

A GPU-Accelerated Adaptation of the PSO Algorithm for Multi-Objective Optimization applied to Artificial Neural Networks to Predict Energy Consumption

J.R.S. Iruela^a, L.G.B. Ruiz^b, D. Criado-Ramón^a, M.C. Pegalajar^a, M.I. Capel^b

^a*Department of Computer Science and Artificial Intelligence, University of Granada, Spain*

^b*Department of Software Engineering, University of Granada, Spain*

Abstract

Optimization research often confronts the challenge of developing time consuming processes. This article introduces an innovative approach that leverages the computational power of Graphics Processing Units (GPUs) to speed up that optimization process. We present an innovative adaptation of Particle Swarm Optimisation (PSO) to meet the requirements of multiobjective optimization problems. This approach aims to leverage the strengths of a multi-objective approach to perform energy consumption prediction using neural networks. By employing GPU parallel techniques, our method not only speeds up the optimisation process but also enhances the efficiency of neural network training execution. The main advantage of our approach lies in its dual ability to simultaneously optimising neural network architectures by determining the minimum number of hidden neurons and fitting the weights of the networks in order to achieve the lowest error. Preliminary results suggest a notable enhancement in prediction accuracy of forecasting electric energy consumption, as a result of optimizing the architecture and parameters of the neural network using the proposed method. This PSO adaptation stands out for its ability to address complex problems, increase efficiency and produce accurate predictions, making it a novel solution in Machine Learning heuristic methods for application in the solution of advanced prediction problems with time constraints from time series.

Keywords: GPU/CUDA, Multi-objective algorithm, Energy Consumption Prediction, Model optimisation, Artificial Neural Networks

1. Introduction

Energy sustainability is a paramount concern in contemporary society, driven by economic demands, environmental considerations, and the need for an improved quality of life. With the ever-increasing demand for energy, coupled with finite resources and the urgent need to mitigate greenhouse gas emissions, the search for efficient and sustainable energy solutions has become crucial. The predominant reliance on fossil fuels, which are both depleting and environmentally detrimental [1], underscores the pressing need for alternative energy sources and the optimisation of energy consumption.

Artificial Intelligence (AI) has experienced remarkable growth in recent years, permeating various domains, including healthcare [2], finance [3], and manufacturing [4], among others. This exponential proliferation is underpinned by AI algorithms demonstrating immense potential in automating intricate processes and facilitating rapid decision-making [5]. Within the energy sector, AI has instigated a paradigm shift in the generation, storage, and consumption of energy. The deployment of AI algorithms, particularly forecasting models, has revolutionized our ability to predict energy consumption [6, 7, 8, 9, 10, 11, 12]. These models serve as useful tools for optimizing energy production and minimizing waste. The ability to analyze large data sets and quickly generate energy consumption forecasts is essential for more efficient and sustainable energy-saving solutions. Furthermore, AI-powered energy systems have emerged as a suitable tool to improve energy efficiency. These systems can adapt and learn from historical usage patterns, enabling swift adjustments to improve energy efficiency and reduce waste. As a result, AI not only facilitates the development of predictive models but also empowers the creation of efficient algorithms and intelligent energy systems. Its ability to provide accurate predictions and enable rapid decision-making makes AI a key player in addressing the multi-faceted challenges of the energy sector.

In the realm of energy applications, AI has assumed a key role in a wide range of applications [13, 14, 15]. One of the most studied is the use of multi-objective algorithms, which are essential for addressing the diverse and often conflicting objectives in the energy domain. For instance, the innovative approach proposed by Hadjout [16], where they employ an ensemble learning method based on neural networks to predict monthly electricity consump-

36 tion. Such models, grounded in AI, are oriented to optimise multiple objec-
37 tives, including accurate predictions and computational efficiency. Similarly,
38 real-time forecasting of electricity, marginal local prices, as demonstrated in
39 Agrawal’s model [17] by using the Relevance Vector Machine, all of them
40 show the pressing need for fast optimization algorithms. In dynamic energy
41 markets, the ability to swiftly adapt to changing conditions is paramount,
42 and AI-driven approaches equipped with rapid optimisation heuristics are
43 capable of meeting this demand. The contribution [18] presents an intri-
44 cate suite of techniques to predict short-term load and price in deregulated
45 power markets. Their approach, encompassing variational mode decomposi-
46 tion, mixed data modelling, feature selection, generalised regression neural
47 networks, and the integration of metaheuristics like the gravitational search
48 algorithm, underscores the importance of AI in orchestrating complex multi-
49 objective optimisations. Highlighting the relevance of multi-objective evo-
50 lutionary approaches, [19] present a solution to optimize the energy perfor-
51 mance of buildings, by means of balancing the complex objectives of energy
52 consumption and comfort levels, which consisted of the use of a Particle
53 Swarm Optimization (PSO) method with an enhanced particle update strat-
54 egy based on adaptive perturbation. In a similar vein, Sahoo et al. [20]
55 follows the clustering approach to address the challenge of limited battery
56 life in sensors for energy efficiency. Their utilization of PSO in this context
57 stresses the advantageous role of resource-efficient optimization algorithms
58 in this area.

59 Finding the optimal hyperparameters for any Machine Learning model is
60 an extremely time-consuming task, as the search process will rely on training
61 and evaluating said model with different configurations. For simpler mod-
62 els, an exhaustive search of a set of hyperparameters may suffice to find
63 the optimal model. However, complex models, such as ANNs, have large
64 hyperparameter search spaces with complex interactions between these hy-
65 perparameters. In fact, one of the most time-consuming steps in tuning a
66 neural network architecture is finding its optimal topology (the number of
67 hidden neurons and layers). Therefore, strategies to guide this search process
68 have been frequently studied and used in the past few decades [21]. One of
69 the most common strategies used for this hyperparameter search is the use
70 of metaheuristic algorithms, such as the classic genetic algorithm or PSO.
71 These algorithms have been used not only to find optimal hyperparameters,
72 but also as an alternative to backpropagation (BP) in the training of neu-
73 ral networks. The use of metaheuristics for ANN training is motivated by

74 the fact that gradient-based training approaches can easily get stuck in lo-
75 cal optima depending on weight initialization [22]. However, it is rare to
76 see approaches that optimize the topology and weights simultaneously, since
77 metaheuristic algorithms are usually designed to work with a fixed number of
78 features per individual. Furthermore, although these algorithms guide the
79 search space for optimal configuration, just training one ANN can be com-
80 putationally expensive when there is a massive amount of data. To solve
81 this issue in an efficient manner, this paper presents a new GPU parallel
82 algorithm designed to address all these issues that incorporates the following
83 contributions to the field:

- 84 • We propose a new algorithm designed to simultaneously find the op-
85 timal topology of a neural network and train its weights in order to
86 obtain the most accurate forecast possible in the forecast while taking
87 advantage of the GPU architecture.
- 88 • We propose an adaptation of the original PSO algorithm to allow the
89 algorithm to work with variable-sized particles without prematurely
90 converging towards a fixed size topology. This is achieved through the
91 modification of the classic PSO operators and the introduction of two
92 mutation operators (structural and genetic).
- 93 • We evaluated the results of our work across six different time series
94 of energy consumption: four collected from different buildings within
95 the University of Granada alongside energy demand data from Spain
96 (REE) and Uruguay.

97 The rest of the document is structured as follows: Section 2 presents
98 a brief state-of-the-art. The proposed methodology is detailed in section
99 3. Section 4 details the parallelization and implementation of the proposed
100 multi-objective PSO algorithm on GPUs to decrease the overall algorithm
101 execution time. Section 5 introduces the conducted experiments. Section 6
102 gathers the main results. And finally, the conclusions are gathered in Section
103 7.

104 2. Related Work

105 The advances in energy consumption forecasting has been pivotal in the
106 broader context of sustainable development and energy management. Its

107 significance is highlighted by its deep impact on economic planning, envi-
108 ronmental protection, and policy making. Over the past decades, the field
109 has seen a proliferation of methodologies and tools, each aiming to predict
110 energy consumption patterns more accurately. These tools are essential for
111 optimizing resources, ensuring grid stability, and facilitating the transition
112 towards renewable energy sources.

113 Charfeddine’s work [23] on electricity consumption in Poland shows the
114 wide range of forecasting options. By employing prediction models and neu-
115 ral networks, they highlighted the importance of quantitative methods in
116 electricity demand forecasting. On a similar note, the study by [24] in the
117 Kunming metro system presented an intriguing case of energy-saving de-
118 sign. By focusing on weak current systems and leveraging algorithms such
119 as Boruta and PCA, they managed to filter and reduce feature dimensions
120 related to power consumption. This was further complemented by the use of
121 the XGBoost algorithm to establish a prediction model.

122 In a more specialized domain, locomotive traction energy consumption,
123 [25] embarked on a comparative analysis of prediction models based on ma-
124 chine learning algorithms. Their evaluation encompassed various neural net-
125 work models and algorithms, with the support of vector machine algorithm
126 emerging as a superior choice in terms of prediction accuracy and overall
127 performance.

128 The exploration of metaheuristic-based hyperparameter tuning for deep
129 learning models, as presented by Stoean et al. [26], underscores the inter-
130 section of machine learning with advanced optimization techniques in the
131 realm of energy forecasting. Specifically focusing on solar energy generation
132 predictions using LSTM and BiLSTM models, the effectiveness of enhanced
133 reptile search algorithms for optimizing model performance is highlighted.

134 Building on the concept of hyperparameter optimization, Bouktif et al.
135 [27] and Kumar et al. [28] explore the synergy between LSTM-RNN deep
136 learning models and metaheuristic algorithms for electric load forecasting.
137 These works emphasize the critical importance of optimal hyperparameter
138 tuning facilitated by metaheuristics to improve the accuracy of forecasting.

139 The advent of GPU in energy consumption prediction marks a signifi-
140 cant shift in the field. Given the increasing complexity and volume of data,
141 the need for efficient and scalable algorithms has never been more pressing.
142 [29] conducted a comprehensive review of ANNs for building energy fore-
143 casting with GPUs. Their insights into the potential of GPUs to improve
144 computational efficiency and accuracy of energy forecasting models are par-

145 ticularly noteworthy. Zhuo et al. [30] further underscore the benefits of
146 GPU-accelerated metaheuristics in solving complex optimization problems,
147 demonstrating a pathway to significantly reduce computational times while
148 maintaining high levels of accuracy.

