Path Integral Quantum Annealing Optimizations Validated on 0-1 Multidimensional Knapsack Problem

Path Integral Quantum Annealing Optimizations Validated on 0-1 Multidimensional Knapsack Problem

Evelina Forno, Riccardo Pignari, Vittorio Fra, Enrico Macii, *Fellow, IEEE*, Gianvito Urgese, *Senior Member, IEEE*

***Abstract*—Quantum Annealing (QA) is a metaheuristic designed to enhance Simulated Annealing by leveraging concepts from quantum mechanics, improving parallelization on classical computers. Studies have shown promising results for this technique in the field of NP-hard problems and constrained optimization. In this article, we examine Path Integral Quantum Annealing (PIQA), a well-known technique for simulating QA on conventional computers. We then propose optimizations to the algorithm, offering hardware software developers a suite of parallelization techniques evaluated for their effectiveness in enhancing quality and speed. The proposed approach encompasses four distinct degrees of optimization, leveraging techniques based on multiple-trial parallelism and a novel pre-optimization method. The article further proposes a methodology for handling multiple instances within the search space, whereby problem data is replicated into slices and allocated to concurrent processes during the simulation. Through empirical trials, we evaluate the impact of our optimization techniques on the convergence speed of the algorithm compared to unoptimized PIQA, using the Multidimensional Knapsack Problem as a benchmark. Our findings show that these optimizations, applied individually or collectively, enable the algorithm to achieve equal or superior results with fewer simulation steps. Overall, the results highlight the potential for future implementations of optimized PIQA on dedicated hardware.**

***Index Terms*—Simulated Quantum Annealing, Path Integral Quantum Annealing, multiple-trial parallelism, metaheuristics, constrained optimization, Multidimensional Knapsack Problem, parallelized computation, Simulated Annealing.**

I. Introduction

OPTIMIZATION problems — concerning the search for an optimal solution from a space of possible candidates — play a key role in our society, with multiple applications in science, engineering and economics [1]. As more and more degrees of freedom are introduced in order to accurately model real-world problems, optimization problems can greatly increase in complexity. For instance, in addition to the optimization process, some variables in the system may be subject to one or more constraints that require satisfaction in order for the solution to be valid. As the search space of these *constrained-optimization problems* (COP) grows exponentially with their number of variables and constraints, brute-force methods quickly become unfeasible; indeed, the computational complexity of many COPs characterizes them as *NP-hard*. One such example is the 0-1 Multidimensional Knapsack Problem (MKP) [2]–[4], with applications such as task scheduling [5], packing problems [6], resource allocation [7]. While efficient exact solvers for the MKP exist, even the best-performing algorithms still report quadratic complexity relative to the size of the problem ($O(m^2n)$ [8]), which can translate to several hours of computation time for complex problem instances [9]. For this reason, MKP remains a compelling benchmark for fast, approximate heuristics such as annealing algorithms.

Although new unconventional approaches are emerging [10], [11], a popular tool for tackling COPs are *metaheuristic* methods. These algorithms, usually incorporating some variety of stochastic optimization, can reach a high-quality optimal solution in reduced time and without requiring any prior information about the search space [12]; the tradeoff is that, unlike exact optimization methods, metaheuristics offer no guarantee of finding the absolute optimum solution [13].

Quantum Annealing (QA) is an extension of the thermal annealing model formulated to include quantum effects. The most well-known formulation of the algorithm [14] was applied to the Ising model with transverse external field, and many proofs of concept to follow have involved applying QA to other NP-hard discrete optimization problems, such as the Traveling Salesman Problem, the MKP [15]. In addition, QA can be applied to many biology problems [16] and shows promise as a machine learning classification tool [17]. The field of artificial intelligence is also showing a growing interest in QA as a tool for training [18] or complete implementation [19] of neural network models.

Early realizations of physical quantum annealers, which have become available in recent years, are able to perform optimization of many types of problems, provided that they can be mapped onto an Ising-type lattice. However, the use of physical quantum annealers for the solution of optimization problems is still in its infancy, due to limitations in the number of qubits, difficulties in the physical realization of fully adiabatic systems [20], non-idealities in the mapping of such problems to the machine graph [21], and more [22]. On the other hand, research on the simulation of QA on classical computers marches on: while many academic investigators employ computer simulations to explore the actual potential of quantum speedup [23], [24], industry engineers are hard at work on hardware accelerators for QA [25] and "quantum-

The authors are with the Electronic Design Automation (EDA) Group at Politecnico di Torino, Turin, Italy (e-mail: evelina.forno@polito.it, gianvito.urgese@polito.it).

Manuscript received April 05, 2023; revised XXXX XX, XXXX.

0000–0000/00$00.00 © 2021 IEEE

inspired" [26] digital annealers [27], promising accessible COP solving with a moderate technological investment.

In this article, we will focus our attention on a popular technique for simulating Quantum Annealing, namely Path Integral Quantum Annealing (PIQA) [28]. PIQA presents unique opportunities for parallelization because of its population-based nature. In previous work by the authors [29], hardware acceleration of PIQA was achieved by independently processing replicas using distributed hardware units and allowing independent simulation with infrequent synchronization. However, this SIMD-style partitioning is not the only possible approach: thanks to the many similarities of PIQA to Simulated Annealing (SA), we may take further inspiration from the many proposed solutions for parallelization of said algorithm. While — to our knowledge — no ad hoc solutions for the acceleration of PIQA on CPU have been proposed, many multi-thread implementations for SA have the potential to be seamlessly adapted to PIQA.

There are two main approaches to SA parallelization [30], [31]. The first is *move acceleration*, which exploits functional parallelism by splitting intensive tasks into sub-tasks and distributing them among several processors. It preserves the sequentiality of the simulation and has the same convergence properties as single-core SA while obtaining a moderate speedup. This type of approach is also referred to as *single-trial parallelism* to distinguish it from the second approach, *multiple-trial parallelism*. In this case, at each step of simulation, several moves are evaluated simultaneously; then, only one of the moves is chosen and applied to the partial solution. The choice criteria for the move to apply can vary and is determined by the algorithm designer. While this approach requires synchronization at every step, it aims instead to reduce the overall number of steps by increasing the probability that a feasible move is found at every iteration. The other technique that can be used to speed up the convergence of optimization algorithms is *pre-optimization*; here, a partial solution is built first, so that — in the case of PIQA — the annealing simulation is starting from an advantageous point in the solution space, and the whole annealing time is dedicated to solution refinement. The pre-optimized solution is usually built via a fast greedy algorithm.

This study aims to show that the aforementioned techniques can be effectively applied to PIQA and, when combined with the intrinsic characteristics of PIQA as a population-based heuristic, represent an opportunity to not only achieve speedup, but also improve exploration of the search space and the algorithm convergence. In Section II, we introduce different optimization algorithms based on annealing; we also give a description of the Multidimensional Knapsack Problem and clarifications on previous work concerning PIQA-based solvers for the MKP. Section III describes the PIQA optimization techniques presented in this work. Section IV contains a detailed benchmark of the performance of the optimized PIQA on the MKP, while Section V outlines a few key observations about these experimental results. Finally, Section VI contains our conclusions and reflections on future work.

II. Background

In this Section, we review the key concepts underlying our work by addressing them in dedicated subsections. In Section II-A, we start from the general definition of annealing to describe the key differences between SA (II-A1), QA (II-A2), and PIQA (II-A3). In Section II-B, we give a description of the MKP, listing its formulation, real-life applications, and solver algorithms. Lastly, in Section II-C, we explain the method employed to apply PIQA to the MKP.

A. Annealing

The term *annealing* refers to a thermal process used in the fields of materials science and engineering to act on certain physical properties of an object during a manufacturing process [32]. Annealing is composed of two phases: heating and cooling. The former exploits high temperatures to make atoms absorb large amounts of energy; the latter leads them into a configuration of minimum energy through a relaxation process. The speed at which cooling down is performed makes the distinction between annealing and quenching. The key feature distinguishing the annealing process is the possibility for atoms to find configurations that minimize energy, not only with respect to the nearest neighbors, but also to distant atoms in the system, thanks to the low cooling rate.

The process of energy absorption with subsequent relaxation in presence of decreasing temperatures can be described through the Hamiltonian formalism. Specifically, physical models like the *Ising model* [33] can be adopted, which allows to describe atoms as singular units with a discrete state variable generally referred to as *spin*. The original formulation of the Ising model, given a pair (i, j) of atoms, is described in (1):

$$H = -\sum_{<ij>} J_{ij} s_i s_j, \tag{1}$$

where s_i is the individual spin, while J_{ij} is the *coupling parameter* representing the energy contribution produced by spin alignment ($s_i = s_j$) or misalignment ($s_i = -s_j$). With such formulation, we can simulate the behavior of a system undergoing annealing with the aid of probabilistic methods based on Markov Chain Monte Carlo (MCMC), such as the Metropolis-Hastings algorithm [34].

