Automated Algorithm Selection: Experiments on the Strip Packing problem

Corina Dimitriu

On the premise of boosting runtime performance and solution quality in the field of NP-hard combinatorial problems, automated algorithm selection finds its meaning under the well justified assumption that the perfect algorithm, capable of solving each and every instance, does not exist. This paper investigates and improves state-of-the-art approaches in automated algorithm selection, while tacking the 2-dimensional strip packing problem (2SP) as experimental framework. A novel solution to objectively selecting the best algorithm in the portofolio is proposed, based on statistically significant optimality metrics, feature space augmentation and (ensembles of) neural networks. The results within the ablation study prove the influence of the feature collection methodology, the dataset's structure and biases and the interaction amongst features along the algorithm's execution, on the instances developing preferences towards certain heuristics within the portofolio.

1 Introduction

As dimensionality, variability and multimodality are known to define the inner core of learning from data and exploiting its properties, anticipating the best possible algorithm within a given portofolio could spare the computational costs of actually running each solution individually, especially if there is a wide variety of already proposed solutions and there is always a chance of new resolutions yet to come. The 2SP problem consists in placing rectangles on a strip of the given width for minimum strip length. In formal terms, for a problem instance (p_i), given a rectangle strip with an undefined height H_{p_i}, of possible complete and feasible solutions from search space (p) and a finite width W_{p_i}, the objective is to pack all n_{p_i} rectangles (allowing 90 degress rotations) into the strip [1, 2]. Since the problem itself only facilitates a combinatorial context for experi-

menting with different augmentation and prediction techniques and is not the main subject of the current study, it is only worth mentioning that while not being one of the most intensively studied tasks in the technical field (not as much as the Satisfiability problem, for instance), it still benefits from a large benchmark of proposed heuristics due to its practical applications in the industry [1, 3]. Nevertheless, the more heuristics to employ for automatic selection, the more features are necessary to be captured from each instance aside in order to accurately distinguish between them; in addition, as the experiments will reveal, a certain uniformity degree among all instances has to be ensured in terms of feature significance, so that the prediction techniques generalize well and stay away from fine grained parameter dependencies and overfitting on the training dataset.

Among the benefits of developing an effective automated algorithm selection methodology, in what concerns both solution quality and resources consumption, large instances in terms of sides' dimensions, strip width or number of rectangles to pack, are difficult to be solved by certain heuristics, meaning their execution requires lots of time and memory [4]. At the same time, computing some defining features usually requires only a few iterations throughout the instance's specifications, which makes a direct feature-based prediction via machine learning algorithms even more convenient as long as it takes a decent amount of computational power (at least for the testing phase, which is redundant, compared to the training phase which is only performed once) [3, 5]. Moreover, an empirical connection between the dataset's features and the preferred solutions within a range of algorithms, no matter which problem to be solved we place under discussion, is a valuable and traditional starting point for generating theoretical content around the problem's statement itself.

Regarding the general flow of an automated algorithm selection method, accepted as the leading standard by the contributions in this paper, it is necessary to have a portofolio displaying results of each algorithm from a collection of polynomial time heuristics on selected benchmark instances [6, 7]. In the Strip Packing problem's context, to the best of our knowledge, there is no portofolio available among the academic paper or code resources, which is why the first contribution of this paper is building a portofolio of 1144 instances for the Strip Packing problem, comprising the results and the one-hot-encoded optimality conclusion for each instance with respect to six heuristic (not machine learning related) approaches: genetic algorithms, particle swarm optimization, flooplanning and skyline, hyper search and local search, priority guillotine constraint optimization and Steinberg algorithm [8, 10, 12, 14, 15, 16]. Each instance is then sub-