149 PSO has also made its mark on energy consumption prediction. The
150 multi-swarm PSO algorithm proposed by [31] for static workflow scheduling
151 in cloud-fog environments outperformed canonical PSO in execution time
152 and stability. This trend of hybridizing PSO with other techniques is evident
153 in the works of [32] and [33], with the latter combining PSO with the Adap-
154 tive Neuro Fuzzy Inference System (ANFIS) for industrial energy demand
155 forecasting in Turkey.

156 The development of parallel PSO algorithms optimized for GPU/CUDA
157 environments, opens new avenues for enhancing the computational efficiency
158 of energy consumption prediction models. Innovative approaches for multi-
159 objective optimization and feature selection, as seen in the works of Liang et
160 al. [34], Lv et al. [35], and Yilmaz and Sen [36], introduce robust method-
161 ologies to address the multifaceted challenges inherent in energy forecasting.

162 Similarly, the application of multi-objective techniques in energy con-
163 sumption prediction has seen notable advancements. Jesus et al. [37] in-
164 troduced a hybrid Particle Swarm Optimisation-Neural Network algorithm,
165 while [38] presented a GPU-based parallel implementation of NGSa-II. Both
166 works highlight the potential of these algorithms in enhancing the efficiency
167 and accuracy of energy consumption predictions. Agarwal et al.'s exploration
168 of multi-objective PSO with guided exploration for multimodal problems [39]
169 further contributes to the sophistication and effectiveness of predictive mod-
170 els, ultimately supporting a more sustainable and economically viable energy
171 management landscape.

172 In short, rapid predictive solutions and optimal models are crucial when
173 predicting energy consumption in buildings. These approaches enable accu-
174 rate forecasting, efficient resource allocation, and effective energy manage-
175 ment. In response to these necessities, the present study provides a GPU
176 parallelization of an adapted PSO for MO to optimize ANN, offering faster
177 and optimal models. This GPU-based solution harnesses the power of paral-
178 lel computing to improve the performance of energy consumption prediction
179 models. To take advantage of this power, we will follow a strategy where
180 each of the particles will update its position and velocity and gets its fitness
181 evaluated in parallel by each GPU thread and the different functions will be
182 developed in GPU kernels.

183 **3. Methodology**

184 In this research, we introduce a method for multi-objective optimisation
 185 that builds upon the framework of the PSO algorithm. While PSO is a
 186 well-established bio-inspired stochastic optimization algorithm, it tradition-
 187 ally operates as a single-objective optimisation method, seeking to find the
 188 optimal solution for a single objective function. To adapt PSO for multi-
 189 objective optimisation, we have incorporated the Pareto and the dominance
 190 operator. These modifications enable our method to efficiently handle multiple
 191 conflicting objectives simultaneously, in our case, error and number of
 192 neurons. The general scheme of the modified version of the PSO can be seen
 193 in figure 1.

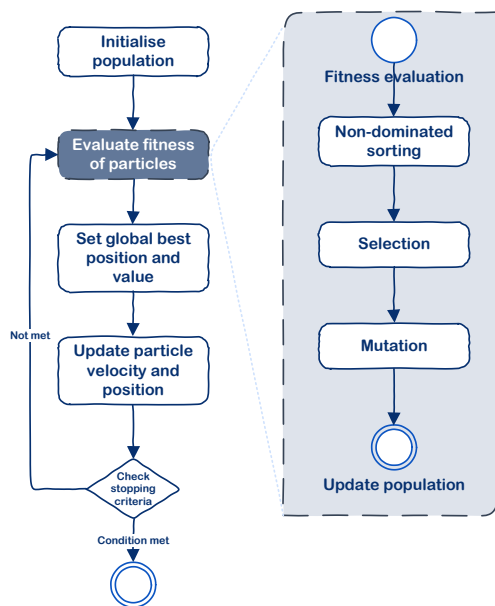


Figure 1: Scheme of the proposed algorithm.

194 *3.1. Particle Swarm Optimisation*

195 The general scheme of PSO is represented in the left side of figure 1. This
 196 technique is a computational method that optimizes a problem by iteratively
 197 trying to improve a candidate solution with regard to a given measure of

198 quality such as the MSE. It mimics the social behavior of birds and other
 199 creatures in a swarm [40]. The key phases of this algorithm are mainly four
 200 as can be seen in the figure. Initialization, where a population of particles is
 201 randomly initialized within the solution space. Each particle represents a po-
 202 tential solution and is assigned an initial position and velocity. After initial-
 203 ization, the fitness of each particle is evaluated using the objective functions
 204 (error and network size minimization) defined for the optimisation problem.
 205 These functions quantify how well a particular solution performs. The next
 206 phase involves updating the velocity and position of each particle based on
 207 its previous values, its distance to its local best solution, and its proximity
 208 to the best solution. The PSO algorithm continues to update particles' posi-
 209 tions and velocities iteratively until a predefined termination criterion is met.
 210 Normally this criterion will be defined by a maximum number of iterations
 211 or when the algorithm stalls and does not improve the solutions.

212 The PSO algorithm can be mathematically represented as follows. Each
 213 particle is represented by a position vector, X_i in the search space. This
 214 vector contains values corresponding to the decision variables of the problem
 215 you're trying to optimize. Each particle also has a velocity vector V_i that
 216 determines the direction and magnitude of its movement in the search space.
 217 The fitness function $f(X_i)$ evaluates the quality of a solution represented
 218 by a particle's position. This function needs to be minimized or maximized
 219 depending on the optimization goal. Each particle follows its own best posi-
 220 tion $pbest_i$ encountered so far based on $f(X_i)$. This represents the particle's
 221 individual learning. Besides, there is a single global best position $gbest$ that
 222 stands for the best solution found by any particle in the entire swarm. Thus,
 223 the velocity is updated as indicated by equation 1.

$$V_i(t+1) = wV_i(t) + c_1 \text{rand}() (pbest_i(t) - X_i(t)) + c_2 \text{rand}() (gbest(t) - X_i(t)) \quad (1)$$

224 Where t is the iteration number, w is the inertia weight, and c_1 and c_2 are
 225 the cognitive and social learning rates, respectively. Likewise, the position is
 226 computed as described in equation 2.

$$X_i(t + 1) = X_i(t) + V_i(t + 1) \quad (2)$$

227 In the context of this study, PSO is utilized as the basis of the optimiza-
 228 tion method to enhance the weight adjustments of ANNs and minimize the
 229 number of hidden neurons. The efficiency of the algorithm in finding high-
 230 quality solutions in a timely manner makes it a valuable addition to the suite

231 of techniques used in other studies [41, 42]. The parallel nature of PSO also
232 lends itself to implementation within a GPU framework, contributing to the
233 reduction in computational time and the quality of solutions found. However,
234 classic PSO operators are not designed to work with a different number of
235 weights per particle, a necessary prerequisite to optimizing different topolo-
236 gies. Thus, in this paper, we propose an adaptation of PSO designed to work
237 in a multi-objective and parallel way using GPU resources. This adapta-
238 tion also includes variations of the classic PSO operators capable of working
239 particles with variable sizes, thereby enabling the use of this algorithm to
240 optimize simultaneously different neural network topologies. The detailed
241 implementation and integration of PSO within the proposed framework are
242 discussed in the following sections.

243 *3.2. Adapted PSO for multi-objective problems*

244 As discussed in the previous section, since the Standard PSO is de-
245 signed to optimize a single objective function, we have designed a new multi-
246 objective adaptation of PSO to simultaneously find the optimal topology and
247 train its weights. Each particle contains a position vector (the weights for its
248 neural network) and its number of neurons. The PSO operators have been
249 adapted to take into account this variable number of neurons (explained in
250 detail in Section 3.4). Additionally, several operators were added to adapt
251 the algorithm for multi-objective purposes (non-dominating sorting and se-
252 lection) and ensure diversity (mutations). The scheme of the proposed algo-
253 rithm can be found in figure 1.

254 In order to evaluate the fitness of each particle, we need to categorise the
255 particles into different levels (or fronts) based on their dominance relation-
256 ships with respect to multiple objectives. Solutions that are not dominated
257 by any other solutions belong to the first front (also known as the Pareto
258 front) and represent the set of non-dominated or optimal solutions. We also
259 employed the crowding distance so as to enhance the diversity of solutions
260 along the Pareto front, in doing so, we ensure a well-distributed set of non-
261 dominated solutions.

262 Following the non-dominated sorting procedure, the next phase in our
263 proposed optimisation process is the selection operator. This step plays a
264 decisive role in determining which parent solution [43] a particular particle
265 will follow as a local leader through the solution space. The selection op-
266 erator’s objective is to choose the local best solution, one that represents

267 a promising trade-off among the objectives being optimized. This selected
268 parent guides the particle in its exploration for better solutions.

269 Then, we introduce a mutation operator to modify particles within the
270 swarm. In doing so, the algorithm’s ability to explore the solution space
271 thoroughly is enhanced. By introducing controlled variations to certain par-
272 ticles, the mutation operator injects diversity into the swarm, thus preventing
273 premature convergence to suboptimal solutions.

274 In the last stage of our method, we calculate the errors for each particle
275 within our swarm. These errors serve as a quantitative measure of the quality
276 of the solutions generated by our algorithm, providing insights into how well
277 each solution aligns with the optimisation objectives. Then, the particles are
278 updated with the corresponding target variables, i.e., the error values and
279 the number of neurons in the neural network architecture.

280 *3.3. Adapted PSO for MO to optimize ANN*

281 In the present study, multi-objective techniques, which allow us to mea-
282 sure the goodness of each solution, together with PSO are employed to opti-
283 mize both the topology and the weights of the ANNs. The integration of these
284 tools facilitates the simultaneous achievement of the lowest possible error and
285 the optimal number of neurons, enhancing the overall performance and ef-
286 ficiency of the ANNs. This multi-objective PSO approach allows solutions
287 to be chosen fairly and randomly. This approach improves the exploration
288 of the search space, mitigating the trend to converge to local optima and
289 strengthening the pursuit of global optimal solutions.