The natural ability of annealing to place the system in a thermodynamically advantageous state has led to its adoption as a heuristic procedure in optimization processes, leading to the SA algorithm [33].

Following SA, more formulations of algorithms based on annealing have been proposed. Drawing inspiration from physical phenomena other than thermal, a quantum version, i.e. the QA algorithm [14], and its reformulation based on statistical mechanics, the PIQA algorithm [28], have been introduced. Having SA, QA and PIQA a common root, what distinguishes them is the phenomenon which drives the exploration in the configuration space of the global optimum candidates. To further outline and clarify the differences between the three, we shall use the Ising model as a reference.

1) Simulated Annealing: In SA, (1) is directly taken as the Hamiltonian of the system, with all the energy represented by the potential energy term U so that $U = -\sum_{<ij>} J_{ij} s_i s_j$. Each site i is described by a spin parameter $s_i = \pm 1$ which identifies its state among the configuration spins $\overline{s} = \{s_1, s_2, s_3, ...\}$. The notation $< ij >$ highlights that the sum is over all the (i, j) pairs of nearest neighbors. The corresponding i^{th} energy contribution is defined by the relationship between the state s_i and its nearest neighbor s_j through the coupling parameter $J_{ij} \in \mathbb{R}$. The lower this sum, the more energetically advantageous the state of the system.

The initial energy of the system is defined by the user via an update algorithm, as the Metropolis–Hastings [34], through a temperature parameter T or an equivalent quantity with the same effect of introducing some energy contribution and consequent agitation. For high temperatures, this corresponds to having a high probability of occupying high energy states represented by the value $P_{\overline{s}}$ which define the probability that a state $\overline{s}$ will be occupied, which in the framework of optimization problems translates into a large set of candidate states for sampling. As the temperature decreases, so does the probability of accepting high-energy solutions, resulting in a more and more reduced and confined movement across the energy landscape which leads to finer selections until the solution is minimized as $T \to 0$. Hence, the dynamic evolution of the SA system depends on thermal hopping, i.e. the phenomenon of overcoming an energy barrier, or escaping an energy well, as a consequence of the thermal agitation supplied to the system.

2) Quantum Annealing: Differently from SA, in Quantum Annealing the dynamic evolution and the related capability of overcoming an energy barrier or escaping an energy well depends on phenomena such as quantum tunneling occurring at constant temperature [14]. The Hamiltonian describing the state of the system is written through the formalism of the Pauli spin matrices which define the configurations $\overline{\sigma} = \{\overline{\sigma_1}, \overline{\sigma_2}, \overline{\sigma_3}, ...\}$. With respect to SA, QA introduces, in addition to the potential term U, a so-called kinetic energy term K which encloses the quantum contribution. Such term is the result of the interaction of the spin state σ_i^x with a transversal field Γ [35]. The total energy is hence defined through (2) as:

$$H = U + K = -\sum_{<ij>} J_{ij} \sigma_i^z \sigma_j^z - \Gamma \sum_i \sigma_i^x \tag{2}$$

With this formulation, the probability $P_{\overline{\sigma}}$ of occupation of an energy state depends on the quantum term containing Γ, rather than just on temperature as in SA.

Despite the different origin and physical meaning, the scheduling process of the Γ parameter follows the same dynamics as for T in SA, with very high values of Γ can be chosen as initial configuration. From the structure of H in (2), it can be observed that, in such case, the dominant term is the kinetic one, which tends to make all spins $\overline{\sigma_i}$ align to the transversal field direction regardless of the interaction that the spins may have with their nearest neighbors. Then, as the parameter Γ decreases, the kinetic contribution K becomes comparable with, or smaller than, the potential term U, thus leading to a relaxation of the quantum effect on the overall energy of the system.

3) Path Integral Quantum Annealing: PIQA is a reformulation of QA from the perspective of mechanical statistics, in which the quantum definition of the partition function Z is applied through (3). By substituting the various terms and carrying out the approximations reported in the work by Martoňák et al. [28], the resulting Hamiltonian is given by (4).

$$Z = \operatorname{Tr} e^{-\beta H} \tag{3}$$

$$H = -\sum_{k=1}^{N} \left(\sum_{<ij>} J_{ij} s_i^k s_j^k + J^{\perp} \sum_i s_i^k s_i^{k+1} \right) \tag{4}$$

Comparing (2) and (4), the first substantial difference is the summation over k. This term is the direct result of applying the formalism introduced in Statistical Mechanics by Gibbs [36] on the QA model, which takes into account the energy contribution deriving from N replicas of the system. The second difference is the inclusion of the spin interaction between the various replicas, denoting with s_i^k the value of the i^{th} spin in the replies k^{th} replicas. Regarding the other terms, we again have the potential term $U = \sum_{<ij>} J_{ij} s_i^k s_j^k$, which models the energy contribution between spins belonging to the same replica, and a kinetic term $K = J^{\perp} \sum_i s_i^k s_i^{k+1}$ which instead determines the contribution of interactions between spins belonging to the different copies with $J^{\perp} = -\frac{NT}{2} \ln \tanh \frac{\Gamma}{NT}$.

It should be noted that the Γ parameter present in $J^{\perp}$ is the same as the one introduced in QA, same for the T parameter. Indeed, the scheduling process of $J^{\perp}$ is increasing in time, unlike that of Γ which is decreasing. Consequently, each replica of the system initially appears to be isolated from the others, allowing an evolution of the spin configurations $\overline{s^k}$ independent of the neighboring replicas. As $J^{\perp}$ increases, the various spins develop a stronger correlation between them which drives the ones with an unfavorable energy configuration into a more energetically advantageous condition. At the macroscopic level, this translates to a collapse of the various replicas in a single configuration: the one with the minimum energy found in the various evaluation paths. The resulting algorithmic flow is outlined in Fig. II-A3.

The presence of replicas in a problem of this entity allows, as a collateral advantage, to speed up the convergence of the solver to an optimal solution, as it has been analytically reported by Martoňák et al. in [28] where an exponent of convergence $\zeta_{PIQA} > \zeta_{SA}$ with $\zeta_{SA} \leq 2$ is shown.

Although in PIQA, as well as in QA, temperature does not play an active role in the generation of thermodynamic states, it still plays an important role in allowing the system to explore the energy panorama by tunneling effect in adiabatic conditions since the higher the temperature, the higher the probability of tunneling.

B. Multidimensional Knapsack Problem

The Knapsack (or Rucksack) Problem is a well-known problem in combinatorial optimization. Given a set of items, each

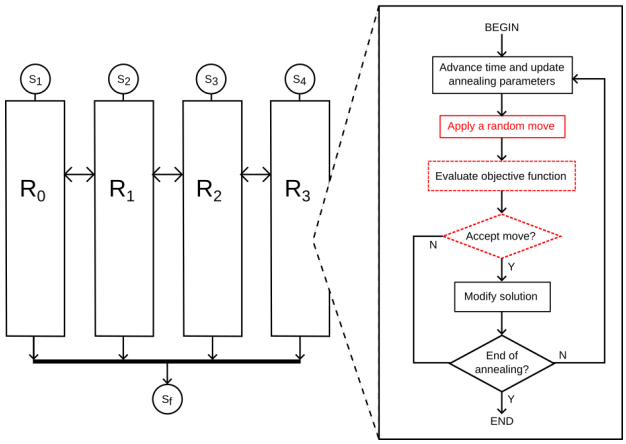

Fig. 1. Block diagram outlining the PIQA algorithm. The figure highlights the presence of a number of correlated replicas R_i, each performing independent MCMC trials in parallel. The connecting arrows represent the exchange of state information, which contributes to the Hamiltonian with a weight $= J_\perp$.

with an associated *weight* and *value*, the solver is requested to select the items to insert in the knapsack such that (a) a maximum weight is not exceeded and (b) the total value of the collection is maximized. It is a classic resource allocation problem that identifies a number of variants by imposing or removing a limit on duplicate items, multiple constraints, etc. The 0-1 MKP is one such variation, where only one copy of each item is allowed and multiple weight constraints must be met:

Given

N : number of items to pack,
U_i : vector of item values (size N),
W_j : vector of constraints on item weights (size C),
$[P_{ij}]$: matrix of positive weights (size $N \times C$), where each entry P_{ij} represents the cost of inserting item i with respect to constraint j,

find an assignment $x = (x_i)_{1 \leq i \leq N} \in \{0,1\}^N$ such that the total value $\sum_{i=1}^{N} U_i x_i$ is maximized, while respecting the constraint set: $\sum_{i=1}^{N} P_{ij} x_i \leq W_j, \forall j \in [\![1, C]\!]$.

