Random-Key Optimizers for Quadratic Unconstrained Binary Optimization

Érico Froehlich, Marcelo de Souza

Departamento de Engenharia de Software
Universidade do Estado de Santa Catarina (UDESC)

erico.froehlich@edu.udesc.br, marcelo.desouza@udesc.br

Abstract. *This paper proposes RK-QUBO, a random-key optimization framework for the Quadratic Unconstrained Binary Optimization (QUBO) problem. The approach encodes both the candidate solution and the algorithmic configuration of the decoder in a continuous random-key vector, allowing any continuous optimizer to jointly search over solutions and decoding strategies. Several decoding strategies are proposed, combining solution construction and local search. Experiments on benchmark instances with up to 2500 variables show that the proposed approach consistently outperforms an iterated local search baseline, while a performance gap relative to a state-of-the-art hybrid heuristic indicates room for improvement through more specialized decoder design.*

1. Introduction

Quadratic Unconstrained Binary Optimization (QUBO) is a fundamental problem in combinatorial optimization defined as follows. Given a symmetric matrix $Q \in \mathbb{R}^{n\times n}$, the objective is to find a binary vector $x \in \{0,1\}^n$ that

$$\text{minimize} \quad x^\top Q x = \sum_{i=1}^{n}\sum_{j=1}^{n} Q_{ij} x_i x_j.$$

Despite its simple formulation, QUBO serves as a unified framework for modeling a wide range of combinatorial problems, including maximum cut, graph coloring, and number partitioning [Kochenberger et al. 2014]. Beyond classical combinatorial optimization, QUBO has gained renewed attention due to its equivalence to the Ising model, which makes it a natural candidate for quantum and quantum-inspired computing frameworks [Glover et al. 2019].

QUBO is NP-hard, which implies that exact solvers are generally limited to small or medium-size instances. For large-scale instances, metaheuristic strategies are frequently employed to provide high-quality approximate solutions within reasonable computational budgets. Several approaches have been proposed in the literature [Punnen 2022], ranging from simulated annealing [Alkhamis et al. 1998] and tabu search [Glover et al. 1998] to more sophisticated hybrid methods combining local search with population-based strategies [Glover et al. 2010, Wang et al. 2012].

Among the diverse metaheuristic approaches, random-key optimization (RKO) is a paradigm in which the search is conducted over a continuous space of random keys, and a problem-specific decoder maps each key vector to a feasible solution [Bean 1994]. This separation between the search space and the solution space al-

lows continuous optimizers to be applied to combinatorial problems without modification, and different decoders can be combined with the same optimizer. The framework has been successfully applied to a variety of combinatorial optimization problems [Gonçalves and Resende 2011, Silva et al. 2025, Santos et al. 2026], but its application to QUBO has not been systematically investigated.

We propose a random-key optimization approach for QUBO that combines constructive heuristics and local search procedures as decoders. The continuous search space is explored using two distinct optimizers: the Random-Key Optimizer [Chaves et al. 2025], a solver that implements multiple metaheuristic strategies within the random-key framework, and Nevergrad [Bennet et al. 2021, Rapin and Teytaud 2018], a gradient-free black-box optimization platform. The proposed approach is evaluated against an iterated local search baseline [de Souza and Ritt 2022] and a state-of-the-art hybrid heuristic [de Souza and Ritt 2018]. Experimental results show that the RKO-based approach outperforms the baseline and produces competitive solutions, though a performance gap relative to the state-of-the-art method indicates room for further development, particularly through more specialized decoders or tighter integration of local search.

The remainder of this paper is organized as follows. Section 2 reviews related work on QUBO algorithms and random-key optimization. Section 3 details the random-key optimization paradigm. Section 4 describes RK-QUBO, the proposed framework that implements several decoders and their integration with the two optimizers. Section 5 presents the experimental setup and results. Section 6 concludes the paper and discusses directions for future work.

2. Related Work

Random-key optimization traces back to the work of [Bean 1994], who introduced the random-key representation to decouple the evolutionary operators from the problem structure in genetic algorithms. [Gonçalves and Resende 2011] formalized this idea into the biased random-key genetic algorithm (BRKGA), in which a biased selection mechanism drives the population toward high-quality solutions while maintaining diversity. [Andrade et al. 2021] extended the framework with the BRKGA-MP-IPR, which employs more than two parents during crossover and introduces a path-relinking procedure that operates directly on the random-key space. The versatility of these foundations has motivated a large body of application work. [Londe et al. 2024] present a comprehensive literature review of BRKGA applications across several problem domains.