290 In subsequent sections, a comprehensive exposition and adaptation of
291 the components of the PSO algorithms will be presented. This detailed
292 exploration aims to elucidate the methodology employed to attain an optimal
293 ANN, so as to enhance the prediction of energy consumption in buildings.

294 *3.4. Algorithm Components*

295 This section delves into each of the components developed of the pro-
296 posed PSO adaptation as mentioned in previous sections. An overview of
297 such components can be seen in figure 2

298

299 **Coding of Particles:** Each particle is clearly characterized by two parts
300 (see purple branch of figure 2): the first represents the position of the particle
301 and the second represents the velocity of the particle. In order to represent

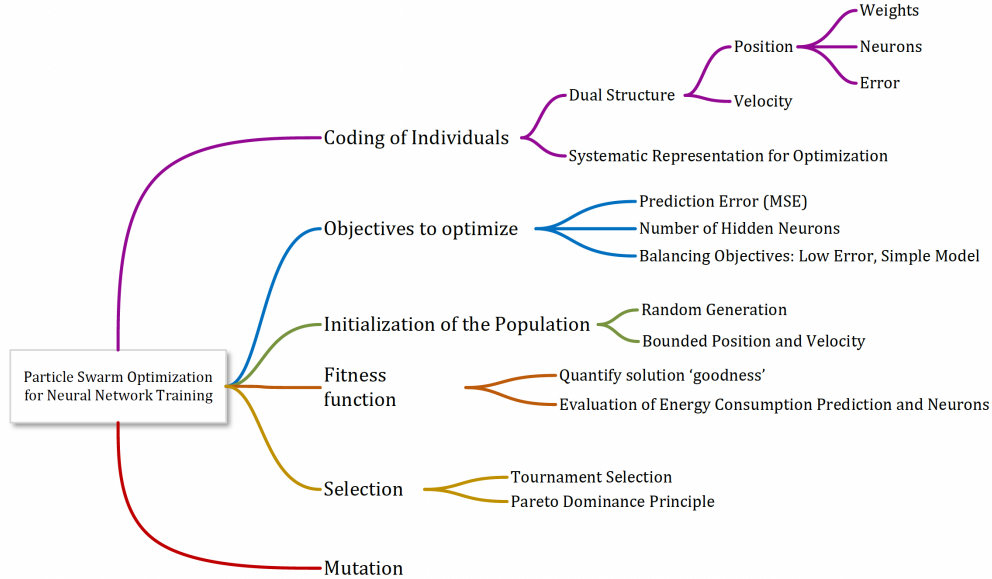


Figure 2: Components of the proposed PSO adaptation

302 the position of the particle, a vector is used to store the weights of the net-
 303 work, and the last 2 values of this vector store the number of hidden neurons
 304 together with the error. This structure is crucial for the simultaneous opti-
 305 mization of both the weights of the neural network and its architecture.

306

307 **Objectives to Optimize:** To identify the optimal neural network for this
 308 study, we have established two fundamental objectives that guide our opti-
 309 mization objective (blue branch of figure 2).

310

311 **Prediction Error:** One metric for evaluating the proficiency of an ANN
 312 is the prediction error. Intuitively, the lower the prediction error, the closer
 313 the ANN is to reaching its ideal configuration. Our methodology aims to
 314 minimize this error in the training dataset. The Mean Squared Error (MSE)
 315 serves as our error quantification metric, represented by equation 3. Here
 316 T signifies the total number of instances contained in the training set, $Y(t)$
 317 is the actual output, and $O(t)$ denotes the output calculated by the neural
 318 network at time t .

319

$$Min \{f_1(s)\} = min \left\{ \frac{1}{T} \sum_{t=1}^T (Y(t) - O(t))^2 \right\} \quad (3)$$

320

321 **Number of Hidden Neurons:** This number serves to evaluate the net-
 322 work complexity, often measured by the number of hidden neurons, as our
 323 second objective. By examining this, we aim to compare networks of varying
 324 complexity to find those that maintain performance efficacy but are lighter
 325 in terms of hidden neurons. The formal definition of this objective, where
 326 $h(s)$ encapsulates the number of hidden neurons for the designated network
 327 s , is shown in equation 4.

$$Min \{f_2(s)\} = min \{h(s)\} \quad (4)$$

328

The crux of our approach is to find models that adequately/skillfully bal-
 329 ance these objectives: aiming at the lowest prediction error whilst getting
 330 the simplest model structure. It is imperative to acknowledge that while a
 331 sophisticated model might intuitively offer superior results, finding a simpler
 332 but equivalently proficient model is a non-trivial challenge.

333

334 **Swarm initialization:** Each of the particles that are part of the swarm rep-
 335 represents a conceivable, non-optimal, solution to the problem. Each solution
 336 characterizes a distinct ANN (green branch of figure 2). The data-values
 337 (neurons, weights...) included in each of the particles are generated ran-
 338 domly, ensuring a diverse exploration of the solution space. In our problem,
 339 each particle is assigned an initial position, which will be a vector containing
 340 the number of neurons, error, and weights, and velocity bounded within a
 341 pre-determined minimum and maximum. These limits are detailed in the
 342 next sections.

343

344 **Fitness Function:** The fitness function, often referred to as the objective
 345 function, quantifies how 'fit' or 'good' a solution is (orange branch of figure
 346 2). For our specific problem, the fitness function is tailored to evaluate the
 347 accuracy of the ANN in predicting energy consumption and the number of
 348 neurons. The lower the prediction error and the number of neurons, the bet-
 349 ter the fitness of the particle representing that specific individual.

350

351 **Selection:** This technique helps us to determine which particles (or solu-
 352 tions) with superior fitness should be kept for the next generation (yellow
 353 branch of figure 2). There exist numerous strategies to execute this selection,
 354 including tournament selection, random selection, roulette selection, etc. .
 355 For this study, we have employed the tournament selection method. In this
 356 approach, two parents are randomly chosen from the entire population and
 357 in opposition against each other, with the comparison based on their fitness
 358 values. The selection process in our algorithm is based on the Pareto domi-
 359 nance principle. Solutions are selected based on their ability to dominate
 360 others (a solution dominates another when it is better in at least one objec-
 361 tive without being worse in the others) in terms of both objectives: error rate
 362 and model complexity. This ensures that the retained solutions are optimal
 363 in terms of one or both objectives.

364
 365 **Mutation:** The mutation operator plays a critical role in maintaining the
 366 diversity within the swarm. It introduces variations by making either sub-
 367 tle or significant alterations to the attributes of existing particles. A visual
 368 representation of the mutation process can be found in Figure 3. Among the
 369 variations, the structural mutation is important because it alters the net-
 370 work’s topology, specifically the number of neurons. Such changes can be
 371 profound, leading to the expansion or contraction of the chromosome’s size
 372 and therefore to the capability of generating variability. In addition, genetic
 373 mutation is used, which modifies one or more genes corresponding to the
 374 weights of the network. This dynamism is essential as it enhances the explo-
 375 ration capabilities within the search space.

376

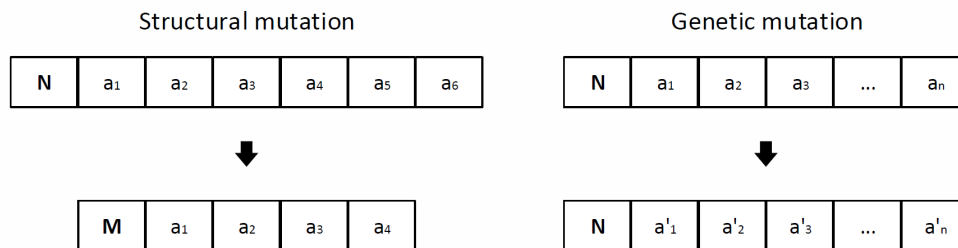


Figure 3: Structural and genetic mutation.

377 **PSO MO description:**

```

1 Begin;
2 particles ← InitializeParticles(numberOfParticles);
3 iteration ← 0;
4 while iteration < maxIterations do
5     fitness ← EvaluateFitness(particle);
6     UpdatePersonalBest(particle, fitness);
7     Front-order(particle) and crowding distant(particle) ;
8     UpdateGlobalBest(particles);
9     SelectionOperator(particles);
10    Mutate(selectedParticles);
11    UpdateVelocity(particle);
12    UpdatePosition(particle);
13    iteration ← iteration + 1;
14
    end
15 optimalSolution ← ExtractOptimalSolution(particles);
16 Return optimalSolution;
17 End;
```

Algorithm 1: General algorithm of the proposed PSO adaptation for multi-objective purposes.

378 In the developed algorithm, the process begins with the initialization of a
379 set of particles, assigning each particle random positions and velocities. Sub-
380 sequently, the fitness of each particle is evaluated using the predefined fitness
381 function. Building on the fitness evaluations, an ordering of fronts is exe-
382 cuted, and the crowding distance is incorporated. With this framework, the
383 global best solutions are updated. The selection operator is then employed
384 to keep the dominant solutions, leading to updating the local best solutions.
385 To propitiate diversity within the swarm, the mutation operator is invoked.
386 The position and velocity of each particle are re-calculated, drawing insights
387 from their local best, global best, and current velocity. This iterative process,
388 referring to steps two through seven, is continued for a number of cycles or
389 until the algorithm converges to a solution.

390 In doing so, we ensure that, over time, the particles converge towards
391 optimal or near-optimal solutions that satisfy both objectives of our opti-
392 mization problem. With the combination of these components, our PSO

393 algorithm effectively addresses the challenges of multi-objective optimization
394 in the realm of energy consumption prediction using ANNs.

395 **4. Parallel Adapted PSO for MO to optimize ANN**

396 We have already explained the basic idea of the PSO algorithm we used
397 and some important concepts. Now, we will explain how we parallelised
398 the PSO algorithm. It is important to remember that designing algorithms
399 for regular CPUs is different from designing them for GPUs. How memory
400 works and how tasks are divided up are different between these two types
401 of processors. But the overall goal of the algorithm stays the same on both
402 CPUs and GPUs.