A variety of successful exact solvers for MKP exist, the great majority of them based on branch-and-bound algorithms; CORAL [9] is an example of this type of approach. However, exact methods still struggle with large sized instances, which require overly long computation times to solve to optimality. Therefore, the study of approximate methods draws great interest, especially in the realm of metaheuristics [37]. Successful approaches include algorithms based on SA [38], Genetic Algorithms [39], Particle Swarm [40], [41], Ant Colony [42], and Tabu Search [43].

C. Quantum Annealing of the Multidimensional Knapsack Problem

When applying PIQA to the MKP, each replica works on a different knapsack; i.e., each worker is assigned an independent N-spin integer vector x. In [15], it was shown that the simplified path-integral Hamiltonian can be used by only allowing moves that take our system to another valid configuration, i.e., only inserting and swapping items that do not violate any of the multidimensional constraints. This measure is taken in order to eliminate the need to account for the constraints in the problem Hamiltonian, greatly simplifying its equation. The fulfillment of constraints is instead enforced by starting with an empty knapsack (all spins = 0) and adding random items until constraints are exceeded; then, all subsequent trials require swapping an item in the bag for a random one outside, provided that constraints are still met.

The classical portion of the Hamiltonian can then be simplified to:

$$H_{\text{pot, MKP}} = \sum_{i=1}^{n} U_i x_i \tag{5}$$

that is, the total value of the items added to the knapsack.

A possible optimization for the MKP-QA solver is the Restrictive Quantum Annealing (RQA) paradigm, also proposed in [15]. RQA consists in disallowing the removal of elements that appear in a given percentage of replicas (i.e., the *blocking frequency*); it works on the assumption that if a spin state appears in many partial solutions, it is highly likely that it will be part of the final solution. In practice, the optimal blocking frequency varies depending on the benchmark problem, and it would be up to the user to run the algorithm with varying frequency values and find out the best value. However, as a rule of thumb, a frequency of 90% has been observed to be effective on a wide range of problems.

III. Implementation

In this Section, we introduce different pre-optimization techniques together with the new methodology proposed by our study. Two main methods can be found in literature as pre-optimization strategies for the SA algorithm. The first one is the use of pseudo-utility equations [37], [41], [42], [44], which allows to increase the convergence efficiency of the algorithm, discussed in Section III-A. The second technique is multiple-trial parallelism [15], [30], presented in Section III-B, along with the criteria for move selection among the results of said parallel trials. These criteria are reported in Section III-C. In Section III-D, the strategy we propose to encourage a more diverse exploration of the state space, called *sliced processing*, is discussed. A full flowchart of the MKP solver based on PIQA, highlighting the optimization blocks, is available in Fig. III.

A. Pre-Optimization

Combinatorial optimization problems can benefit of pre-optimization tactics. In the case of the MKP, this translates into starting the annealing with a pre-filled knapsack, containing a subset of items selected through analysis of the problem data.

Several different criteria can be used to perform this pre-optimization. We selected the Kochenberger, McCarl and Wyman (KMW) heuristic described in [44], which consists in ordering items based on their *pseudo-utility*, a parameter that combines in a single fitness metrics the utility (i.e., the value) and the weight with respect to the partially filled knapsack (Fig. III.A). The items are then inserted into the

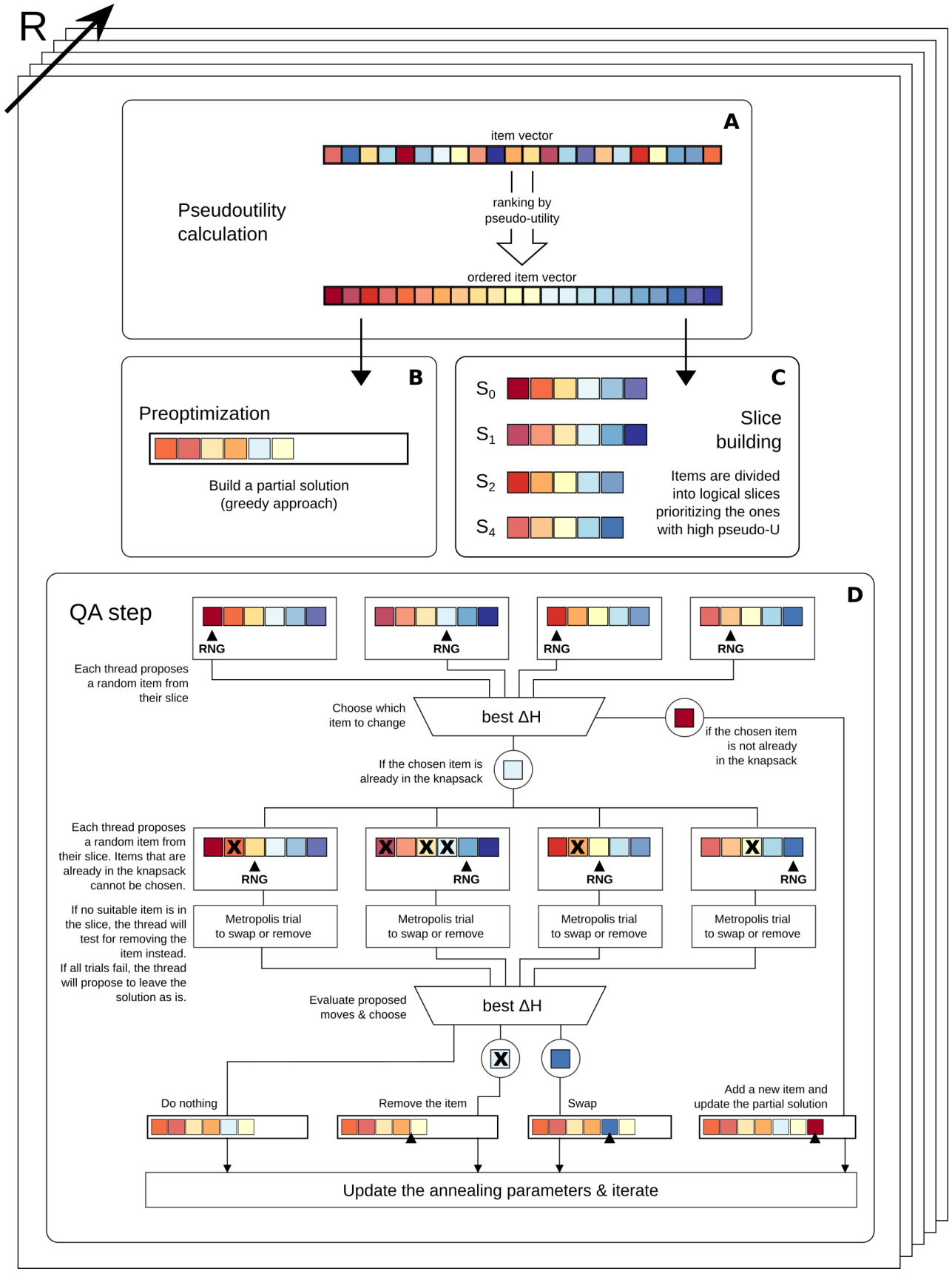

Fig. 2. Block diagram of the algorithm flow for the MKP solver based on PIQA. The *pseudo-utility calculation* (A) is performed at the beginning of the program, followed by optional *pre-optimization* (B) and *slice building* (C) phases. In each *simulation step* (D), two regions of multiple-trial parallelism are launched, each followed by a customizable filter which selects the move to make. The "R" axis (marked in the top left) visually represents how the entire process is performed for each PIQA replica.

knapsack in decreasing order of pseudo-utility, and the pre-optimization stops when at least one of the constraints is exceeded (Fig. III.B).

The pseudo-utility for each item i is calculated by (6):

$$psU_i = \frac{U_i}{\sum_{k=1}^{C}(P_{k,i}/rc_k)}, \tag{6}$$

where U_i is the utility of item i, $P_{k,i}$ is the weight of item i under the k^{th} constraint, and rc is a vector of size C that quantifies the remaining capacity available for each constraint, as defined in (7):

$$rc_k = 1 - \frac{\sum_{i=0}^{N} P_{k,i} \cdot x_i}{W_k}, \tag{7}$$

where x_i is a binary vector of size N which contains 1 at position i if the item is in the knapsack, 0 if it is not; W_k is the total capacity for constraint k.