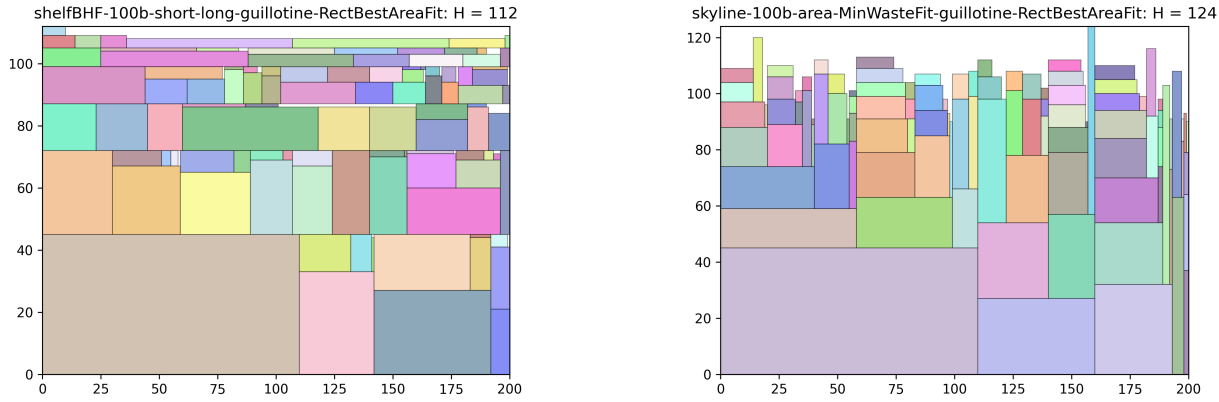

Figure 1: Strip Packing graphed solution example using floorplanning heuristic and skyline heuristic, respectively [8].

ject to extracting meaningful features, which will be further fed as input to the machine learning prediction algorithm. In the context of the solution proposed within this paper, the prediction algorithm (re)composes and decomposes features while it is in turn composed of neural networks. The learning process keeps improving metrics exposing the difference between the real optimal algorithm(s) and the one(s) predicted to be optimal by the networks. In our case, these metrics are given by the chosen loss functions and the prediction algorithm is only run once for all heuristics (which is why the one-hot-encoded optimality conclusion is computed in the first place, during the portofolio creation stage), but other approaches consider each heuristic separately to predict a number of numerical problem's outputs equal to the number of algorithms within the portofolio, and then select the best performing algorithm by applying a min(imum) operation on the results. The latter methodology is known to be more successfull in terms of training and testing accuracy, especially when it comes to rather small datasets, since it is not expected to learn joint connection patterns, which would occur between the instances and all the algorithms taken together [3].

The paper logic is structured as follows. Section 2 describes State-of-the-Art approaches in the field of automated algorithm selection, with respect to the Strip Packing problems, while also emphasising the contributions detached from these approaches in order to be integrated within the method this paper aims to build and experiment with. Section 3 mentions some generic classification criteria of benchmark instances and exemplifies their forming. Section 4 details the own contribution in the present paper, through step-by-step building four neural network models for algorithm selection: a neural network with linear layers displayed in a hill topology

(the concept of hill topology is introduced by the current paper in Section 4), a novel type of convolutional network, obtained by reversing the encoder-decoder architecture, a voting assembly of neural networks of the first category and an assembly of neural networks of the second category; moreover, a new concept, model-based feature space augmentation, is added and proved with respect to the base methods previously referred, along with Friedman / ANOVA statistical tests for the portofolio creation and distribution-based clusterization techniques for the dataset expansion. Section 5 details the process of parameter hypertuning, including specifications of manual and automatic heuristics selection, and compares the approach in Section 4 with two other ensemble approaches working well with small datasets: Random Forest and XGBoost. Section 6 aims to formulate conclusions related to the experimental setup in Section 5, while Section 7 and 8 briefly summarize further directions and final remarks on the automated algorithm selection topic.