403 *4.1. Memory management*

404 To take advantage of the GPU's processing capabilities, data is moved
405 between the CPU's main memory and the GPU's memory using special func-
406 tions. These functions ensure efficient processing on the GPU. They allow
407 us to transfer large datasets, such as particle locations, speeds, performance
408 scores, and random number generators, to the GPU's memory for much faster
409 calculations. Once processed, the results can be easily brought back to the
410 CPU's memory for further use.

411 *4.2. Kernel functions*

412 CUDA code is structured into kernels, parallel functions in which the
413 programmer specifies the code to be executed by one thread. In our design,
414 each kernel corresponds to one of the operators used. Therefore, the com-
415 putations associated with each operator are done in parallel for all particles.
416 Once a kernel starts to be processed in the main stream, the GPU will wait
417 until all computation related to the kernel has finished, which ensures that
418 no race conditions occur.

419 In the realm of CUDA, a leading platform for GPU-accelerated appli-
420 cations, this block is conceptualized as a 'thread block'. A thread block,
421 particularly with CUDA Toolkit 10, encompasses up to 1024 threads, i.e., 32
422 Warps or thread groups that are run at once by SMs. Threads within a warp
423 are executed on the same SM, and crucially, can communicate seamlessly
424 through shared memory or atomic operations.

425 To harness the full potential of the highly parallel architecture of GPUs,
426 our kernels usually associate the computations related to a particle with a

427 block and the computations related to a neuron within that particle to a
428 thread. Thus, all operations are done in parallel at their finest granularity
429 and memory accesses are coalesced, ensuring optimal times when reading or
430 writing data. This distribution also allows the threads within each block to
431 cooperate faster, as they share the same registers and L1 cache, which can be
432 directly managed by the programmer through the use of “shared” memory,
433 a key factor in improving the performance in certain operations that require
434 cooperation, such as computing the final output of the neural network (that
435 relies on the weighted sum of all threads within the same block).

436 The following subsections provide the implementation details of the ker-
437 nels used in the proposed algorithm.

438 *4.2.1. Generation of particles (GP)*

439 On the GPU’s main memory, i.e., global memory, we store information
440 about each potential particle in a two-dimensional array. Each row in this
441 array represents a single particle, and its elements contain details like the
442 weights and the number of neurons in that specific architecture. An addi-
443 tional one-dimensional array also resides on the GPU’s memory and stores the
444 number of neurons for each particle. To initialize these particles, we launch
445 a grid of thread blocks on the GPU. Each block works on a specific group
446 of particles. The threads within each block initialize the particle weights fol-
447 lowing a uniform distribution in $[-1, 1]$. Each thread uses the Xoroshiro [44]
448 function to generate random values for the corresponding weight elements
449 within that particle. A unique identifier for each thread helps distribute the
450 initialization tasks among the threads within a block. To prevent conflicts
451 when multiple threads try to update the same weight element simultaneously,
452 the algorithm uses atomic operations, ensuring safe updates to the particle’s
453 weight values.

454 *4.2.2. Non dominated sorting (NDS)*

455 The second kernel focuses on non-dominated sorting, a process that iden-
456 tifies the most promising solutions in the swarm. Each thread within this
457 kernel is tasked with evaluating how dominant each particle is compared to
458 others. To speed up this comparison, the particles’ fitness scores are loaded
459 into a shared memory area, allowing for faster access. After each particle’s
460 dominance is determined, a single thread takes charge of assembling these
461 particles into different fronts. Figure 4 illustrates the structure used for orga-
462 nizing particles. The first array in this figure stores the indexes of particles

463 grouped by their fronts, while the second array indicates the size of each
 464 front. To run this kernel efficiently, we assign as many threads as there are
 465 particles in the swarm to each thread block. Finally, this kernel returns a
 466 classification similar to the one depicted in Figure 5. This figure showcases
 the Pareto front (or 0-front), where each point represents a particle.

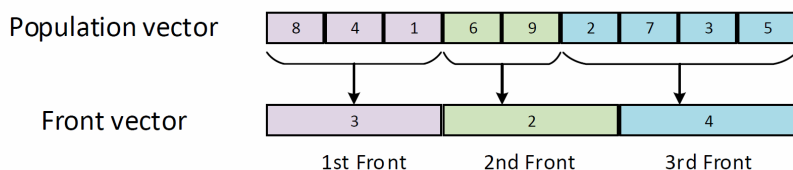


Figure 4: Data structure for fronts.

467

4.2.3. Selection operator (SO)

468 The SO (Selection Operator) kernel employs a tournament-style compe-
 469 tition to identify the single best particle (global position) from the entire
 470 swarm. Each thread within this kernel manages a head-to-head competition
 471 between two particles. The winner of this competition is then considered
 472 the best overall solution found so far. To speed up comparisons, crowding
 473 distance information for the competing particles is loaded into a shared mem-
 474 ory area for faster access. It is important to note that only particles on the
 475 Pareto front participate in this tournament. To determine the winner, their
 476 crowding distances are compared. The particle with the shorter crowding
 477 distance, indicating less competition in its surrounding area, emerges victo-
 478 rious. Finally, a probability factor is introduced to add a layer of randomness
 479 and further refine the selection of the ultimate winner.
 480

4.2.4. Mutation operators (MuO)

481 The MuO kernel handles the mutation process for the particles in the
 482 swarm. Each thread is assigned a specific particle and is responsible for
 483 introducing mutations. Both the position and velocity information for all
 484 particles are stored in the GPU's global memory for easy access during this
 485 process. A random chance determines whether a particular particle will
 486 be mutated. If selected, a vector containing random numbers is used to
 487



Figure 5: The Pareto optimal front.

488 define the new size of the particle, i.e., number of neurons, and update the
 489 corresponding weights within the particle’s genes.

490 4.2.5. Update velocity (*UV*)

491 The UV kernel plays a crucial role in the proposed method. This kernel
 492 focuses on updating the velocity of each particle in the swarm. Each thread
 493 within the UV kernel works concurrently to update a single particle’s veloc-
 494 ity. To achieve this, the kernel retrieves the velocity and position information
 495 for all particles from the GPU’s global memory. The kernel then considers
 496 the particle’s personal best position, *pbest* and the overall best position dis-
 497 covered so far, global best, to calculate the updated velocity for each particle.
 498 Importantly, each thread block handles a specific group of particles, and the
 499 threads within each block collaborate to calculate the velocity updates for
 500 their assigned particles using the provided formulas.

501 4.2.6. Update position (*UP*)

502 The UP (Update Position) kernel is responsible for the crucial task of
 503 updating the location, or position of each particle within the swarm. To
 504 ensure maximum efficiency, this kernel gets both the position and velocity
 505 information for all particles directly from the GPU’s global memory. Similar
 506 to the velocity update process, each thread block is assigned a specific group
 507 of particles to handle. Each thread within a block then calculates the new
 508 position for its assigned particle by simply adding the particle’s updated
 509 velocity to its current position. This process effectively moves each particle

510 in the swarm based on its momentum and the influence of the best positions
511 it has encountered.

512 5. Experiments

513 This section describes the experiments we conducted to explore and over-
514 come the task of predicting EC. We begin by outlining the specific hardware
515 and software used. Next, we describe the dataset employed for training and
516 evaluation, along with any cleaning or preparation steps performed on the
517 data. To assess the efficiency gains achieved, we then present a speedup anal-
518 ysis, highlighting the improvements in processing speed. Finally, we define
519 the performance metrics used to measure the success of our approach. An
520 overview of the designed general scheme to address our problem can be seen
521 in figure 6.

522 5.1. Experimental setup

523 To ensure the replicability and reliability of our experiments, we now
524 detail our computational environment. All computations were performed
525 on an NVIDIA GeForce GTX 1080 graphics card with 2560 cores and a
526 maximum thread capacity of 1024 per block (refer to Figure 7 for the memory
527 hierarchy). Python was chosen as the programming language due to its
528 versatility and extensive compatibility with machine learning libraries and
529 algorithms. For efficient GPU execution of the developed models, the Numba
530 compiler, supported by CUDA, was utilized.

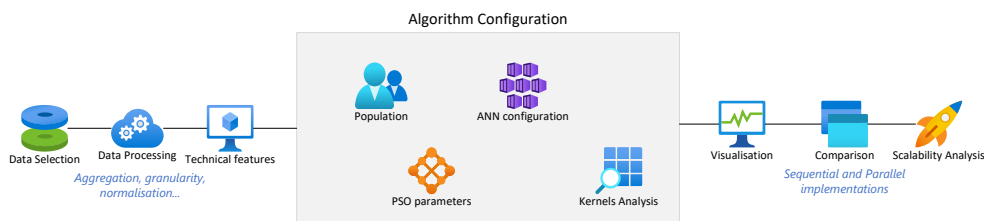


Figure 6: Overview of the general scheme for the conducted experimentation

531 5.2. Data description and preprocessing

532 In our experiments we used data collected from a variety of buildings at
533 the University of Granada. These buildings were specifically chosen because

534 their energy consumption patterns closely resemble those of other university
 535 buildings, ensuring the generalizability of our findings. The data encom-
 536 passes four years of hourly readings from each of the four facilities, resulting
 537 in a dataset of approximately 35.000 data points per building. To broaden
 538 the scope of our analysis and ensure our models can handle diverse time-
 539 series scenarios, we additionally incorporated data from the Spanish Elec-
 540 tricity Network (REE). The REE dataset has daily electricity consumption
 541 data spanning over a decade, ranging from January 1, 2009, to December 31,
 542 2019. Besides, to explore performance in a different geographical context,
 543 we included data from buildings located in Uruguay. This dataset features
 544 hourly energy consumption readings from January 1, 2007, to January 31,
 545 2019. By incorporating data with a different geographical location, we es-
 546 tablish a multifaceted approach that provides a robust picture to evaluate
 547 the effectiveness of our models. Finally, once the data is collected, it is pre-
 548 processed and the consumption aggregated on a daily basis. To enhance the
 549 predictive capacity of our models, a 24-hour time window is employed. Ad-
 550 ditionally, we normalize the data to ensure uniformity in attribute scaling
 551 and partition it into training (70%) and testing (30%) subsets.