The RKO paradigm has since been extended beyond the BRKGA. [Schuetz et al. 2022] demonstrated its use for robot trajectory planning by combining BRKGA with dual annealing within the same random-key framework. Subsequent work expanded the portfolio of compatible metaheuristics. [Mangussi et al. 2023] applied RKO variants based on simulated annealing, iterated local search, and variable neighborhood search to the tree of hubs location problem. [Chaves et al. 2024] proposed a random-key GRASP by extending the continuous GRASP algorithm with a problem-specific decoder, obtaining competitive results on five NP-hard problems. [Chaves et al. 2025] introduced a C++ framework that integrates eight metaheuristics operating in parallel over a shared elite solution pool, applied to three NP-hard location

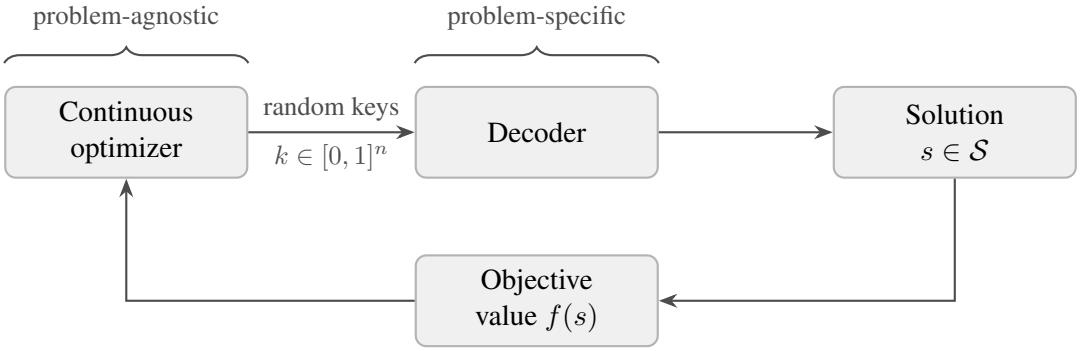


Figure 1. General structure of the Random-Key Optimization framework.

and partitioning problems. More recently, [Chaves et al. 2026] extended the RKO framework to mixed-integer programming, showing that the paradigm can interface with exact solvers in a hybrid fashion.

Applications of random-key methods span a wide range of combinatorial domains. In scheduling, [Fontes and Homayouni 2023] proposed a bi-objective multi-population BRKGA for joint scheduling of quay cranes and vehicles in container terminals, while [Homayouni et al. 2023] addressed the flexible job shop scheduling problem with transportation via a multistart BRKGA. [Silva et al. 2025] combined RKO with reinforcement learning to solve the capacitated multi-period cutting stock problem with setup costs. In routing, [Festa et al. 2023] applied BRKGA with implicit path-relinking to the vehicle routing problem with occasional drivers and time windows. [Brubaker et al. 2025] combined constraint programming with RKO for next-day delivery coverage optimization. [dos Santos Becker and de Souza 2026] applied RKO to drone routing for victim search in disaster scenarios. In cutting and packing, [Gonçalves and Wäscher 2020] developed a BRKGA for the two-dimensional non-guillotine cutting problem with defects, outperforming the best algorithms in the literature on over five thousand benchmark instances. [Santos et al. 2026] addressed the quadratic multiple constraints variable-sized bin packing problem using RKO combined with a linearization strategy. In graph optimization, [Melo et al. 2023] applied BRKGA to the minimum quasi-clique partitioning problem, and [Carvalho et al. 2025] proposed a local-search enhanced BRKGA for the connected Grundy coloring problem.

3. Random-Key Optimization

Random-Key Optimization is a metaheuristic framework in which the search for good solutions to a combinatorial optimization problem is conducted in a continuous space of real-valued vectors, called random keys [Bean 1994]. Let n be the number of decision variables. Each candidate solution is represented as a vector $k = (k_1, k_2, \ldots, k_n) \in [0,1]^n$, where each component k_i is a real number drawn from the unit interval. A problem-specific function $\mathcal{D} : [0,1]^n \to \mathcal{S}$ maps each key vector to a solution $s \in \mathcal{S}$, where $\mathcal{S}$ denotes the solution space of the problem. When the problem admits infeasible solutions, the decoder is typically designed to guarantee feasibility by construction. This function is called the *decoder*.