Because of the need to re-evaluate the rc vector, the pseudo-utility for the KMW heuristic needs to be recomputed every time a new item is inserted into the knapsack. In order to reduce the computational load of the pre-optimization algorithm, we propose to substitute (6) with a simpler pseudo-utility calculation, as represented in (8):

$$psU_i = \frac{U_i}{\sum_{k=0}^{C}(P_{k,i}/\sum_{j=0}^{N} P_{k,j})}. \tag{8}$$

This pseudo-utility vector remains independent of the status of the knapsack, therefore it can be computed only once, at the beginning of the pre-optimization sequence. In order to distinguish the two pre-optimization methods used within this work, we dubbed the KMW heuristic using (6) *Dynamic pre-optimization*, and the one using (8) *Static pre-optimization*.

B. Multiple-Trial Parallelism

Because of the similarities between PIQA and SA, techniques for the acceleration of SA can be applied to PIQA as well. Parallelization of SA [30] can generally be attained via *single-trial parallelism* or *multiple-trial parallelism*. In the first case, the algorithm is accelerated by exploiting functional parallelism, and the sequentiality of the simulation is preserved. By contrast, multiple-trial parallelism is based on the idea that, by executing the same simulation step on several processors at once and selecting one of the results, the overall number of successful random trials will increase: the execution time on a von Neumann computer is generally not reduced by this approach, but the number of simulation steps needed to reach acceptable results decreases, which will speed up the execution on a dedicated architecture.

In this work, we applied multiple-trial parallelism to a PIQA-based MKP solver based on the algorithm proposed in [15]. Specifically, we split each update move into two phases: `tryInsert` and `trySwap`. Each of these two phases identifies a parallel region, as depicted in Fig. III.D.

In the `tryInsert` phase, each thread evaluates a random item r and calculates the difference in energy ΔH obtained by "flipping the spin" corresponding to the item. In this context, "flipping the spin" corresponding to r means removing r if it is already inside the backpack, inserting it otherwise. Every thread proposes a move for every replica in the PIQA array. Moves that would cause one or more of the constraints to be violated are refused during the choice of r, so that each thread exits this parallel region returning a feasible move.

At the end of `tryInsert`, all proposed moves are compared, and only one is selected (the selection criteria are outlined in sec. III-C). If the selected move proposes to *insert* a new item, it is immediately accepted, as it always improves the overall value of the knapsack; the corresponding knapsack is updated with the new item, and no more changes are made to the replica until the next simulation step.

On the other hand, if the selected move proposes to *remove* an item, the second parallel region `trySwap` is activated. In this phase, each thread tries to find a new random item outside the knapsack which can be swapped with the item to be removed without violating constraints; the PIQA dynamics are applied here to allow the acceptance of moves that take the system out of local optima, and the random update is accepted only if its respective Metropolis-Hastings trial is successful. At the end of `trySwap`, each thread can propose either to *swap* the item with a new one, *remove* the item, or *do nothing*.

The proposed moves are evaluated again, and the chosen one is finally applied to the replica. Once all replicas have updated, the algorithm proceeds to the next simulation step.

C. Insert/swap criteria

At the end of the `tryInsert` and `trySwap` phases, only one of the proposed moves must be selected. In this work, we implement 2 selection criteria: `random` and `best`. When `random` selection is in use, we pick a random move among the proposed ones; this is equivalent to running a single-thread, unoptimized PIQA instance that evaluates a single random update at every step. With `best` selection, the difference in energy ΔH contributed by each move is compared, and we choose the move with the most favorable result. These two selection methods can be independently applied to either or both the `tryInsert` and `trySwap` phases.

D. Sliced processing

To further improve the effects of parallelization, we propose a strategy called *Sliced processing*. The aim of this technique is to improve the random item selection step by trying to enforce fair choice among all items. With this aim, each thread is assigned a subset (*slice*) of items that it is limited to choose from during the `tryInsert` and `trySwap` steps. This assignment can be made according to several strategies; for example, we can use a random assignment, or we can apply the pseudo-utility ranking based on (8) to distribute the items among threads in such a way that the average pseudo-utility of each slice is similar.

Other than enforcing diversity in the items proposed by the various threads at each simulation step, this approach also facilitates the implementation of the algorithm in systems that do not have a shared memory, as each thread only needs to store the weights of the items it has been assigned.

In our implementation, the optimizations described in III-A, III-B, III-C and III-D can be independently applied to each different replica in the Quantum Annealing simulation array.

IV. Evaluation

The strategies discussed in Section III allow us to define 4 degrees of freedom, as portrayed in Fig. IV. The corresponding parameters are defined as follows:

- **L1:** Specifies whether the system undergoes optimization. The options are *No pre-optimization*, *Dynamic pre-optimization* or *Static pre-optimization*.
- **L2:** Specifies the criteria used for sliced processing. The options are:
  - *no slices*;
  - *sliced-Balanced*: slices are built using the static pseudo-utility ranking from (8). The items are distributed in such a way that the average pseudo-utility per slice is roughly the same;
  - *sliced-Ordered*: slices are built by leaving the items in the same order as they are given in the problem;
  - *sliced-Random*: slices are built by ranking the items in a random order.
- **L3:** Specifies the criteria used to select the item in the `tryInsert` phase. *insert-Random* selects a random valid move among those proposed by threads; *insert-Best* selects the move with the more positive ΔH.
- **L4:** Specifies the criteria used to select the item in the `trySwap` phase. *insert-Random* selects a random valid move among those proposed by threads; *insert-Best* selects the move with the more positive ΔH.

In this Section, we report and discuss the results of experiments conducted by combining the parameters in Fig. IV. These analysis were performed on the Hactar cluster, which is part of the HPC@PoliTO infrastructure and it is equipped with an Intel Xeon E5-2683 v3 (12) processor operating at 2.5 GHz (3.3 GHz OC), 3.7 TB of DDR4 RAM and 349 TB of storage.

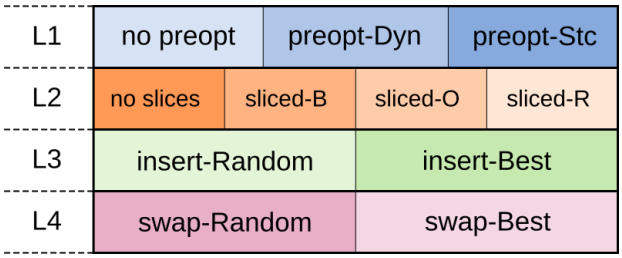

Fig. 3. Layers of parameter combinations for the optimized PIQA algorithm.

A. Dataset selection

We tested our PIQA implementation on a restricted dataset of MKP benchmark instances [15], selected for variety in size, weight distribution, value range, and difficulty. The instances utilized in our tests are characterized in Table I: the number of items and constraints are reported, together with the average *tightness ratio*. The tightness ratio α of an instance is a measure of how hard the problem is to solve. More specifically, α is defined for the k^{th} constraint in vector W as (9).

TABLE I
Characteristics of the test instances

Instance name	Items	Constraints	Average tightness
OR5x100-0.75_1	100	5	0.75
OR30x250-0.75_1	250	30	0.75
OR10x250-0.50_1	250	10	0.50
OR5x500-0.25_1	500	5	0.25
OR30x500-0.25_1	500	30	0.25
sento1	60	30	0.37
weing5	28	2	0.42
gk01	100	15	0.50
gk06	200	50	0.50
gk11	2500	100	0.50

$$\alpha_k = \frac{W_k}{\sum_{i=0}^{N} P_{k,i}} \tag{9}$$

We calculated and reported the average tightness value for the `sac94` and `GK` instances, which were not specified by either benchmark, using (9). It is worth noting that, while the tightness value across all constraints in the instance was close to uniform for the `GK` problems, the instances in `sac94` have different tightness values for each constraint: for example, the two constraints of `weing5` have $\alpha_0 = 0.53$ and $\alpha_1 = 0.30$, while the 30 constraints of `sento1` have tightness values varying between $\alpha_{\text{min}} = 0.30$ and $\alpha_{\text{max}} = 0.48$.

B. Performances of Optimized PIQA vs Standard PIQA

In Fig. IV-B we report simulation results for notable PIQA configurations. Said versions correspond to the parameter combinations described in Table II. Note that configuration S0 corresponds to the serialized, non-optimized PIQA algorithm.