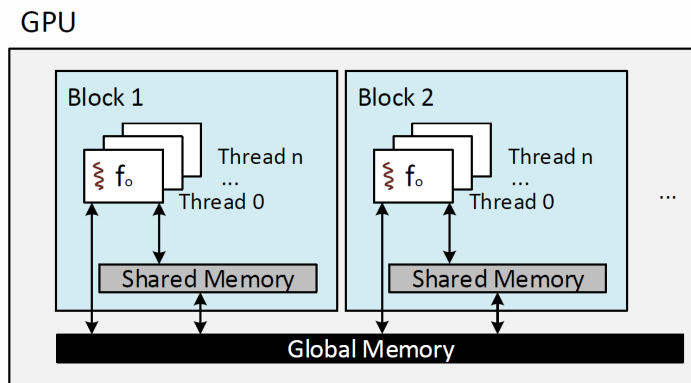


Figure 7: Memory management and design for each kernel f_i leveraging shared memory.

5.3. Algorithm's settings

552 We then proceed to describe the algorithm configuration, broken down
 553 into several clear steps. Each stage tackles specific aspects such as how
 554

555 many elements are in each group (population sizes), how the algorithm makes
556 choices within those groups (multi-objective algorithm parameters), how the
557 algorithm learns according to the cognitive and social coefficients (PSO pa-
558 rameters), how artificial neural networks are structured (ANN configura-
559 tions), and finally, a deep dive into the individual building blocks that make
560 up the entire algorithm (individual kernels). To assess the effectiveness of our
561 multi-objective algorithm, we used a technique called Pareto analysis. This
562 analysis method helps us visualize the different solutions generated by the
563 algorithm. In our case, we are particularly interested in understanding the
564 trade-off between two key aspects: the number of neurons within the ANN
565 and the error achieved. By examining these solutions visually using Pareto
566 analysis, we can identify configurations that achieve a balance between model
567 complexity (number of neurons) and prediction accuracy. This allows us to
568 select the most suitable configuration for our specific needs. Due to space
569 limitations, we cannot present all the experiments conducted to fine-tune
570 our algorithm. However, we will highlight some key parameters explored
571 reported in Table 1. The number of epochs, which determines the number
572 of times the algorithm iterates through the training data, was tested with
573 values of 10, 20, and 30. Eventually, 10 epochs yielded the best performance.
574 Similarly, we evaluated the impact of the particle swarm size, ranging from
575 50 to 300 particles. After experimentation, 200 particles proved to be the
576 most effective configuration.

577 We also explored the optimal structure for the ANN within the algorithm.
578 The minimum number of neurons in the hidden layer was tested between 2
579 and 6, with 4 neurons showing the best results. On the other hand, the
580 maximum number of hidden neurons was tested within a range of 26 to 36,
581 and 32 neurons provided the most favorable outcome. Furthermore, we op-
582 timized various hyperparameters related to mutation within the algorithm.
583 The mutation rate, which controls the likelihood of introducing changes dur-
584 ing the optimization process, was tested with values of 0.00, 0.1, and 0.2. We
585 also evaluated different structural mutation rates (0.005, 0.2, and 0.3) and
586 genetic mutation rates (0.005, 0.2, and 0.3), determining that values of 0.2
587 and 0.01, respectively, were most effective.

588 In addition to these parameters, we experimented with a range of values
589 for the cognitive coefficient ($c1$) and the social coefficient ($c2$), which are crit-
590 ical for promoting effective information sharing among particles. Specifically,
591 $c1$ was tested within a range of 1.5 to 3.0, eventually settling on 2.8 as the
592 optimal value. Similarly, $c2$ was evaluated across a spectrum from 1.0 to 2.0,

593 with 1.3 identified as the most effective setting. The Factor, controlling the
 594 speed updates of particles, was also scrutinized with values tested from 5 to
 595 15, leading us to the optimal setting of 10, which allows for speed updates
 596 within a range of -1 to 1.

597 Furthermore, we explored the inertia weight (w), a pivotal factor influ-
 598 encing the balance between exploration and exploitation, by testing it be-
 599 tween 0.5 and 1.0. The chosen setting of 0.9 emerged as the most conducive
 600 to achieving a harmonious balance. Additionally, the maximum velocity
 601 (V_{max}) parameter, which ensures bounded particle movements, was eval-
 602 uated with values ranging from 20 to 40, with 30 proving to be the most
 suitable for our needs.

Parameters	
Tournament's winner	0.9
Structural mutation	0.2
Genetic mutation	0.1
c1	2.8
c2	1.3
factor	10
minimum speed	-1
maximum speed	1
w	0.9
V_{max}	30

Table 1: Chosen parameters for the adapted PSO for MO to optimize ANN purposes implementation

603

604 To ensure a fair and balanced comparison between our proposed algorithm
 605 and the BP algorithm, we configured both methods with similar settings for
 606 a specific parameter. Since the number of epochs (iterations through the
 607 training data) is a crucial factor in both algorithms, we set this value to 10 for
 608 both approaches. However, another important hyperparameter to consider is
 609 the learning rate. We conducted experiments with the BP algorithm using a
 610 range of learning rates from 0.001 to 0.1. Interestingly, the best performance
 611 for BP in our experiments was achieved with a learning rate of 0.001.

612 5.4. Speedup analysis

613 To highlight the benefits of our proposed method's parallelization tech-
 614 niques, we present a thorough comparison between its sequential and paral-

615 lel implementations. This comparison focuses on both speed and accuracy,
616 allowing us to directly assess the performance gains achieved through paral-
617 lelization. We do not sacrifice accuracy for speed, the goal is to demonstrate
618 that parallelization can significantly reduce execution time while maintaining
619 the same level of accuracy. Furthermore, we evaluate the scalability of our
620 algorithm. In our case, we will be varying the population size (the number
621 of particles considered during optimization). By analysing the algorithm’s
622 performance under different population sizes, we can determine whether its
623 execution time scales proportionally with the increasing workload.

624 5.5. Performance metrics

625 Our evaluation relies on three performance metrics to assess the accuracy
626 of our predictions. The first metric is Mean Squared Error (MSE). MSE is a
627 widely used metric that calculates the average squared difference between the
628 predicted values and the actual values. It provides a clear indication of how
629 large the errors are on average, with smaller MSE values signifying better
630 model performance. This error can be defined as indicated in equation 5:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad (5)$$

631 Where N is the total number of samples, y_i is the actual value of the i -th
632 sample and \hat{y}_i is the predicted value of the i -th sample. However, MSE can
633 be overly sensitive to outliers, meaning a single large error can significantly
634 increase the overall MSE score. To address this limitation, we also employed
635 Mean Absolute Error (MAE). MAE calculates the average of the absolute
636 differences between predicted and actual values. Unlike MSE, MAE is less
637 influenced by outliers, offering a complementary perspective on prediction
638 accuracy. It highlights how consistently close the predictions are to the actual
639 values, regardless of their direction (positive or negative). This metric is
640 defined as follows:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i| \quad (6)$$

641 Finally, for time series forecasting specifically, we incorporate Mean Absolute
642 Scaled Error (MASE). MASE is a relative error metric that normalizes the
643 MAE by the average of the absolute values of the actual data series. This
644 normalization makes MASE particularly useful to compare the performance

645 of models on different datasets, as it removes the influence of the scale of
 646 the data. MASE values closer to 1 indicate similar average errors to the
 647 naive forecast (simply predicting the previous value), while values less than
 648 1 suggest the model outperforms the naive approach. The definition of this
 649 metric is described in the equation 7.

$$\text{MASE} = \frac{\frac{1}{N} \sum_{i=1}^N |y_i - \bar{y}|}{\frac{1}{N-1} \sum_{i=2}^N |y_i - y_{i-1}|} \quad (7)$$

650 By utilizing these three metrics we gain a better understanding of our model’s
 651 accuracy. MSE provides a general sense of error magnitude, MAE highlights
 652 consistency, and MASE offers a scale-independent comparison for time series
 653 tasks. This combination allows us to identify potential issues like outliers
 654 and pinpoint areas for improvement in the model’s prediction capabilities.

655 **6. Results and discussion**

656 This section summarises the results obtained during the whole experi-
 657 mental stage.

658 *6.1. GPU Performance Analysis*

659 As contrasted in Table 2, most of the parallel kernels significantly outper-
 660 form the serial version in terms of temporal performance, with the singular
 661 exception of one kernel (GP). This anomaly is attributed to the singular
 662 execution of this kernel, responsible for the random initialization of parti-
 663 cles, thereby incurring the overhead of various GPU memory initialization.
 664 Another kernel, the non-dominant sorting (NDS) operator, though exhibit-
 665 ing a speedup superior to GP, encounters multiple barriers within the NDS
 666 function.

667 The optimisation algorithm employs various operators, including the se-
 668 lection (SO) and mutation (MuO) operators. However, these operators are
 669 not required to execute across all particles, so they do not contribute signif-
 670 icantly to the speedup enhancement. On the other hand, the time-intensive
 671 functions are the fitness function (FF), velocity update (UV), and position
 672 update (UP), which are characterized by their iterative nature. They need
 673 to be launched recurrently throughout the algorithm’s execution.

674 In light of these observations (those which have more impact on speedup),
675 our proposal focuses on the development of an optimized parallel implemen-
676 tation for FF, UV, and UP, recognizing their critical impact on the overall
677 algorithmic performance. This strategy is highly reflected in Table 2, high-
678 lighting the substantial disparity in execution times between the sequential
679 and parallel versions of all functions, highlighting the higher efficiency and
680 performance of the parallel version compared to its best sequential version.