2 State-of-the-Art approaches

This paper makes use of a few landmarks from previous research, while keeping others from the ones listed below, within the sections dedicated to each relevant paper, for further work. The solution described within Section 4 studies the distribution of the best performing algorithm, similarly to the third reference, instead of a collection of distributions, each for one algorithm in the portofolio, as employed by the more common method [3]. Keeping in mind the observation that ensembles perform better than one classifier standalone, stated in the third reference, this paper comes up with ensembles of neural networks instead of ensembles of classical machine learning algorithms [2, 3]. Taking inspiration from the first reference and the second one, the proposed solution analyzes only critical features from the problem's dedicated literature, while artificially creating new, unannotated features, under the hood of fitting gaussian distributions with the annotated ones [1, 2]. Last but not least, the proposed solution allows more than one optimal algorithm per instance, while trying to ensure a quite balanced overall ratio between winning algorithms, as it is the case of the fourth reference, which is why the one-hot-encoded performed for classification is different from the classical strategy and is able to contain as many values of 1 as there are algorithms obtaining the minimum height value [4].

2.1 A framework to select heuristics for the rectangular two-dimensional strip packing problem [1]

(Alvaro Neuenfeldt Júnior, Julio Siluk, Matheus Francescatto, Gabriel Stieler and David Disconzi)

This solution connects instance features (the *characteristics* space) to typical CH (constructive heuristics) algorithms, associates CH with IH (improvement heuristics, what we are looking for) through machine learning techniques, and returns back to instances through clusterization (even though never explicitly) depending on the most suitable CH in approaching them. In characteristics space, a total of 19 features were developed to represent the rectangular 2D-SPP main characteristics. All features are related to composition and statistics variations in the rectangles and the projected strip area, perimeter, and dimensions, in addition to intrinsic problem instances information such as the number of rectangles to be packed, the number of different rectangles dimensions, proportion of different rectangles, and larger/smaller rectangles dimensions values etc. In heuristics space, a total of six CH were selected: bottom-left; bottom-left-fill; best-fit; best-fit-fill; fast layer-based heuristic; and fast layer-fill-based heuristic. While the CH are used to fit the classification process (by transitioning from the characteristics space to a grouping logic of the instances), the corresponding class label of a problem instance is one or more, not mutually exclusive IH, with the lowest value for the fitness function. A total of six IH were selected to be used in this study: dynamic random local search (DR); complete random local search (CR); random weight local search (RW); tabu search (TS); genetic algorithm (GA); and simulated annealing (SA). Regarding the fitting of supervised ML models, the svmPoly and nnet are used as reference values to create a CD (critical difference) precedence and, consequently, pls, spls, xgbTree, gbm, hda, rf, and Rborist are significantly equal. Therefore, with the best accuracy performance, the svmPoly was selected as the best option for BL. In all five remaining CH, the same procedure was adopted to select supervised machine learning techniques. The rf was selected for BF and FH, the svmPoly for BLF, the gbm for BFF, and the xgbTree for FHF. To briefly encapsulate the solution's data flow, for the problem space, three different types of dataset must be generated: "Main dataset"; "Validation dataset"; and "New dataset". Existing instances are used as a base for the "Main dataset", which, along with the other two datasets (which are artificially generated entirely), is supplied with additional instances generated by the the cutting and packing problem instances

generator 2DCPackGen. For the "Main dataset", problem instances variations were fitted using eight different beta probabilistic non-uniform distribution curves to guarantee a variability in the geometric rectangles and strip shapes. The uniform distribution curve was used for the "Validation dataset" to obtain different problem instances' characteristics compared to the "Main dataset". The multi-label transformation method applied prioritizes the elimination of multi-label instances (the exclusion method), and each CH has its specific number of multi-label instances. One significant quality metric is the space waste in the strip. For the rectangular 2D-SPP, solutions of good quality are found when few waste spaces are obtained to pack all rectangles into the strip. The solution quality peaks around 80% if considering the first two best performing algorithms at the testing stage.