The general structure of the framework is illustrated in Figure 1. The optimizer operates exclusively in $[0,1]^n$ and has no knowledge of the problem structure. Given a

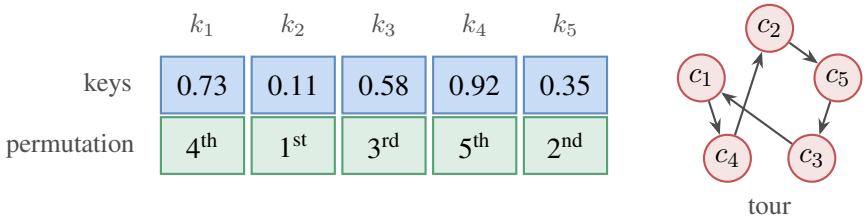


Figure 2. Permutation decoding for the traveling salesperson problem.

key vector k, it queries the decoder to obtain the corresponding solution $s = \mathcal{D}(k)$ and its objective value $f(s)$. The optimizer then uses this information to guide the search toward regions of the key space associated with high-quality solutions. Since the optimizer treats the decoder as a black box, any continuous optimizer can be used within the RKO framework without modification, including population-based metaheuristics, simulated annealing, and gradient-free methods [Chaves et al. 2025].

The decoder is the only problem-specific component of the framework. Its design determines the quality of the solutions that can be obtained and directly affects the efficiency of the search. A well-designed decoder must be (i) deterministic, so that the same key vector always produces the same solution; (ii) computationally efficient, since it is invoked at every function evaluation; and (iii) capable of producing good-quality solutions across a wide range of key vectors. In practice, decoders are often implemented as constructive heuristics, and their output can be further improved by applying a local search procedure before returning the solution to the optimizer [Gonçalves and Resende 2011, Chaves et al. 2025].

3.1. Decoder for TSP: Permutation Decoding

A natural application of random keys is the encoding of permutations, which arise in problems such as the Traveling Salesperson Problem (TSP), sequencing, and scheduling. Given a key vector $k \in [0, 1]^n$, the induced permutation π is obtained by sorting the components of k in ascending order: $\pi(i)$ is the index of the i-th smallest key. The permutation π is then interpreted directly as a solution, for instance as the order in which cities are visited in the TSP.

Figure 2 illustrates this decoding strategy for a TSP instance with five cities $c_1, \ldots, c_5$. The example key vector $(0.73, 0.11, 0.58, 0.92, 0.35)$ induces the permutation $(4, 1, 3, 5, 2)$, which corresponds to the tour $c_2 \to c_5 \to c_3 \to c_1 \to c_4$. Since any key vector in $[0, 1]^n$ induces a valid permutation, feasibility is guaranteed by construction and the optimizer operates freely in the continuous space without constraint handling.

3.2. Decoder for Knapsack: Subset-Selection Decoding

Random keys can also encode subset-selection decisions, which arise in binary optimization problems such as the knapsack problem. In this decoding scheme, item i is included in the solution when $k_i \geq 0.5$, and excluded otherwise. Although other threshold values could be used, a common choice of 0.5 partitions the unit interval symmetrically, giving equal prior probability to inclusion and exclusion. Figure 3 illustrates this decoding for a knapsack instance with five items. The random key vector defines the selection of items 1, 3, and 4, whose keys exceed 0.5, while items 2 and 5 are excluded. This scheme produces

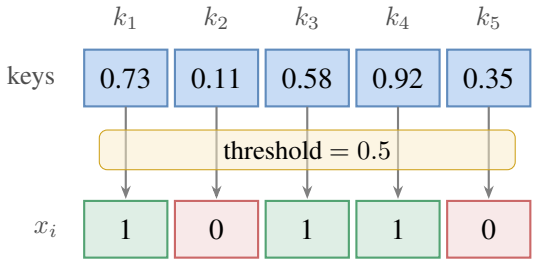


Figure 3. Subset-selection decoding for the binary knapsack problem.

a direct binary assignment without any sorting step, and the optimizer adjusts the key values to shift items in or out of the solution. Note that the selected subset may exceed the knapsack capacity, resulting in an infeasible solution. In such cases, feasibility can be restored through a repair procedure applied after decoding, or the infeasibility can be penalized directly in the objective function returned to the optimizer.

3.3. Decoder for MaxCut: Constructive Decoding with Processing Order

A more general use of random keys is to define the order in which a constructive heuristic processes the elements of the problem. The key vector induces a permutation of the elements, and the heuristic assigns values or makes decisions sequentially according to that permutation, at each step choosing the locally best option given the current partial solution. This scheme combines the generality of the continuous search space with the problem knowledge embedded in the constructive procedure.