Kernel Name	Parallel	Sequential	Speedup
GP	0.96423	0.03748	0.03852
NDS	0.02487	0.03311	1.37555
SO	0.00043	0.01003	25
MuO	0.00041	0.01448	35
UV	0.00049	0.10329	257.502
UP	0.00014	0.03954	390
FF	0.12399	655.921	5332.69

Table 2: Time (ms) of each kernel, generation of population (GP), non-dominated sort (NDS), selection operator (SO), mutation operator (MuO), update velocity (UV), update position (UP) and the fitness function (FF).

681 A suite of experiments was systematically carried out to compare the
682 sequential and parallel version of the proposed algorithm, involving a varied
683 range of swarm sizes, and the results are presented in Table 3. Upon
684 careful examination of the data, it becomes evident that there is a subtle
685 increase in the running time as the swarm size expands. While this incre-
686 ment is indeed perceptible, it remains relatively minor in comparison to the
687 substantial growth of the swarm size. Notably, the MSE also shows a slight
688 improvement as the swarm size increases. Despite this observed improve-
689 ment, empirical analysis has revealed the existence of a saturation point at
690 approximately 200 particles. Beyond this numerical threshold, the algorithm
691 does not improve the forecast accuracy. This phenomenon can be attributed
692 to the finite capacity of individual particles to contribute valuable informa-
693 tion, thus limiting the degree of diffusion of useful knowledge within the
694 algorithm. Furthermore, the table offers an analysis, not only of memory
695 usage in megabytes (MB) but also of the percentage of GPU utilization of
696 the parallel version of the multi-objective algorithm. This dual perspective
697 provides a view of the algorithm’s operational dynamics across different pop-

698 ulation sizes. This multifaceted evaluation enhances our understanding of
699 the algorithm’s performance, enabling us to identify and optimize the most
700 efficacious population size. Such optimisation is crucial for achieving both
701 enhanced algorithmic efficiency and improved accuracy.

Population Size	Time (s)	MSE	Memory Usage (MB)	GPU Usage
80	17.198	0.009	400	70%
160	31.732	0.007	441	70%
200	47.101	0.005	445	72%
250	78.102	0.007	450	85%
300	113.814	0.005	450	97%

Table 3: Scalability of the designed algorithm using different population size.

702 Table 4 presents an overview of the average execution times associated
703 with particle computations, with each row representing a distinct number of
704 particles (blocks), with each block having as many threads as the maximum
705 amount of neurons in a topology. The table’s first column enumerates the
706 varying number of particles, followed by columns displaying the minimum,
707 maximum, and average execution times. These results offer insights into the
708 systematic increase of computation time for each particle as the total number
709 of particles within the experiment grows. This progressive trend in execu-
710 tion time can be attributed to the larger data volumes being processed on
711 the device, a direct consequence of the increasing number of particles under
712 consideration. Understanding this correlation between computational load
713 and execution time is essential for optimizing the algorithm’s performance
714 when scaling it to accommodate larger populations.

Particles	Min	Max	Avg
80	0.065	2.562	0.198
160	0.284	2.759	0.400
200	0.462	3.134	0.612
250	0.788	3.496	0.994
300	1.291	4.622	1.598

Table 4: Population size - Time (sec) execution of an particle.

715 Table 5 provides a detailed breakdown of the GPU utilization percent-
716 ages for each kernel in our algorithm, offering an exhaustive view of their
717 computational costs. Beginning with the least demanding, the GP function
718 demonstrates minimal GPU utilization, followed by SO and UP. The subse-
719 quent tier comprises MO and NDS, with the NDS kernel demanding more
720 computational resources due to its sorting algorithm based on solution dom-
721 inance. The MO kernel, responsible for traversing the particle structure for
722 mutation, also exhibits significant computational intensity. The UV kernel
723 emerges as the second most computationally intensive, as it uses each parti-
724 cle’s structure within every iteration for updates. At the top of the hierarchy,
725 the FF kernel stands out as the most time-consuming component, primarily
726 due to its extensive data processing and iterative evaluations for all particles
727 and data points.

728 In response to these observations, dedicated efforts were directed toward
729 optimizing the FF kernel. We adopted a strategy of allocating one particle
730 per block and optimizing the number of threads to align with the maximum
731 number of neurons. This allowed for the parallel execution of all computa-
732 tions within the FF kernel. This approach resulted in a substantial acceler-
733 ation of computational speed. These observed findings emphasise the signif-
734 icant advancements achieved by our implementation. Moreover, it delivers
735 outstanding speed-up metrics, particularly in the execution of complex algo-
736 rithmic functions. This demonstrates its remarkable performance efficiency,
737 making a noteworthy contribution to the contemporary state-of-the-art in
738 multi-objective optimisation.

Kernel Name	Usage Rate
Fitness function (FF)	36%
Update velocity (UV)	21%
Front order (NDS)	15%
Mutation operators (MuO)	14%
Selection operator (SO)	7%
Update position (UP)	5%
Generation of particles (GP)	2%

Table 5: Kernel profiling.

739 *6.2. Model Performance Analysis*

740 The next experiment is intended to compare the accuracy of both solu-
741 tions (sequential and parallel) and the results are outlined in Table 6. The
742 columns labelled 2, 3, 4, and 5 provide a detailed breakdown of the exper-
743 imental outcomes for buildings B1, B2, B3, and B4, respectively, while the
744 first column explains the specific metric under scrutiny. In the context of
745 both sequential and parallel experiments, this table effectively shows the av-
746 erage fitness values, gathering the results of the MSE calculations out of 10
747 iterations of each version. Additionally, the table presents the best and worst
748 fitness values, representing the minimum and maximum MSE obtained over
749 ten runs, complemented by the standard deviation, which offers a profound
750 understanding of the result distribution. Upon careful examination of the
751 data within Table 6, an interesting trend emerges. Both the sequential and
752 parallel implementations yield results with remarkable similarities. However,
753 it is worth highlighting that the parallel version exhibits a discernible reduc-
754 tion in average error values, underscoring its superior performance efficacy
755 when compared to its sequential counterpart. This observation underscores
756 the tangible benefits of parallelization in enhancing the accuracy of our al-
757 gorithm.

	Measures	B1	B2	B3	B4
Sequential	Average Fitness	0.020	0.145	0.035	0.026
	Best Fitness	0.006	0.014	0.009	0.018
	Worst Fitness	0.035	0.350	0.093	0.045
	Standard Deviation	± 0.001	± 0.012	± 0.032	± 0.010
Parallel	Average Fitness	0.009	0.010	0.013	0.019
	Best Fitness	0.006	0.005	0.011	0.015
	Worst Fitness	0.011	0.014	0.018	0.022
	SD	± 0.001	± 0.002	± 0.001	± 0.001

Table 6: Average MSE of sequential and parallel implementations.

758 The following experiment aims to compare the accuracy of BP and the
759 adapted PSO for MO to optimize ANN, both in their parallel versions, show-
760 casing various error metrics: Mean Squared Error (MSE), Mean Absolute Er-
761 ror (MAE), and Mean Absolute Scaled Error (MASE). The experimentation
762 has been carried out for buildings B1, B2, B3, and B4, which were previ-
763 ously used for the experiments in Table 6. The results distinctly highlight
764 the advantages of our algorithm over the BP technique.

	MSE	B1	B2	B3	B4	REE	Uruguay
Adapted PSO for MO to optimize ANN	Average Fitness	0.009	0.010	0.013	0.019	0.020	0.014
	Best Fitness	0.006	0.005	0.011	0.015	0.015	0.008
	Worst Fitness	0.011	0.014	0.018	0.022	0.024	0.189
	Standard Deviation	0.001	0.002	0.001	0.001	0.002	0.003
BP	Average Fitness	0.051	0.018	0.013	0.019	0.039	0.027
	Best Fitness	0.035	0.018	0.011	0.015	0.032	0.025
	Worst Fitness	0.135	0.018	0.018	0.022	0.043	0.029
	Standard Deviation	0.024	0.002	<0.001	<0.001	0.002	0.001

Table 7: Comparative Analysis of MSE for Parallel Adapted PSO for MO to optimize ANN and Back-propagation Algorithms Across Buildings.

	MAE	B1	B2	B3	B4	REE	Uruguay
Adapted PSO for MO to optimize ANN	Average Fitness	0.078	0.077	0.082	0.096	0.118	0.096
	Best Fitness	0.061	0.058	0.071	0.083	0.103	0.074
	Worst Fitness	0.094	0.090	0.097	0.115	0.129	0.112
	Standard Deviation	0.009	0.009	0.008	0.011	0.008	0.012
BP	Average Fitness	0.181	0.103	0.099	0.136	0.163	0.132
	Best Fitness	0.139	0.102	0.092	0.133	0.153	0.128
	Worst Fitness	0.334	0.106	0.119	0.139	0.171	0.139
	Standard Deviation	0.050	0.001	0.007	0.001	0.004	0.003

Table 8: Comparative Analysis of MAE for Parallel Adapted PSO for MO to optimize ANN and Back-propagation Algorithms Across Buildings.

765 The average outcomes presented in Tables 7, 8, and 9, demonstrate across
766 all metrics and for each building that our algorithm delivers superior perfor-
767 mance, achieving lower average values of MSE, MAE, and MASE.

768 Beyond the average values, the optimal (best) figures achieved by our
769 algorithm in all metrics are lower than those of the BP algorithm, under-
770 scoring its effectiveness. Additionally, the least favorable (worst) values and
771 the standard deviation further support the observation that our algorithm
772 provides more precise results.

773 The graph in Figure 8 displays the predicted EC over a span of 100
774 hours from a test data set. On these charts, the x-axis indicates the time
775 for which each prediction is made, while the y-axis indicates the normalized
776 consumption for that specific hour.

	MASE	B1	B2	B3	B4	REE	Uruguay
Adapted PSO for MO to optimize ANN	Average Fitness	0.510	0.569	0.779	0.096	0.118	0.096
	Best Fitness	0.433	0.494	0.636	0.083	0.103	0.074
	Worst Fitness	0.570	0.642	0.935	0.115	0.129	0.112
	Standard Deviation	0.049	0.038	0.091	0.011	0.008	0.012
BP	Average Fitness	0.843	0.797	0.943	0.799	0.797	0.744
	Best Fitness	0.662	0.787	0.897	0.782	0.745	0.721
	Worst Fitness	1.815	0.835	1.182	0.817	0.831	0.781
	Standard Deviation	0.289	0.009	0.072	0.008	0.019	0.016

Table 9: Comparative Analysis of MASE for Parallel Adapted PSO for MO to optimize ANN and Back-propagation Algorithms Across Buildings.