TABLE II
Optimization configurations tested

Name	Preopt	Slices	Insert	Swap
S0	-	-	Random	Random
P4	-	-	Best	Best
P8	-	Balanced	Best	Best
P22	Dynamic	Balanced	Random	Best
P23	Dynamic	Balanced	Best	Random
P24	Dynamic	Balanced	Best	Best
K20	Static	-	Best	Best
K24	Static	Balanced	Best	Best

For all configurations, the number of replicas was fixed at 32; for the sliced versions, the number of slices was set to 16. The graphs in Fig. IV-B map the Mean Absolute Percent Error (MAPE) reached at each simulation time step τ. The MAPE is calculated as (10)

$$\frac{\sum_{i=0}^{N}(H_{opt} - H_i)}{N} \cdot \frac{1}{H_{opt}}, \tag{10}$$

where H_{opt} is the known optimal result for the problem instance, H_i is the result found by the algorithm at the i^{th}

attempt, and N is the number of times the algorithm was run. In our case, we ran each version of the algorithm 10 times.

Fig. IV-B.A through Fig. IV-B.D show results on problem instances from the `ORLib` benchmark library [45]. This benchmark contains procedurally generated problems with a fairly even distribution of weights and item values. In these tests, the versions with pre-optimization and sliced processing consistently obtain low error and early convergence. The serial algorithm also eventually converges to low error. Conversely, the parallel versions without pre-optimization perform slightly worse.

We find a different situation in Fig. IV-B.E and Fig. IV-B.F, showing results from instances in the `Sac94` benchmark[1]. This benchmark is a collection of real-life use cases for the MKP; it contains smaller instances with less regularity in the distribution of parameters. In this case, it appears that the pre-optimization we applied is sometimes not suitable for the problems: in the `sento1` problem, pre-optimized algorithms P22, P23 and P24 have a slow convergence, while the parallelized versions that skip pre-optimization (P4, P8) reach the exact result very quickly, beating the original serial PIQA (S0). This suggests that multiple-trial parallelism, together with the evaluation of the best item to insert/swap, remains a winning strategy for this type of problem instance.

Fig. IV-B.H and Fig. IV-B.I report results for the Glover and Kochenberger (`GK`) benchmarks. This library contains the largest problem instance available to us, `gk11` (2500 items, 100 constraints). In this sequence, the parallelized versions with pre-optimization again reach better results compared to the serial algorithm, especially as the size of the problem grows; moreover, the `swap-Random` strategy shows a disadvantage with respect to the strategies implementing `swap-Best`.

Restricted QA: Restricted Quantum Annealing (RQA) [15] is a variant of PIQA where a partial solution gets "locked" if the simulation detects that the same spin configurations are common across a given percentage (*blocking frequency*) of replicas in the system. We experimented with adding the RQA strategy to our algorithm; the results are displayed in Fig. IV-B. We found that while the addition of RQA can speed up convergence for the serial version of the algorithm, for our optimized version it was not beneficial. We can attribute this to the fact that the pre-optimization and choose-Best strategies already restrict the search space to such an extent that the contribution of the blocking frequency is superfluous. We report for reference the results of adding RQA to versions K20 and K24 of our algorithm. While the two versions of the parallel algorithm have different behavior on the two instances reported, the addition of RQA makes little difference and does not ensure a faster convergence. The results for other instances exhibit a similar trend and have been omitted for brevity.

C. Static vs dynamic optimization

We tested the performance of one of the versions of the parallel algorithm we examined in Section IV-B, P24, when substituting *Dynamic pre-optimization* with *Static*

[1]https://www.cs.cmu.edu/Groups/AI/areas/genetic/ga/test/sac/0.html

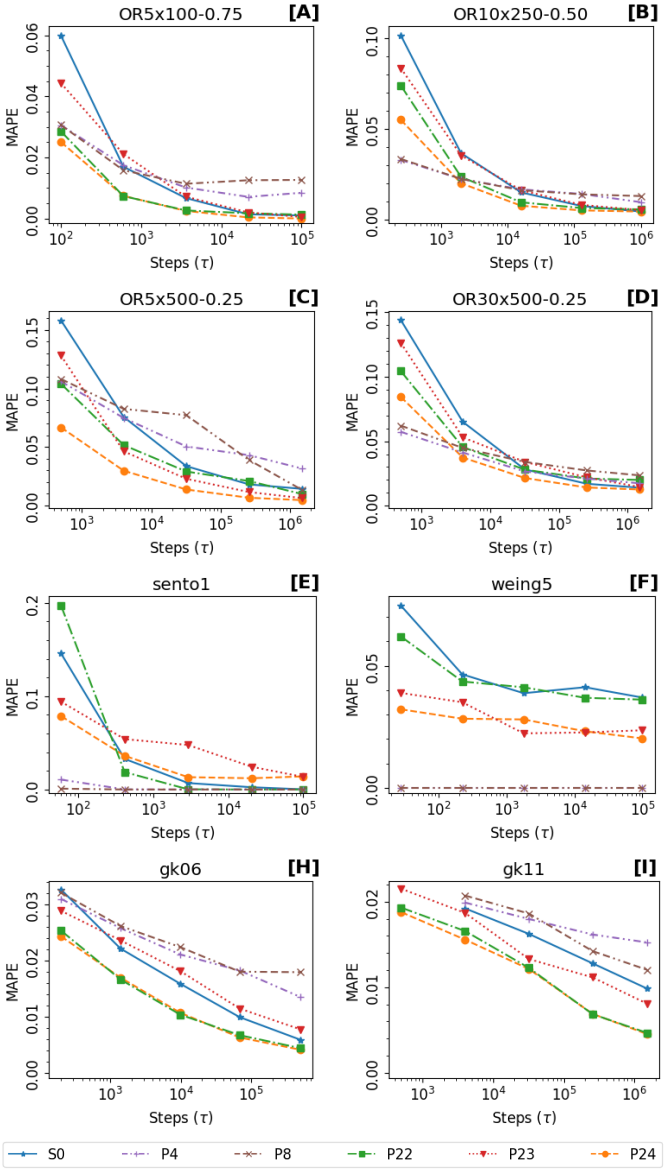

Fig. 4. Evaluation of the convergence to result of various implementations of optimized PIQA

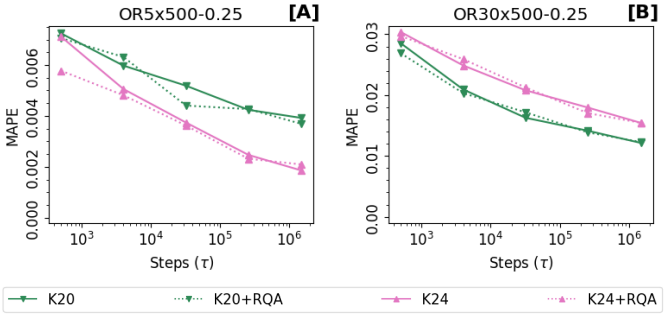

Fig. 5. Evaluation of the effect of adding RQA strategies to two versions of our optimized PIQA algorithm.

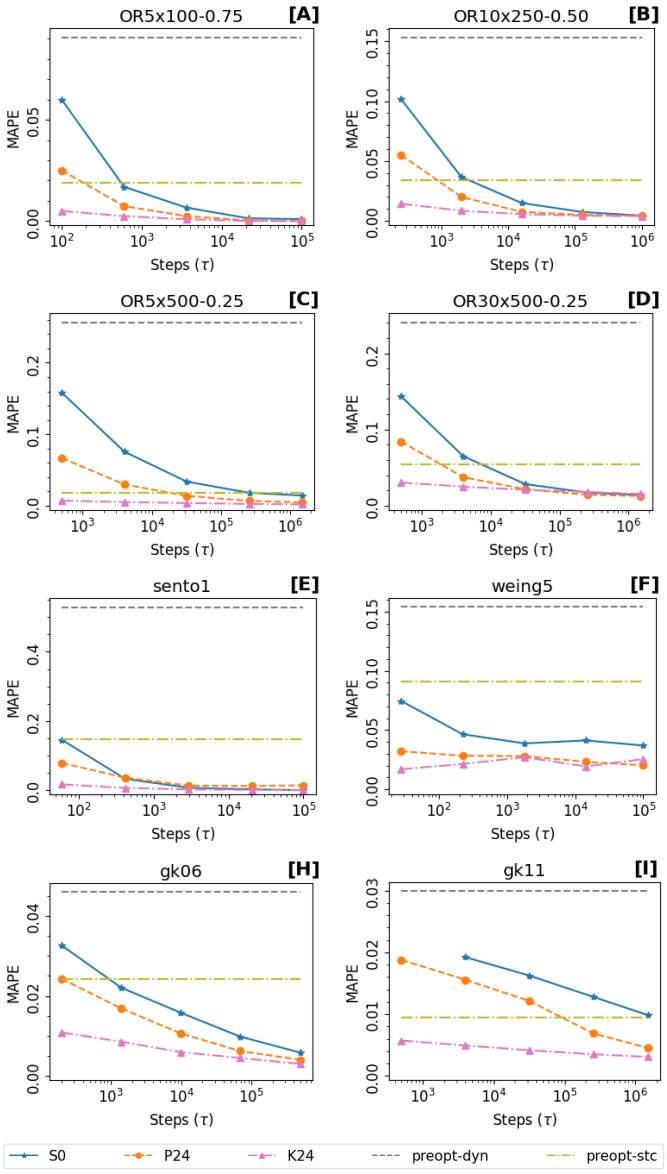

Fig. 6. Evaluation of the effects of static and dynamic pre-optimization on PIQA processing

pre-optimization. We marked the version with *Static pre-optimization* as K24. As seen in Fig. IV-C, *Static pre-optimization* proved more efficient for every problem instance in the test. The horizontal lines indicate the average starting solution, obtained by applying pre-optimization alone; we can see how *Static pre-optimization* provides a better starting point for the search, while PIQA is able to improve on the pre-optimized state more and more as the simulation times increase.