Figure 4 illustrates this decoding for the MaxCut problem, where the objective is to partition the vertices of a graph into two sets S_0 and S_1, maximizing the number of edges between the two sets. The key vector determines a permutation of the vertices, which are processed in ascending order of their keys. Each vertex is assigned to the partition that maximizes the number of cut edges with respect to already-assigned vertices. A refinement procedure, such as a single-vertex flip local search, can subsequently be applied to the decoded solution to further improvement before returning the objective value to the optimizer.

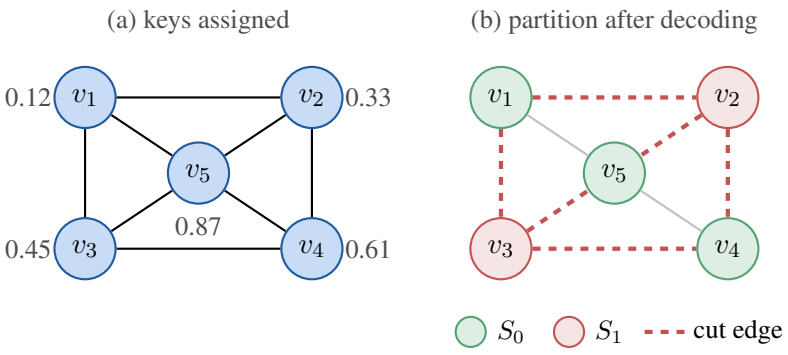


Figure 4. Constructive decoding for the MaxCut problem.

Table 1. Algorithmic parameters encoded in the random-key vector.

i	Parameter	Type	Description
1	decoding	discrete	Selects among nine decoding strategies, as described below
2	construction	discrete	Selects among three constructive heuristics
3	order	discrete	Exploration order: sequential or random-key induced
4	t	continuous	Threshold value for construction
5	α	continuous	Fraction of candidates in the restricted candidate list
6	c	integer	Number of constructions

4. RK-QUBO: Random-Key Optimizers for QUBO

This section describes RK-QUBO, the proposed random-key optimization framework for QUBO. The central design choice is that the random-key vector encodes not only the candidate solution but also the algorithmic configuration of the decoder. As a result, the continuous optimizer searches simultaneously for good solutions and for the combination of algorithmic components that produces them.

Let n be the number of binary variables in the QUBO instance. The random-key vector has dimension $n + p$, where the first n components encode the solution and the remaining $p = 6$ components encode algorithmic parameters. Formally, $k = (k_1, \ldots, k_n, k_{n+1}, \ldots, k_{n+p}) \in [0, 1]^{n+p}$. The first n components are used by the decoder to assign values to the binary variables or to define a processing order, depending on the decoding strategy. The remaining components are mapped to algorithmic parameters as described in Table 1. Discrete parameters (decoding type, construction type, neighborhood order) are obtained by uniformly partitioning the unit interval into as many segments as there are options. Continuous parameters (threshold t, greediness α) are used directly, while the integer parameter c is scaled to the range $[1, 100]$ by multiplying the key value by 100 and truncating.

4.1. Constructive Heuristics

Three constructive heuristics are implemented, selected via the construction parameter. All three receive as input the random-key vector k, the QUBO matrix Q, and a permutation π of the variable indices. The *Greedy by Marginal Cost* (*GMC0* and *GMC1*) constructive heuristics assign binary values to variables sequentially according to the order given by π, choosing at each step the value that minimizes the marginal contribution to the objective. GMC0 starts from the all-zeros solution $x = 0$ and evaluates flipping each variable from 0 to 1, while GMC1 starts from the all-ones solution $x = 1$ and evaluates flipping each variable from 1 to 0. The flip is accepted if it improves (or does not worsen) the objective. The marginal cost of flipping variable v from its current value x_v to $\bar{x}_v = 1 - x_v$ is

$$\Delta_v = Q_{vv}(1 - 2x_v) + (1 - 2x_v) \sum_{\substack{i=1 \\ i \neq v}}^{n} x_i (Q_{vi} + Q_{iv}).$$

In the *Probabilistic Construction* (*P*), each variable x_i is set to 1 independently with probability k_i, following a Bernoulli distribution. Unlike *GMC0* and *GMC1*, this

construction does not evaluate or apply marginal-cost flipping, i.e. the solution is obtained solely from the Bernoulli sampling step. This construction introduces stochasticity directly from the key values and is used in conjunction with the repeated construction.