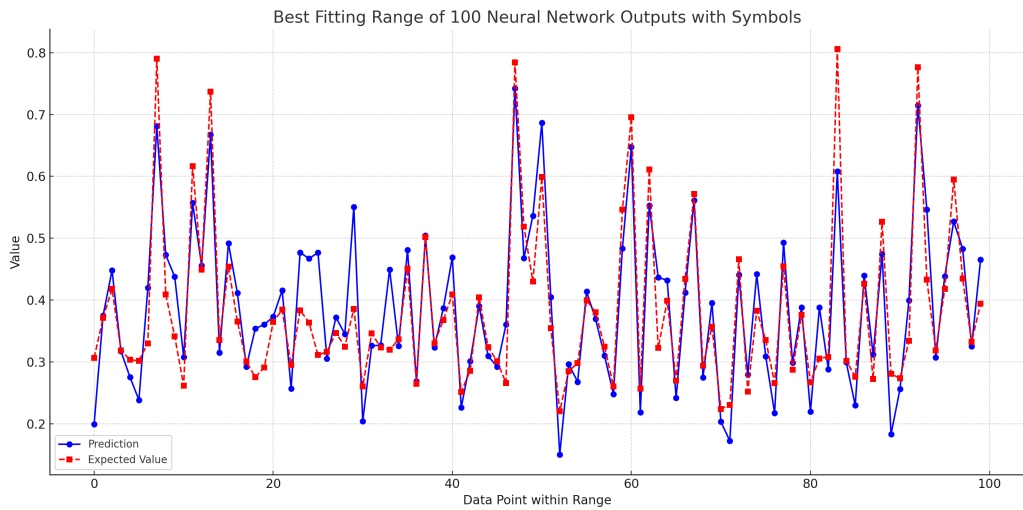


Figure 8: Example of predictions of energy consumption

777 *6.3. Advantages and Limitations*

778 The main advantage of the proposed method is its ability to simultane-
779 ously find the minimum number of hidden neurons and fit the weights of the
780 neural network specifically at the same time in a single framework, achieved
781 by the proposed adaptation of the multi-objective PSO with the inclusion of
782 mutation operators and variations of the original PSO operators. All of these
783 operators were designed taking into account the dynamic size of each indi-
784 vidual to guarantee neural network topology diversity in the swarm. This is
785 a key element of the proposed method, as otherwise the swarm would quickly
786 converge towards only using a fixed number of hidden neurons. Furthermore,
787 due to the parallel implementation provided for GPU in this paper, the pro-
788 posed method will be able to efficiently train and find the optimal topology
789 even with large volumes of data, as it can make great use of the massively
790 parallel architecture of the GPU.

791

792 However, the proposed method still has some minor limitations. First,
793 similarly to other machine learning models, even though the proposed method
794 reported great results in the experiments made in this paper, there is no
795 guarantee that it will always provide the best results. Additionally, the hy-
796 perparameters that control the proposed method may need to be tuned when
797 the algorithm is applied to other contexts. In those cases, it is quite likely
798 that at least the hyperparameters that control the complexity of the trained
799 neural network (minimum and maximum number of hidden neurons) may
800 need to be tuned differently. Second, for low volumes of data, it is likely that
801 the sequential CPU implementation will be faster than the parallel GPU im-
802 plementation. This is a common issue for GPU implementations, since GPUs
803 are capable of doing computation faster by exploiting their massively paral-
804 lel architecture. If there is not enough data to use all of the GPU's cores,
805 its lower instruction latency and the slow communication time between the
806 CPU and GPU will become a major bottleneck that will limit the perfor-
807 mance of the parallel implementation. Nonetheless, considering the current
808 trend toward greater data availability and advances in sensor technologies,
809 this limitation tends to be less significant in the energy sector, where the
810 volume of data available is increasingly larger.

811 **7. Conclusions and Future Works**

812 In this research paper, we have successfully developed an adapted version
813 of the PSO algorithm for multi-objective purposes. Our adaptation aims
814 to achieve the most accurate and optimal model in terms of error and the
815 optimal structure of the predictive model. We employed ANNs as the model
816 to predict the energy consumption of a series of buildings on an hourly ba-
817 sis. Additionally, we have not only implemented the adaptation of PSO for
818 multi-objective tasks but also proposed a parallel GPU implementation. The
819 problem of obtaining the optimal topology, number of neurons and weights
820 that obtain a minimum error is known for its computational cost and that
821 through this proposal we have seen a significant reduction.

822 The importance of obtaining speedy and parallel implementations of the
823 PSO algorithm, especially for multi-objective tasks, can be highlighted for
824 the following reasons. First, the efficiency in solving large-scale problems as
825 traditional PSO algorithms may struggle with large swarm sizes and high-
826 dimensional problems. By leveraging parallel GPU implementations, we sig-
827 nificantly improved the efficiency of solving such complex problems. Sec-
828 ondly, enhancing real-time decision-making due to the fact that in various do-
829 mains, including energy consumption prediction, real-time decision-making is
830 crucial. Speedy and parallel implementations of the PSO algorithm enabled
831 us to obtain optimal solutions quickly, facilitating timely decision-making
832 processes. Thirdly, the optimisation of limited computational resources be-
833 cause GPU-based parallel PSO algorithms allow us to harness the power of
834 consumer-level Graphics Processing Units, making efficient use of available
835 computational resources. This is particularly important in energy efficiency
836 scenarios where access to high-performance computing infrastructure is lim-
837 ited. And lastly, advancements in swarm intelligence research as the devel-
838 opment of efficient and effective parallel PSO algorithms contributes to the
839 broader field of swarm intelligence research. By pushing the boundaries of
840 optimisation techniques, we can continue to explore new applications and
841 solve increasingly complex problems as the one discussed in this study.

842 The main advantage of the proposed method lies in its ability to simulta-
843 neously determine the minimum number of hidden neurons and fit the neural
844 network weights within a unified framework, which improves both efficiency
845 and performance. This approach not only guarantees diverse neural network
846 topologies, but also circumvents the conventional challenge of convergence to
847 suboptimal architectures. Despite these strengths, the method is not without
848 limitations, as some of its parameters may need to be fine-tuned depending
849 on the complexity of the problem and small data volumes may not be able

850 to take advantage of the parallel implementation due to the lower latency of
851 the GPU architecture.

852 Finally, as future work and in light of the findings from this study, we sug-
853 gest several avenues for further exploration. The first one is the fusion of PSO
854 with other optimization algorithms, leading to the creation of hybrid models.
855 By tapping into the strengths of other swarm intelligence algorithms, such
856 as ant colony optimization or bee algorithm, there is potential to formulate
857 solutions of even greater robustness and efficacy. Moreover, while the current
858 research hinged on the application of ANNs, the prediction could be further
859 enriched by incorporating more intricate neural architectures such as Con-
860 volutional Neural Networks or Recurrent Neural Networks. Another frontier
861 for expansion lies in the realm of parallel implementations. Our study has al-
862 ready highlighted the merits of GPU-based parallelism; however, catapulting
863 this to multi-GPU systems or even distributed computational frameworks
864 could improve the way for better optimizations, especially fitting for grand-
865 scale challenges.

866

867 **Acknowledgments**

868 We acknowledge financial support from Ministerio de Ciencia e Innovación
869 (Spain) (Grant PID2020-112495RB-C21 funded by MCIN/ AEI /10.13039/501100011033).

870 **References**

- 871 [1] Anam Kalair, Naeem Abas, Muhammad Shoaib Saleem, Ali Raza
872 Kalair, and Nasrullah Khan. Role of energy storage systems in energy
873 transition from fossil fuels to renewables. *Energy Storage*, 3(1):e135,
874 2021.
- 875 [2] F. Martínez-Álvarez, G. Asencio-Cortés, J. F. Torres, D. Gutiérrez-
876 Avilés, L. Melgar-García, R. Pérez-Chacón, C. Rubio-Escudero, J. C.
877 Riquelme, and A. Troncoso. Coronavirus optimization algorithm: A
878 bioinspired metaheuristic based on the covid-19 propagation model. *Big*
879 *Data*, 8(4):308–322, 2020. PMID: 32716641.
- 880 [3] Martin Nerlinger and Sebastian Utz. The impact of the russia-ukraine
881 conflict on energy firms: A capital market perspective. *Finance Research*
882 *Letters*, 50:103243, 2022.

- 883 [4] Benoit Eynard, Vincenzo Nigrelli, Salvatore Massimo Oliveri, Guillermo
884 Peris-Fajarnes, and Sergio Rizzuti, editors. *Advances on Mechanics, De-*
885 *sign Engineering and Manufacturing: Proceedings of the International*
886 *Joint Conference on Mechanics, Design Engineering & Advanced Man-*
887 *ufacturing (JCM 2016), 14-16 September, 2016, Catania, Italy.* Lec-
888 ture Notes in Mechanical Engineering. Springer International Publish-
889 ing, Cham, 2017.
- 890 [5] Christos Sardianos, Iraklis Varlamis, Christos Chronis, George Dimi-
891 trakopoulos, Abdullah Alsalemi, Yassine Himeur, Faycal Bensaali, and
892 Abbes Amira. The emergence of explainability of intelligent systems:
893 Delivering explainable and personalized recommendations for energy ef-
894 ficiency. *International Journal of Intelligent Systems*, 36(2):656–680,
895 2021.
- 896 [6] J. R. S. Iruela, L. G. B. Ruiz, M. I. Capel, and M. C. Pegalajar. A
897 tensorflow approach to data analysis for time series forecasting in the
898 energy-efficiency realm. *Energies*, 14(13), 2021.
- 899 [7] D. Criado-Ramón, L.G.B. Ruiz, and M.C. Pegalajar. Electric demand
900 forecasting with neural networks and symbolic time series representa-
901 tions. *Applied Soft Computing*, 122:108871, 2022.
- 902 [8] M.C. Pegalajar, L.G.B. Ruiz, M.P. Cuéllar, and R. Rueda. Analysis and
903 enhanced prediction of the spanish electricity network through big data
904 and machine learning techniques. *International Journal of Approximate*
905 *Reasoning*, 133:48–59, 2021.
- 906 [9] Luis Gonzaga Baca Ruiz, Manuel Pegalajar Cuéllar, Miguel Delgado
907 Calvo-Flores, and María Del Carmen Pegalajar Jiménez. An application
908 of non-linear autoregressive neural networks to predict energy consump-
909 tion in public buildings. *Energies*, 9(9), 2016.
- 910 [10] L.G.B. Ruiz, R. Rueda, M.P. Cuéllar, and M.C. Pegalajar. Energy
911 consumption forecasting based on elman neural networks with evolutive
912 optimization. *Expert Systems with Applications*, 92:380–389, 2018.
- 913 [11] L.G.B. Ruiz, M.I. Capel, and M.C. Pegalajar. Parallel memetic algo-
914 rithm for training recurrent neural networks for the energy efficiency
915 problem. *Applied Soft Computing*, 76:356–368, 2019.