A notable result is found in Fig. IV-C.E with instance `sento1`, where *Static pre-optimization* reliably led the algorithm to the exact solution, whereas *Dynamic pre-optimization* did not.

TABLE III
OPTIMIZED PIQA PARAMETERS USED IN SECTION IV-E

C	N	threads	τ	sliced	MAPE	SD (σ)
5	100	8	100000	yes	0.2121	4.5993
5	250	8	625000	yes	0.1615	5.0416
5	500	8	2500000	yes	0.1143	4.9239
10	100	16	100000	yes	0.5628	5.9702
10	250	16	625000	yes	0.4937	6.5319
10	500	16	2500000	yes	0.2983	6.2977
30	100	32	100000	yes	0.6226	6.1022
30	250	32	625000	yes	0.7911	7.4567
30	500	16	2500000	no	0.6094	8.1490

D. Slice size

We performed experiments varying the number of slices for version K24. Fig. IV-D portrays the results over our data set.

Once again, the impact of this variation seems to be dependent on the problem instance. On instances `OR10x250-0.50`, `OR30x500-0.25` and `gk06`, for example, the number of slices has little impact on the quality of solution. However, we can observe that generally having a high (> 16) number of slices is detrimental. This may be explained by the fact that the variety of pseudo-utility distribution significantly decreases when the slices are small, therefore certain slices become less competitive within the `paradigm` and their items rarely get chosen, even though they may be part of an optimal solution. By contrast, systems with a lower number of slices ($4 \sim 8$) give an overall better performance. We deduce that this number of slices is sufficient to encourage fair choice among items, while still taking advantage of the speedup provided by parallel trials.

E. Optimized PIQA versus RQA and SA

Two additional experiments were conducted on the entire `ORLib` benchmark to characterise the performance of Optimized PIQA and to compare it with other annealing algorithms. The purpose of these investigations is to underline the advantages of the pre-optimization techniques that have been introduced in our work. The first analysis is designed to evaluate the quality of the solution derived from Optimized PIQA in relation to SA and RQA (Fig. IV-E). The second evaluation aims to verify the convergence rate of the individual optimization algorithms (Fig. IV-E).

The base version chosen for this first comparison is once again version K24, featuring *Static pre-optimization* as described in Section III-A. The only exception is the `30x500` family of benchmarks, where we turned off sliced processing, since in that particular group we obtained significantly better results without the slices (as a point of reference, the MAPE reached by the sliced version for the `30x500` family was roughly $2\times$ that reached by the non-sliced version). We also tuned the annealing time τ and the number of slices to obtain the best performance for each instance size; the final parameter assignments employed are reported in Table III. The results are compared to those of SA and (serialized) RQA, both of them run for 1 million annealing steps in compliance with the conditions reported in [15]. RQA ran with $R = 16$ replicas

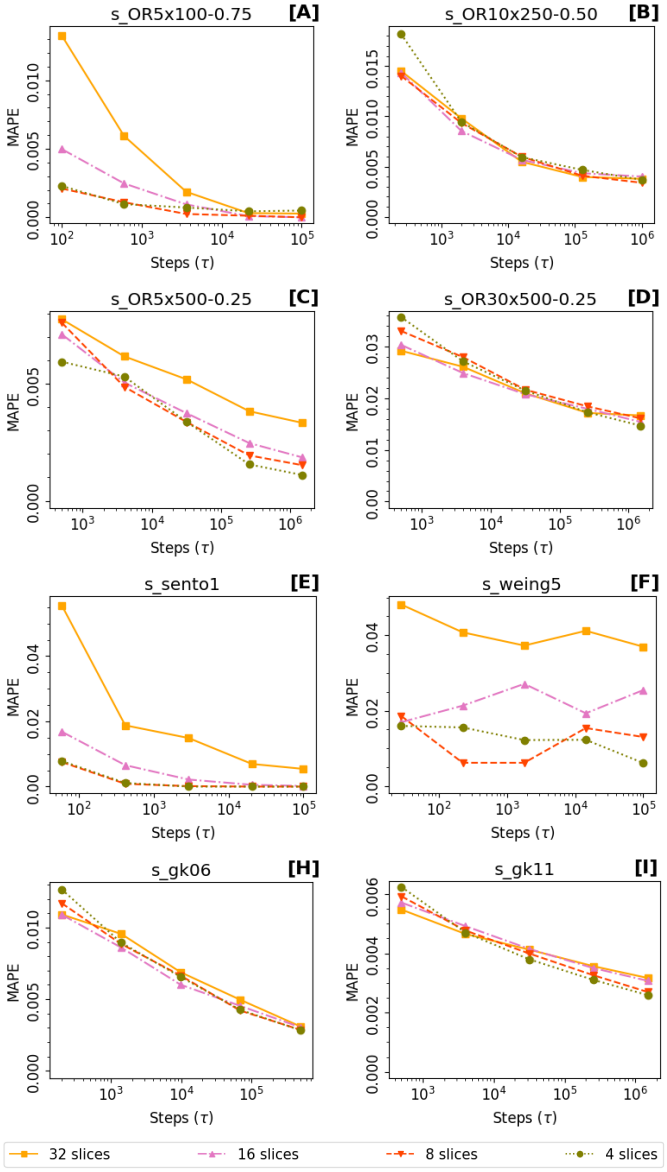

Fig. 7. Impact of the number of slices on the performance of the sliced processing version of PIQA

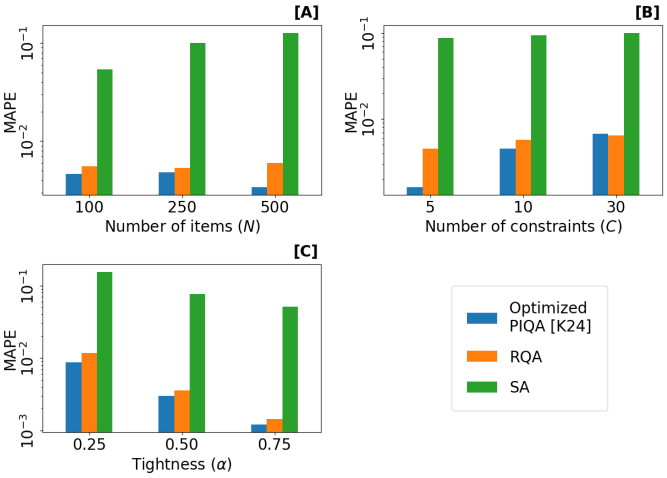

Fig. 8. Summary of the performance of competing annealing algorithms on the ORLib benchmark.

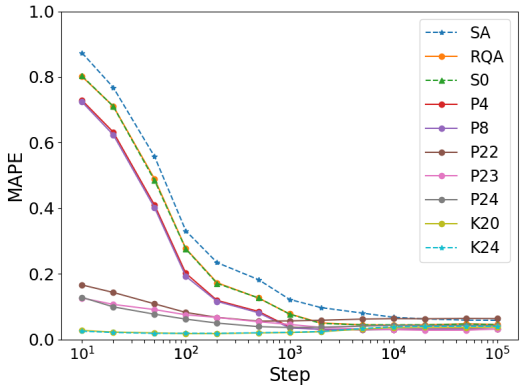

Fig. 9. Comparison of convergence dynamics for different optimization and pre-optimization techniques is performed on weing5, sento1, GK and a subset of ORLib instances

and a blocking frequency of 90%. The average error in the solutions produced by each algorithm is reported in Fig. IV-E.