4.2. Local Search

The local search procedure implements a first-improvement 1-flip strategy. The order in which variables are examined is controlled by the *order* parameter encoded in the random-key vector: variables are explored either sequentially from 1 to n, or according to the permutation π induced by the first n keys. When a flip is accepted (improving move), the neighborhood exploration restarts from the beginning of the ordering. The procedure terminates when no improving flip exists in the current solution. To avoid recomputing the objective function from scratch after each flip, the procedure maintains a vector of delta values $\delta \in \mathbb{R}^n$, where δ_v represents the change in objective value that would result from flipping variable v. After accepting a flip of variable v, the delta values of all other variables u are updated incrementally as

$$\delta_u \leftarrow \delta_u + \sigma_u \cdot \sigma_v \cdot (Q_{uv} + Q_{vu}),$$

where $\sigma_v = 1$ if x_v was set to 1 and $\sigma_v = -1$ if x_v was set to 0, and analogously for σ_u with respect to the current value of x_u before the update. This incremental update reduces the per-flip cost from $O(n^2)$, required by a naive recomputation of the full objective function, to $O(n)$ for updating the delta vector, while keeping it consistent throughout the search.

4.3. Decoding Strategies

Nine decoding strategies are implemented and selected via the decoding parameter encoded in the random-key vector. Four decoding strategies operate without local search. *T* sets each variable $x_i = 1$ if $k_i \geq 0.5$ and $x_i = 0$ otherwise, with the threshold fixed at 0.5. *TA* is identical but uses an adaptive threshold t extracted from component k_{n+4}, allowing the optimizer to learn a bias toward sparser or denser solutions. *C* applies the selected constructive heuristic to the permutation π induced by the first n keys. *RC* executes the construction c times. In each construction, a complete permutation of the n variables is built iteratively. At each step, a variable is sampled uniformly at random from the $\lceil \alpha \cdot n \rceil$ leading variables of the permutation induced by the keys, restricted to those not yet selected, and appended to the permutation being built. This is repeated until all n variables are selected. *RC* then returns the best of the c constructed solutions.

The remaining five strategies incorporate local search as a refinement step. *RLS* generates a random binary solution and applies local search directly to it. *CLS* and *TLS* use, respectively, the constructive heuristic and the fixed threshold ($t = 0.5$) to produce the initial solution before local search. *TALS* replaces the fixed threshold with the adaptive value t from k_{n+4}, combining solution bias with local refinement. Finally, *GRASP* [Feo and Resende 1995] executes c randomized constructions as in RC and applies local search to the best constructed solution, combining the diversification of repeated randomized construction with the intensification of local search.

4.4. Continuous Optimizers

The RK-QUBO decoder is used in conjunction with two continuous optimizers that explore the key space $[0, 1]^{n+p}$. The first is the so-called *Random-Key Optimizer* (also denoted by "RKO" in the results) [Chaves et al. 2025], a C++ framework that runs multiple

Table 2. Results for medium-scale instances ($n = 1000$).

Instance	RK-QUBO + RKO		RK-QUBO + NGOpt		ILSBQP		HHBQP	
	success	adev	success	adev	success	adev	success	adev
bqp1000.1	5/20	99.95	3/20	992.55	1/20	528.05	20/20	0.00
bqp1000.2	12/20	29.70	4/20	655.55	1/20	1042.75	20/20	0.00
bqp1000.3	6/20	123.95	4/20	1152.60	2/20	1090.65	20/20	0.00
bqp1000.4	10/20	46.70	2/20	878.80	0/20	665.40	20/20	0.00
bqp1000.5	2/20	144.20	0/20	576.75	0/20	1100.20	19/20	0.20
bqp1000.6	1/20	205.00	0/20	1444.10	0/20	1207.40	20/20	0.00
bqp1000.7	6/20	134.95	1/20	1353.10	0/20	992.70	20/20	0.00
bqp1000.8	0/20	219.90	0/20	2445.40	0/20	1465.80	19/20	1.15
bqp1000.9	1/20	110.65	1/20	1055.15	1/20	1419.80	20/20	0.00
bqp1000.10	0/20	220.40	1/20	1084.40	1/20	939.70	20/20	0.00
Average	—	133.54	—	1163.84	—	1045.25	—	0.14

metaheuristic algorithms in parallel threads, each operating over the same key space and sharing a common pool of elite solutions. The metaheuristics included in the framework encompass population-based and trajectory-based methods, and the elite pool promotes cooperation between them by providing high-quality starting points for intensification. The second is Nevergrad [Bennet et al. 2021], a gradient-free black-box optimization platform that includes differential evolution, CMA-ES, and Bayesian optimization variants in its portfolio of continuous optimizers.