- 916 [12] L. Cabezón, L. G. B. Ruiz, D. Criado-Ramón, E. J. Gago, and M. C.
917 Pegalajar. Photovoltaic energy production forecasting through machine
918 learning methods: A scottish solar farm case study. *Energies*, 15(22),
919 2022.
- 920 [13] Ioannis P. Panapakidis and Athanasios S. Dagoumas. Day-ahead elec-
921 tricity price forecasting via the application of artificial neural network
922 based models. *Applied Energy*, 172:132–151, 2016.
- 923 [14] M. Bouzerdoum, A. Mellit, and A. Massi Pavan. A hybrid model
924 (sarima–svm) for short-term power forecasting of a small-scale grid-
925 connected photovoltaic plant. *Solar Energy*, 98:226–235, 2013.
- 926 [15] A. Lahouar and J. Ben Hadj Slama. Day-ahead load forecast using
927 random forest and expert input selection. *Energy Conversion and Man-
928 agement*, 103:1040–1051, 2015.
- 929 [16] D. Hadjout, J.F. Torres, A. Troncoso, A. Sebaa, and F. Martínez-
930 Álvarez. Electricity consumption forecasting based on ensemble deep
931 learning with application to the algerian market. *Energy*, 243:123060,
932 2022.
- 933 [17] Rahul Kumar Agrawal, Frankle Muchahary, and Madan Mohan Tri-
934 pathi. Ensemble of relevance vector machines and boosted trees for
935 electricity price forecasting. *Applied Energy*, 250:540–548, 2019.
- 936 [18] Azim Heydari, Meysam Majidi Nezhad, Elmira Pirshayan, Davide Asti-
937 aso Garcia, Farshid Keynia, and Livio De Santoli. Short-term electricity
938 price and load forecasting in isolated power grids based on composite
939 neural network and gravitational search optimization algorithm. *Applied
940 Energy*, 277:115503, 2020.
- 941 [19] Zhang Yong, Yuan Li-juan, Zhang Qian, and Sun Xiao-yan. Multi-
942 objective optimization of building energy performance using a particle
943 swarm optimizer with less control parameters. *Journal of Building En-
944 gineering*, 32:101505, 2020.
- 945 [20] Biswa Mohan Sahoo, Tarachand Amgoth, and Hari Mohan Pandey.
946 Particle swarm optimization based energy efficient clustering and sink
947 mobility in heterogeneous wireless sensor network. *Ad Hoc Networks*,
948 106:102237, 2020.

- 949 [21] Mohamed Abd Elaziz, Abdelghani Dahou, Laith Abualigah, Liyang Yu,
950 Mohammad Alshinwan, Ahmad M Khasawneh, and Songfeng Lu. Ad-
951 vanced metaheuristic optimization techniques in applications of deep
952 neural networks: a review. *Neural Computing and Applications*, pages
953 1–21, 2021.
- 954 [22] M. McInerney and A.P. Dhawan. Use of genetic algorithms with back-
955 propagation in training of feedforward neural networks. In *IEEE Inter-
956 national Conference on Neural Networks*, pages 203–208 vol.1, 1993.
- 957 [23] Lanouar Charfeddine, Esmat Zaidan, Ahmad Qadeib Alban, Hamdi
958 Bennisr, and Ammar Abulibdeh. Modeling and forecasting electricity
959 consumption amid the COVID-19 pandemic: Machine learning vs. non-
960 linear econometric time series models. *Sustainable Cities and Society*,
961 98:104860, November 2023.
- 962 [24] Chang Haili. Energy-saving design and implementation in metro weak
963 current systems: a case study of the kunming metro system. *Urban Rail
964 Transit*, 7, 12 2021.
- 965 [25] Huize Liang, Yuying Zhang, Peiyu Yang, Lie Wang, and Chunlei Gao.
966 Comparison and analysis of prediction models for locomotive traction
967 energy consumption based on the machine learning. *IEEE Access*, 2023.
- 968 [26] Catalin Stoean, Miodrag Zivkovic, Aleksandra Bozovic, Nebojsa Ba-
969 canin, Roma Strulak-Wójcikiewicz, Milos Antonijevic, and Ruxandra
970 Stoean. Metaheuristic-based hyperparameter tuning for recurrent deep
971 learning: Application to the prediction of solar energy generation. *Ar-
972 ioms*, 12(3), 2023.
- 973 [27] Salah Bouktif, Ali Fiaz, Ali Ouni, and Mohamed Adel Serhani. Multi-
974 sequence lstm-rnn deep learning and metaheuristics for electric load fore-
975 casting. *Energies*, 13(2), 2020.
- 976 [28] Manoj Kumar, Aryabartta Sahu, and Pinaki Mitra. A comparison of
977 different metaheuristics for the quadratic assignment problem in accel-
978 erated systems. *Applied Soft Computing*, 100:106927, 2021.
- 979 [29] Jason Runge and Radu Zmeureanu. Forecasting Energy Use in Buildings
980 Using Artificial Neural Networks: A Review. *Energies*, 12(17):3254,
981 January 2019.

- 982 [30] Yanhong Zhuo, Tao Zhang, Feng Du, and Ruilin Liu. A parallel par-
983 ticle swarm optimization algorithm based on gpu/cuda. *Applied Soft*
984 *Computing*, 144:110499, 2023.
- 985 [31] Dineshan Subramoney and Clement N. Nyirenda. Multi-swarm pso al-
986 gorithm for static workflow scheduling in cloud-fog environments. *IEEE*
987 *Access*, 10:117199–117214, 2022.
- 988 [32] Sehrish Malik and DoHyeun Kim. Prediction-learning algorithm for effi-
989 cient energy consumption in smart buildings based on particle regenera-
990 tion and velocity boost in particle swarm optimization neural networks.
991 *Energies*, 11(5), 2018.
- 992 [33] Abdal Kasule and Kürşat Ayan. Using pso and genetic algorithms to op-
993 timize anfis model for forecasting uganda’s net electricity consumption.
994 *Sakarya University Journal of Science*, 24:324–337, 04 2020.
- 995 [34] Shunpan Liang, Ze Liu, Dianlong You, Weiwei Pan, Junjie Zhao,
996 and Yefan Cao. PSO-NRS: an online group feature selection algo-
997 rithm based on PSO multi-objective optimization. *Applied Intelligence*,
998 53(12):15095–15111, June 2023.
- 999 [35] Zhiming Lv, Dangdang Niu, Shuqin Li, and Hongguang Sun. Multi-
1000 surrogate assisted pso with adaptive speciation for expensive multimodal
1001 multi-objective optimization. *Applied Soft Computing*, 147:110724, 2023.
- 1002 [36] Selim Yilmaz and Sevil Sen. Chapter 2 - metaheuristic approaches for
1003 solving multiobjective optimization problems. In Seyedali Mirjalili and
1004 Amir H. Gandomi, editors, *Comprehensive Metaheuristics*, pages 21–48.
1005 Academic Press, 2023.
- 1006 [37] Kevin Jesus, Delia Senoro, Jennifer Dela Cruz, and Eduardo Chan. A
1007 hybrid neural network–particle swarm optimization informed spatial in-
1008 terpolation technique for groundwater quality mapping in a small island
1009 province of the philippines. *Toxics*, 9:273, 10 2021.
- 1010 [38] J.R.S. Iruela, L.G.B. Ruiz, M.C. Pegalajar, and M.I. Capel. A parallel
1011 solution with gpu technology to predict energy consumption in spatially
1012 distributed buildings using evolutionary optimization and artificial neu-
1013 ral networks. *Energy Conversion and Management*, 207:112535, 2020.

- 1014 [39] Parul Agarwal, R.K. Agrawal, and Baljeet Kaur. Multi-objective par-
1015 ticle swarm optimization with guided exploration for multimodal prob-
1016 lems. *Applied Soft Computing*, 120:108684, 2022.
- 1017 [40] James Kennedy and Russell Eberhart. Particle swarm optimization.
1018 In *Proceedings of ICNN'95-international conference on neural networks*,
1019 volume 4, pages 1942–1948. IEEE, 1995.
- 1020 [41] Wafa Shafqat, Sehrish Malik, Kyu-Tae Lee, and Do-Hyeun Kim. Pso
1021 based optimized ensemble learning and feature selection approach for
1022 efficient energy forecast. *Electronics*, 10(18):2188, 2021.
- 1023 [42] Luis F Grisales-Noreña, Oscar Danilo Montoya, and Carlos Andrés
1024 Ramos-Paja. An energy management system for optimal operation of
1025 bss in dc distributed generation environments based on a parallel pso
1026 algorithm. *Journal of Energy Storage*, 29:101488, 2020.
- 1027 [43] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. A review
1028 on genetic algorithm: past, present, and future. *Multimedia tools and
1029 applications*, 80:8091–8126, 2021.
- 1030 [44] Xoshiro, xoroshiro generators and the PRNG shootout,
1031 <http://xoshiro.di.unimi.it/>, May 2018.