Our Optimized PIQA algorithm exhibits a similar trend to RQA, while providing slightly better results. Two deviations from the trend are evident. First, when grouping the results by number of items (Fig. IV-E.A), we can observe that our optimized algorithm performs significantly better for instances of 500 items. Part of this can be attributed to the longer annealing time we devoted to these large instances compared to the SA and RQA runs; however, since for smaller instances our algorithm performed better than RQA even though it ran for a fraction of the time, we expect that some advantage will be preserved. Second, the optimized PIQA consistently achieves lower error rates for problems with fewer constraints (Fig. IV-E.B) and a slight advantage over RQA across all tightness ratios (Fig. IV-E.C). This demonstrates that the PIQA exhibits a superior capacity to address complex problems while showing a reduced MAPE in comparison to alternative algorithms. This finding suggests that the sliced processing approach may be less robust when faced with complex solution spaces that result in the system becoming trapped in degenerate states, particularly when employed in conjunction with the *choose-Best* strategies that promote rapid descent to the nearest minimum. However, other factors exist, such as the system temperature, which have the potential to be further refined to enhance the outcomes for both optimised and unoptimised versions. A comprehensive investigation of these parameters is reserved for future endeavours.

The second evaluation reported in Fig. IV-E is performed on a distinct set of instances with the objective of highlighting the convergence capacity. This analysis is performed on weing5, sento1, GK and a subset of ORLib instances. To obtain a statistically significant result, the experiment is repeated on 10 trials.

As demonstrated in Fig. IV-E, the algorithms that do not use pre-optimization start from an initial value that is significantly distant from the optimal value. This is evident in the high MAPE exhibited by the SA, RQA, S0, P4 and P8 algorithms. The simulations with a pre-optimization step, such as the P22, P23, P24, K20 and K24 configurations, result in a much lower

error than the others. This advantage is maintained up to approximately 10^3 steps. The best MAPE value is attained by the K24 configuration at step 10^2. This analysis suggests that the integration of pre-optimization strategies in conjunction with Insert/Swap strategies can yield faster convergence in comparison to conventional optimization implementations.

V. Discussion

The optimizations presented in this paper help reduce the number of steps required to reach a minimal error in the PIQA solution. Performing the algorithm on classical multi-core computers takes longer than regular PIQA, because of the frequent synchronization that must happen between threads; therefore, the ultimate outcome of this work is the identification of optimized strategies that can open the way to an improved dynamics of PIQA on alternative hardware substrates like Field Programmable Gate Arrays (FPGAs) or heterogeneous platforms. From this perspective, some remarks can be made:

- The *Static pre-optimization* we developed is the most beneficial to the final result. This type of processing can be done offline or implemented as a configurable module in hardware; it is also much more feasible than *Dynamic pre-optimization*, as it only requires to compute the pseudo-utility of items once. Said module should allow for the option to bypass pre-optimization as needed.
- *Sliced processing* appeared to give results better than, or equal to, serial implementations of PIQA. Adding sliced processing to a hardware version of PIQA would greatly reduce the memory allocated per worker and avoid using shared memory, as each worker would only need to store the weights for the items contained in its assigned slice.
- We evaluated the ability of a *choose-best* filter to determine the most suitable solution update for each annealing step. This filter could also be re-configured to allow for random choice or other selection methods instead.
- The modular nature of the proposed optimizations results in a customizable chain of independent blocks that may be easily turned on or off depending on the needs of the problem instance under investigation. Said optimizations may also be added to existing annealing accelerators to improve their performance.

VI. Conclusions

In this work, we investigated several avenues for optimization of the PIQA algorithm, using the MKP as a benchmark. We used the *multiple-trial parallelism* paradigm as a starting point, then gradually introduced other strategies.

Different pre-optimization types were initially tested. Then, we introduced new selection methods for the solution update, first using decomposition of the search space by *Sliced processing*, then proposing *choose-Best* criteria to determine the most suitable update. Finally, we compared our results to the performance of RQA (a simpler optimized variant of PIQA) and SA. We performed experiments running all combinations of optimization techniques.

We found that applying pre-optimization based on pseudo-utility can significantly speed up the convergence of PIQA, as it lets the search start from a more advantageous position in the solution space. Additionally, we found that pre-optimization based on *Static pre-optimization* — which we introduce as a new contribution in this paper — is more effective than previously proposed techniques based on *Dynamic pre-optimization*, which requires recalculation of the pseudo-utility vector at each insertion step. This means we can improve on results while saving on computation time.

Our newly developed *Sliced processing* technique was also found to help improve convergence. Versions with a moderate number of slices, between 4 and 16, were the most effective. Versions using the *choose-Best* filter to select the proposed move with the most desirable ΔH always performed better than the ones choosing any random valid move.

While these optimizations improve the overall performance of QA, there is no single configuration that is universally good for solving all problem instances. Particularly, we found that there are some benchmark instances for which pre-optimization was not as effective as for others. Statistical analysis of the dataset may be needed in order to determine the best combination of optimization parameters, along with other degrees of freedom allowed by PIQA, such as changing the annealing temperature or adding offsets and scaling factors to the system Hamiltonian. We reserve this investigation for future work.

Overall, the optimizations we proposed by leveraging dynamics such as pseudo-utility initialization, multiple-trial parallelism, sliced processing and insert/swap criteria reveal promising results in the perspective of adapting the algorithm to run on dedicated hardware. Although a direct physical emulation of quantum phenomena is not explicitly performed, we reported on their adoption and the consequent benefits, which might effectively take shape on alternative hardware substrates too. FPGAs, for instance, represent an appealing alternative to CPUs, as their configurable architecture can enable a more efficient computation of the Hamiltonian parameter variation thus allowing for faster execution of PIQA.

Acknowledgments

This Research is funded by the European Union - NextGenerationEU Project 3A-ITALY MICS (PE0000004, CUP E13C22001900001, Spoke 6) and the Fluently project with Grant Agreement No. 101058680. We acknowledge a contribution from the Italian National Recovery and Resilience Plan (NRRP), M4C2, funded by the European Union – NextGenerationEU (Project IR0000011, CUP B51E22000150006, "EBRAINS-Italy") and support from Politecnico di Torino through the Open Access initiative. Computational resources provided by hpc@polito, which is a project of Academic Computing within the Department of Control and Computer Engineering at the Politecnico di Torino (http://www.hpc.polito.it)

References

[1] D. Mehta and C. Grosan, "A collection of challenging optimization problems in science, engineering and economics," in *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2015, pp. 2697–2704.

[2] Y. Feng, G.-G. Wang, S. Deb, M. Lu, and X.-J. Zhao, "Solving 0–1 knapsack problem by a novel binary monarch butterfly optimization," *Neural Computing and Applications*, vol. 28, pp. 1619–1634, 2017.

[3] Y. Feng, G.-G. Wang, J. Dong, and L. Wang, "Opposition-based learning monarch butterfly optimization with gaussian perturbation for large-scale 0-1 knapsack problem," *Computers and Electrical Engineering*, vol. 67, pp. 454–468, 2018.

[4] Y. Feng, G.-G. Wang, W. Li, and N. Li, "Multi-strategy monarch butterfly optimization algorithm for discounted 0-1 knapsack problem," *Neural Computing and Applications*, vol. 30, no. 10, pp. 3019–3036, 2018.

[5] N. Kumaraguruparan, H. Sivaramakrishnan, and S. S. Sapatnekar, "Residential task scheduling under dynamic pricing using the multiple knapsack method," in *2012 IEEE PES Innovative Smart Grid Technologies (ISGT)*. IEEE, 2012, pp. 1–6.

[6] R. S. Camati, A. Calsavara, and L. Lima Jr, "Solving the virtual machine placement problem as a multiple multidimensional knapsack problem," *ICN 2014*, vol. 264, 2014.

[7] D. C. Vanderster, N. J. Dimopoulos, and R. J. Sobie, "Metascheduling multiple resource types using the mmkp," in *2006 7th IEEE/ACM International Conference on Grid Computing*. IEEE, 2006, pp. 231–237.

[8] P. Buayen and J. Werapun, "Efficient 0/1-multiple-knapsack problem solving by hybrid dp transformation and robust unbiased filtering," *Algorithms*, vol. 15, no. 10, p. 366, 2022.

[9] R. Mansini and M. G. Speranza, "Coral: An exact algorithm for the multidimensional knapsack problem," *INFORMS Journal on Computing*, vol. 24, no. 3, pp. 399–415, 2012.

[10] G. A. Fonseca Guerra and S. B. Furber, "Using stochastic spiking neural networks on spinnaker to solve constraint satisfaction problems," *Frontiers in neuroscience*, vol. 11, p. 714, 2017.

[11] R. Pignari, V. Fra, E. Macii, and G. Urgese, "Efficient solution validation of constraint satisfaction problems on neuromorphic hardware: the case of sudoku puzzles," *IEEE Transactions on Artificial Intelligence*, 2025.