5. Experimental Evaluation

All algorithms were implemented in C++ and compiled with g++ using the -O3 optimization flag. Experiments were conducted on a machine equipped with an Intel Core i7-10700 processor (8 physical cores, 2.90 GHz base frequency), 16 GB of RAM, and running Ubuntu 24.04 LTS. Each algorithm was executed 20 independent runs per instance, with a time limit of 60 seconds per run. Performance is reported in terms of two metrics: the average absolute deviation from the best known value (adev); and the success rate (success), defined as the number of runs in which the algorithm matched the best known value.

Experiments were conducted on 20 instances from the OR-Library [Beasley 1990], divided into two groups: ten instances with $n = 1000$ variables (bqp1000) and ten instances with $n = 2500$ variables (bqp2500). Both groups consist of sparse random instances with matrix density of 0.1, meaning that approximately 10% of the off-diagonal entries of Q are nonzero. The nonzero coefficients are integers drawn uniformly at random from $[-100, 100]$. These instances are widely used in the QUBO literature, which allows direct comparison with previous results.

Four algorithms are evaluated. *RK-QUBO + RKO* is the proposed approach using the Random-Key Optimizer as the continuous search engine. *RK-QUBO + NGOpt* uses the proposed approach with NGOpt as search engine, the default optimizer of the Nevergrad platform. Comparing these two configurations isolates the effect of the optimizer while keeping the decoder fixed. *ILSBQP* is an iterated local search algorithm for QUBO [de Souza and Ritt 2022], used as a baseline. *HHBQP* [de Souza and Ritt 2018]

Table 3. Results for large-scale instances ($n = 2500$).

Instance	RK-QUBO + RKO		RK-QUBO + NGOpt		ILSBQP		HHBQP	
	success	adev	success	adev	success	adev	success	adev
bqp2500.1	0/20	4147.00	0/20	3710.55	0/20	9995.65	20/20	0.00
bqp2500.2	0/20	4131.00	0/20	4161.15	0/20	9342.95	18/20	20.50
bqp2500.3	0/20	3418.55	0/20	4728.50	0/20	10242.60	20/20	0.00
bqp2500.4	0/20	3198.95	0/20	3540.90	0/20	6971.75	20/20	0.00
bqp2500.5	0/20	2839.85	0/20	5674.40	0/20	7144.05	20/20	0.00
bqp2500.6	0/20	3350.50	0/20	3421.55	0/20	6459.15	20/20	0.00
bqp2500.7	0/20	4216.00	0/20	4630.45	0/20	9744.00	20/20	0.00
bqp2500.8	0/20	2454.25	0/20	2650.00	0/20	6719.30	20/20	0.00
bqp2500.9	0/20	3027.50	0/20	3942.60	0/20	8190.25	20/20	0.00
bqp2500.10	0/20	4421.05	0/20	5285.55	0/20	9777.55	18/20	26.00
Average	—	3520.47	—	4174.57	—	8458.73	—	4.65

is a state-of-the-art hybrid heuristic obtained via automatic grammar-based algorithm design, used as the primary reference for solution quality.

Tables 2 and 3 report the results for the bqp1000 and bqp2500 instance sets, respectively. On the bqp1000 instances, RK-QUBO + RKO outperforms both ILSBQP and RK-QUBO + NGOpt by a substantial margin. The average absolute deviation from the best known values of RK-QUBO + RKO is 133.54, compared to 1045.25 for ILSBQP and 1163.84 for RK-QUBO + NGOpt. RK-QUBO + RKO also achieves nonzero success rates on eight of the ten instances, including 12 successes out of 20 runs on bqp1000.2 and 10 out of 20 on bqp1000.4, finding the best known solutions on the easier instances of this group. The random-key framework with the RKO solver present satisfactory performance on medium-size instances.

The comparison between the proposed approach under different optimizers (i.e., RK-QUBO + RKO and RK-QUBO + NGOpt) is informative, since both use the same decoder and the performance difference is explained by the use of the distinct search engines. The RKO optimizer outperforms NGOpt on all bqp1000 instances, suggesting that the metaheuristic portfolio and elite-pool cooperation implemented in RKO are better suited to explore the random-key space for this problem, in comparison to the black-box continuous optimizers available in Nevergrad.

On the bqp2500 instances, none of the proposed methods achieves the best known value within the time limit of 60 seconds, reflecting the increased difficulty of large-scale instances. In this setting, RK-QUBO + RKO obtains an average absolute deviation of 3520.47, which is lower than RK-QUBO + NGOpt (4174.57) and ILSBQP (8458.73). RK-QUBO + RKO outperforms ILSBQP on all instances, with reductions in absolute deviations ranging from 37.6% (bqp2500.4) to 75.5% (bqp2500.3). The relative ordering of methods is consistent across both instance sizes.

