan, H. K. Hammood, A. H. Alridha, Z. Al-Khafaji, An evaluation of the reliability optimization problem for the electromagnetic system of an airplane using a comparison of PSO and GA, AIP Conference Proceedings, 3264 (1) (2025), 050010.
[9] H. K. Sulaiman, A. H. Alridha, S. A. Abd Al-Zhara, Enhancing neural network structure diagrams: Optimal allocation and reliability calculations using genetic algorithms, AIP Conference Proceedings, 3408 (1) (2026).
[10] H. K. Sulaiman, A. H. Alridha, S. A. Abd Al-Zhara, Mathematical reliability allocation and optimization of neural network structures using Grey Wolf Optimizer and Python-based simulation, Journal of Al-Qadisiyah for Computer Science and Mathematics, in press.

[11] M. Dehghani, Z. Montazeri, E. Trojovská, P. Trojovský, Coati Optimization Algorithm: A new bio-inspired metaheuristic algorithm for solving optimization problems, Knowledge-Based Systems, 259 (2023), 110011.
[12] L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, A. H. Gandomi, The Arithmetic Optimization Algorithm, Computer Methods in Applied Mechanics and Engineering, 376 (2021), 113609.
[13] A. K. Govil, Reliability Engineering, Tata McGraw-Hill, (1983).
[14] D. W. Coit, A. E. Smith, Reliability optimization of series-parallel systems using a genetic algorithm, IEEE Transactions on Reliability, 45 (2) (1996), 254–260.
[15] G. Abdullah, Z. A. H. Hassan, Using particle swarm optimization (PSO) to address reliability allocation of complex network, Journal of Physics: Conference Series, 1664 (1) (2020), 012125.
[16] G. Abdullah, Z. A. H. Hassan, A comparison between genetic algorithm and particle swarm to investigate the reliability allocation of complex network, Journal of Physics: Conference Series, 1818 (1) (2021), 012163.
[17] E. K. Mutar, Estimating the reliability bounds of a communication network by using the sum-of-disjoint-products method, Discrete Mathematics, Algorithms and Applications, (2024), 2450018.
[18] Z. W. Birnbaum, On the importance of different components in a multicomponent system, In: P. R. Krishnaiah (Ed.), Multivariate Analysis II, Academic Press, (1969), 581–592.
[19] S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey wolf optimizer, Advances in Engineering Software, 69 (2014), 46–61.
[20] M. Braik, A. Hammouri, J. Atwan, M. A. Al-Betar, M. A. Awadallah, White Shark Optimizer: A novel bio-inspired meta-heuristic algorithm for global optimization problems, Knowledge-Based Systems, 243 (2022), 108457.
[21] B. Abdollahzadeh, N. Khodadadi, S. Barshandeh, P. Trojovský, F. S. Gharehchopogh, E.-S. M. El-kenawy, L. Abualigah, S. Mirjalili, Puma optimizer (PO): A novel metaheuristic optimization algorithm and its application in machine learning, Cluster Computing, 27 (2024), 5235–5283.
[22] K.-H. Han, J.-H. Kim, Quantum-inspired evolutionary algorithm for a class of combinatorial optimization, IEEE Transactions on Evolutionary Computation, 6 (6) (2002), 580–593.
[23] G. Zhang, Quantum-inspired evolutionary algorithms: a survey and empirical study, Journal of Heuristics, 17 (3) (2011), 303–351.
[24] D. H. Wolpert, W. G. Macready, No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation, 1 (1) (1997), 67–82.
[25] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics Bulletin, 1 (6) (1945), 80–83.
[26] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm and Evolutionary Computation, 1 (1) (2011), 3–18.