[12] O. E. Turgut, M. S. Turgut, and E. Kırtepe, "A systematic review of the emerging metaheuristic algorithms on solving complex optimization problems," *Neural Computing and Applications*, pp. 1–104, 2023.

[13] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Computing Surveys*, vol. 35, no. 3, pp. 268–308, 2003.

[14] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse ising model," *Physical Review E*, vol. 58, no. 5, p. 5355, 1998.

[15] P. Bergé, B. Cavarec, A. Rimmel, and J. Tomasik, "Restricting the search space to boost quantum annealing performance," in *2016 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2016, pp. 3238–3245.

[16] C. Outeiral, M. Strahm, J. Shi, G. M. Morris, S. C. Benjamin, and C. M. Deane, "The prospects of quantum computing in computational molecular biology," *WIREs Computational Molecular Science*, vol. 11, no. 1, p. e1481, 2021.

[17] R. K. Nath, H. Thapliyal, and T. S. Humble, "A review of machine learning classification using quantum annealing for real-world applications," *SN Computer Science*, vol. 2, pp. 1–11, 2021.

[18] S. H. Adachi and M. P. Henderson, "Application of quantum annealing to training of deep neural networks," *arXiv preprint arXiv:1510.06356*, 2015.

[19] S. Abel, J. C. Criado, and M. Spannowsky, "Completely quantum neural networks," *Physical Review A*, vol. 106, no. 2, p. 022601, 2022.

[20] J. Marshall, D. Venturelli, I. Hen, and E. G. Rieffel, "Power of pausing: Advancing understanding of thermalization in experimental quantum annealers," *Physical Review Applied*, vol. 11, no. 4, p. 044083, 2019.

[21] A. Zaribafiyan, D. J. Marchand, and S. S. Changiz Rezaei, "Systematic and deterministic graph minor embedding for cartesian products of graphs," *Quantum Information Processing*, vol. 16, no. 5, p. 136, 2017.

[22] P. Hauke, H. G. Katzgraber, W. Lechner, H. Nishimori, and W. D. Oliver, "Perspectives of quantum annealing: Methods and implementations," *Reports on Progress in Physics*, vol. 83, no. 5, p. 054401, 2020.

[23] S. Mandra and H. G. Katzgraber, "A deceptive step towards quantum speedup detection," *Quantum Science and Technology*, vol. 3, no. 4, p. 04LT01, 2018.

[24] D. Volpe, G. A. Cirillo, M. Zamboni, and G. Turvani, "Integration of simulated quantum annealing in parallel tempering and population annealing for heterogenous-profile qubo exploration," *IEEE Access*, 2023.

[25] H. M. Waidyasooriya and M. Hariyama, "Highly-parallel fpga accelerator for simulated quantum annealing," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 4, 2019.

[26] K. Yamamoto and T. Kawahara, "Scalable fully coupled annealing processing system and multi-chip fpga implementation," *Microprocessors and Microsystems*, vol. 95, p. 104674, 2022.

[27] M. Ayodele, "Comparing the digital annealer with classical evolutionary algorithm," *arXiv preprint arXiv:2205.13586*, 2022.

[28] R. Martoňák, G. E. Santoro, and E. Tosatti, "Quantum annealing by the path-integral monte carlo method: The two-dimensional random ising model," *Physical Review B*, vol. 66, no. 9, p. 094203, 2002.

[29] E. Forno, A. Acquaviva, Y. Kobayashi, E. Macii, and G. Urgese, "A parallel hardware architecture for quantum annealing algorithm acceleration," in *2018 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*. IEEE, 2018, pp. 31–36.

[30] E. J. Kontoghiorghes, *Handbook of parallel computing and statistics*. CRC Press, 2005.

[31] A. Roy and K. G. Pillai, "Parallel simulated annealing for vlsi cell placement problem," *Montana State University*, 2009.

[32] B. Wunderlich, *Macromolecular Physics. Volume 2: Crystal Nucleation, Growth, Annealing*. New York: Academic Press, 1976.

[33] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[34] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[35] J. Brooke, D. Bitko, Rosenbaum, and G. Aeppli, "Quantum annealing of a disordered magnet," *Science*, vol. 284, no. 5415, pp. 779–781, 1999.

[36] J. W. Gibbs, *Elementary principles in statistical mechanics: developed with especial reference to the rational foundations of thermodynamics*. C. Scribner's sons, 1902.

[37] S. Laabadi, M. Naimi, H. El Amri, B. Achchab *et al.*, "The 0/1 multidimensional knapsack problem and its variants: a survey of practical models and heuristic approaches," *American Journal of Operations Research*, vol. 8, no. 05, p. 395, 2018.

[38] S. Gupta and S. Arora, "Boosting simulated annealing with fitness landscape parameters for better optimality," *Computing*, no. 14, Issue 2, pp. 107–112, 2015.

[39] X. Liu, F. Xiang, and J. Mao, "An improved method for solving the large-scale multidimensional 0–1 knapsack problem," in *2014 International Conference on Electronics and Communication Systems (ICECS)*. IEEE, 2014, pp. 1–6.

[40] L. F. Mingo López, N. Gómez Blas, and A. Arteta Albert, "Multidimensional knapsack problem optimization using a binary particle swarm model with genetic operations," *Soft Computing*, vol. 22, pp. 2567–2582, 2018.

[41] B. Haddar, M. Khemakhem, S. Hanafi, and C. Wilbaut, "A hybrid quantum particle swarm optimization for the multidimensional knapsack problem," *Engineering Applications of Artificial Intelligence*, vol. 55, pp. 1–13, 2016.

[42] W. Nakbi, I. Alaya, and W. Zouari, "A hybrid lagrangian search ant colony optimization algorithm for the multidimensional knapsack problem," *Procedia Computer Science*, vol. 60, pp. 1109–1119, 2015.

[43] X. Lai, J.-K. Hao, F. Glover, and Z. Lü, "A two-phase tabu-evolutionary algorithm for the 0–1 multidimensional knapsack problem," *Information Sciences*, vol. 436, pp. 282–301, 2018.

[44] B. de Almeida Dantas and E. N. Cáceres, "A parallel implementation to the multidimensional knapsack problem using augmented neural networks," in *2014 XL Latin American Computing Conference (CLEI)*. IEEE, 2014, pp. 1–9.

[45] P. C. Chu and J. E. Beasley, "A genetic algorithm for the multidimensional knapsack problem," *Journal of Heuristics*, vol. 4, pp. 63–86, 1998.

BIOGRAPHY

Evelina Forno is a Ph.D. student at the Department of Control and Computer Engineering (DAUIN) in Politecnico di Torino. She obtained a M.Sc. in Computer Engineering with a focus on Embedded Systems at the same institution. The main topic of her research is neuromorphic engineering, and her other interests include hardware design, edge computing and optimization heuristics.
Email address: evelina.forno@polito.it

Riccardo Pignari is a Ph.D. student at Politecnico di Torino in Department of Control and Computer Engineering (DAUIN) and member of SmartData@Polito research center. He obtained the B.Sc. in Physical Engineering and the M.Sc. in Physics Of Complex Systems both at Politecnico di Torino, (Torino, Italy). His main focus at the Electronic Design Automation (EDA) Group as a researcher is on neuromorphic and neuro-inspired computing.
Email address: riccardo.pignari@polito.it

Vittorio Fra is a Researcher and Assistant Professor at Politecnico di Torino in the Interuniversity Department of Regional and Urban Studies and Planning (DIST). He holds a B.Sc. in Physical Engineering and a M.Sc. in Nanotechnologies for ICTs, received from Politecnico di Torino where he also obtained his Ph.D. in Physics. In the Electronic Design Automation (EDA) Group, he focuses his activity on neuromorphic and neuro-inspired computing.
Email address: vittorio.fra@polito.it

Enrico Macii is a Full Professor of Computer Engineering with the Politecnico di Torino, Torino, Italy. He holds a Laurea degree in electrical engineering from the Politecnico di Torino, a Laurea degree in computer science from the Università di Torino, Turin, and a PhD degree in computer engineering from the Politecnico di Torino. His research interests are in the design of electronic digital circuits and systems, with a particular emphasis on low-power consumption aspects. He is a Fellow of the IEEE.
Email address: enrico.macii@polito.it

Gianvito Urgese is an Assistant Professor with the Politecnico di Torino (Italy) where he received a PhD in Computer and Systems Engineering in 2016. His main research interests are Neuromorphic Computing and Engineering, Parallel and Heterogeneous Architectures, AIoT application development, and Algorithm Optimisation focused on Bioinformatics and Embedded-Systems domains. He is a Senior Member of the IEEE.
Email address: gianvito.urgese@polito.it