On the other hand, HHBQP achieves the best performance on both instance sets, with average absolute deviations of 0.14 and 4.65 for bqp1000 and bqp2500, respectively, and success rates of 20/20 on most instances. The gap between HHBQP and RK-QUBO + RKO indicates that the current implementation of RK-QUBO has room for improvement, particularly with respect to the decoder design. More specialized constructive

heuristics, more sophisticated local search methods, or population-based approaches that exploit problem structure may reduce this gap in future work.

6. Concluding Remarks

This paper presented RK-QUBO, a random-key optimization framework for the Quadratic Unconstrained Binary Optimization problem. The proposed approach encodes both the candidate solution and the algorithmic configuration of the decoder in a single continuous random-key vector, allowing a continuous optimizer to search simultaneously over the solution space and the space of decoding strategies. Nine decoding strategies were proposed, combining threshold-based assignment, greedy constructive heuristics, and first-improvement local search in different configurations. The framework was evaluated using two continuous optimizers, namely the Random-Key Optimizer (RKO) and the Nevergrad platform (NGOpt), and compared against an iterated local search baseline (ILSBQP) and a state-of-the-art hybrid heuristic (HHBQP) on benchmark instances from the OR-Library.

The main contributions of this work are threefold. First, RK-QUBO provides a generic and modular framework for QUBO in which all problem-specific knowledge is encapsulated in the decoder, and any continuous optimizer can be integrated without modification. Second, the encoding of algorithmic parameters in the random-key vector enables automatic selection and tuning of decoding components at search time, eliminating the need for offline parameter configuration. Third, since QUBO captures a wide range of combinatorial problems, the proposed framework can be applied to any problem that admits a QUBO formulation, broadening its practical scope beyond the instances studied here.

The experimental results show that the proposed approach consistently outperforms ILSBQP on both medium- and large-scale instances, despite relying on simple and general-purpose decoding components. This result indicates the random-key paradigm as a promising approach for binary optimization. At the same time, a substantial gap remains between the proposed approach and the state-of-the-art method HHBQP. This gap suggests that the generic nature of the current decoders limits solution quality, and that more problem-aware components could improve the competitiveness of the framework. The comparison between RK-QUBO + RKO and RK-QUBO + NGOpt further shows that the choice of continuous optimizer has an impact on performance.

About AI assistance. The authors used Claude Sonnet 4.6 as a writing assistance tool during the preparation of this manuscript. The tool was employed to support text structuring, academic writing, grammar correction, and figure preparation. All content generated with AI assistance was critically reviewed and validated by the authors, who take full responsibility for the accuracy and integrity of the work.

References

Alkhamis, T. M., Hasan, M., and Ahmed, M. A. (1998). Simulated annealing for the unconstrained quadratic pseudo-Boolean function. *European Journal of Operational Research*, 108(3):641–652.

Andrade, C. E., Toso, R. F., Gonçalves, J. F., and Resende, M. G. C. (2021). The multi-parent biased random-key genetic algorithm with implicit path-relinking and its real-world applications. *European Journal of Operational Research*, 289(1):17–30.

Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6(2):154–160.

Beasley, J. E. (1990). OR-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072.

Bennet, P., Doerr, C., Moreau, A., Rapin, J., Teytaud, F., and Teytaud, O. (2021). Nevergrad: Black-box optimization platform. *ACM SIGEVOlution*, 14(1):8–15.

Brubaker, K., Booth, K. E., Schuetz, M. J. A., Loick, P., Shen, J., Ramamurthy, A., and Paschos, G. (2025). Optimization of next-day delivery coverage using constraint programming and random key optimizers. Preprint arXiv:2504.18749 [math.OC].

Carvalho, M., Melo, R. A., Resende, M. G. C., Santos, M. C., and Toso, R. F. (2025). The connected grundy coloring problem: Formulations and a local-search enhanced biased random-key genetic algorithm. *Computers & Operations Research*, 183:107136.

Chaves, A. A., Resende, M. G. C., Schmidt, C. E., Brubaker, J. K., Katzgraber, H. G., and Schuetz, M. J. A. (2026). Applying a random-key optimizer on mixed integer programs. Preprint arXiv:2602.22173 [math.OC].

Chaves, A. A., Resende, M. G. C., Schuetz, M. J. A., Brubaker, J. K., Katzgraber, H. G., de Arruda, E. F., and Silva, R. M. A. (2025). A random-key optimizer for combinatorial optimization. *Journal of Heuristics*, 31(4):32.

Chaves, A. A., Resende, M. G. C., and Silva, R. M. A. (2024). A random-key GRASP for combinatorial optimization. *J. Nonlinear and Variational Analysis*, 8:855–881.

de Souza, M. and Ritt, M. (2018). Automatic grammar-based design of heuristic algorithms for unconstrained binary quadratic programming. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 67–84. Springer.

de Souza, M. and Ritt, M. (2022). Improved regression models for algorithm configuration. In *The Genetic and Evolutionary Computation Conference*, pages 222–231.

dos Santos Becker, C. and de Souza, M. (2026). Roteamento de drones para busca de vítimas em cenários de desastre. *Revista Tecnologia e Sociedade*, 21(67):379–397.

Feo, T. A. and Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133.

Festa, P., Guerriero, F., Resende, M. G. C., and Scalzo, E. (2023). A BRKGA with implicit path-relinking for the vehicle routing problem with occasional drivers and time windows. In *Metaheuristics. MIC 2022*, volume 13838 of *Lecture Notes in Computer Science*, pages 17–31. Springer.

Fontes, D. B. M. M. and Homayouni, S. M. (2023). A bi-objective multi-population biased random-key genetic algorithm for joint scheduling quay cranes and speed adjustable vehicles in container terminals. *Flexible Services and Manufacturing Journal*, 35(1):241–268.

Glover, F., Kochenberger, G., and Du, Y. (2019). Quantum bridge analytics I: A tutorial on formulating and using QUBO models. *4OR*, 17(4):335–371.

Glover, F., Kochenberger, G. A., and Alidaee, B. (1998). Adaptive memory tabu search for binary quadratic programs. *Management Science*, 44(3):336–345.

Glover, F., Lü, Z., and Hao, J.-K. (2010). Diversification-driven tabu search for unconstrained binary quadratic problems. *4OR*, 8(3):239–253.

Gonçalves, J. F. and Resende, M. G. C. (2011). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5):487–525.

Gonçalves, J. F. and Wäscher, G. (2020). A MIP model and a biased random-key genetic algorithm based approach for a two-dimensional cutting problem with defects. *European Journal of Operational Research*, 286(3):867–882.

Homayouni, S. M., Fontes, D. B. M. M., and Gonçalves, J. F. (2023). A multistart biased random-key genetic algorithm for the flexible job shop scheduling problem with transportation. *International Transactions in Operational Research*.

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(1):58–81.

Londe, M. A., Pessoa, L. S., Andrade, C. E., and Resende, M. G. C. (2024). Early years of biased random-key genetic algorithms: A systematic review. *Journal of Global Optimization*.

Mangussi, A. D., Pola, H., Macedo, H. G., Julião, L. A., Proença, M. P. T., Gianfelice, P. R. L., Salezze, B. V., and Chaves, A. A. (2023). Meta-heurísticas via chaves aleatórias aplicadas ao problema de localização de hubs em árvore. In *Anais do Simpósio Brasileiro de Pesquisa Operacional*, São José dos Campos. Galoá.

Melo, R. A., Queiroz, M. F., and Ribeiro, C. C. (2023). A biased random-key genetic algorithm for the minimum quasi-clique partitioning problem. *Annals of Operations Research*.

Punnen, A. P., editor (2022). *The Quadratic Unconstrained Binary Optimization Problem: Theory, Algorithms, and Applications*. Springer.

Rapin, J. and Teytaud, O. (2018). Nevergrad - A gradient-free optimization platform. `https://GitHub.com/FacebookResearch/Nevergrad`.

Santos, N. A., Jeske, M., and Chaves, A. A. (2026). Random-key optimizer and linearization for the quadratic multiple constraints variable-sized bin packing problem. *Computers & Operations Research*, page 107482.

Schuetz, M. J. A., Brubaker, J. K., Montagu, H., van Dijk, Y., Klepsch, J., Ross, P., Luckow, A., Resende, M. G. C., and Katzgraber, H. G. (2022). Optimization of robot trajectory planning with nature-inspired and hybrid quantum algorithms. *Physical Review Applied*, 18:054045.

Silva, E. M., Chaves, A. A., de Araujo, S. A., and Jans, R. (2025). Random-key optimizer with reinforcement learning for the capacitated multi-period cutting stock problem with setup cost. *Computers & Operations Research*, 183:107159.

Wang, Y., Lü, Z., Glover, F., and Hao, J.-K. (2012). Path relinking for unconstrained binary quadratic programming. *European Journal of Operational Research*, 223(3):595–604.