2.2 A machine learning approach for automated strip packing algorithm selection [2]

(R.G. Rakotonirainy)

Unlike the previous approach, this solution comes with explicit clusterization of problem instances based on geometric features, without intermediately passing through the CH space. Being the first framework in the direction of automated algorithm selection for the strip packing problem, this solution defines the concepts of "training" and "prediction", for the first time in the field. The training phase involves six steps. In step 1, instance generation, suitable packing instances are selected. This is achieved by collecting benchmark problem instances documented in the literature, and also by generating new instances from existing benchmark generators. Step 2, feature selection, consists of selecting the most appropriate features in respect of which to cluster the benchmark data and to measure the influence of problem characteristics on the algorithm performance. After exploratory analysis of the characteristics that best describe the data instances, which accurately boost the algorithm's performance, the solution ends up with four meaningful features: the maximum aspect ratio of all items of an instance, the maximum area ratio of all pairs of items of an instance, the heterogeneity ratio, and the width ratio. For instance, the aspect ratio (which, attached to a rectangle, is defined as the ratio between its length along the smaller side dimension and its length along the larger side dimension, respectively) provides information on the shapes of the items in an instance: relatively small values of the maximum aspect ratio feature

indicate that the respective instance is heavily populated by approximately square-shaped items. In the next step, characteristics measurement, the characteristic values of each instance generated in step 1 are calculated based on the selected features of step 2. In step 4, performance evaluation, the problem instances were solved using seven state-of-the-art strip packing metaheuristics from the literature: ISA (the two-stage intelligent search algorithm), Hybrid GA, SRA (the simple randomised algorithm), IA (the efficient intelligent search algorithm), ISH (improved skyline-based heuristic algorithm), CIBA and IAm (the modified intelligent search algorithm). The relative effectiveness of these algorithms were evaluated according to a performance measure – the ratio between the packing height returned by an algorithm and the height of an optimal solution. The cluster analysis performed in the fifth step consists of grouping a training data, more precisely a sample of instances randomly extracted from the entire problem instances (original and generated), into different classes of test problems based on their characteristics as calculated in the third step. During the last step, which points towards classification, decision trees based on gini index are employed to build a set of classification rules using the cluster labels as target variables and the characteristics from step 3 as conditions. As a side note, one could expect a different result by directly classifying the instances based on their characteristics **AND** best performing heuristics. This research direction is the focus in an ongoing project of the author. The performance of this solution if using gini index for decision tree classification reaches 91% for a maximum tree depth of 10.

2.3 Algorithm Selection for Combinatorial Packing Problems [3] *(Ioana Sitaru and Mădălina Răschip)*

As an incremental improvement with respect to the previous solution, this method considers the quality of a run as a trade-off between the runtime of the algorithm and how close to the optimum the obtained solution is. Therefore, both metrics are used in building the algorithm selection system. Starting from the input data, the feature set is constructed, meaning a set of 23 features that characterize the input instance are extracted. Some of them are newly encountered, such as the baseline strip height feature, which implies solving the packing using a shelf division to get an upper limit for the solution that can be obtained. This solution refers to the algorithm/heuristic space in terms of the algorithm portfolio, which consists of all solvers that candidate to be the best choice for a given instance. For building a robust algorithm selection system, the algorithms

should all solve the given problem, but none of them overruling the others for all types of instances. Therefore, the algorithm portfolio for this solution consists of five candidates: the greedy approach, the genetic algorithm, the binary tree heuristic, the priority heuristic, and the area heuristic. The criterion for selecting the best algorithm through machine learning techniques is split into runtime performance and closeness to optimtality. For the runtime prediction, a regression model is employed no matter of the algorithm. Among regression frameworks, the experimental results show that XGBoost is performing the best for our task, especially with additional hyperparameter tuning. For solution quality prediciton, the method relies on the (23) characteristics to build an ensemble of models and return the average predicted result. The regression models used are: ARDRegression, SVR, Random Forest, and Extra Trees. With a total of 103 correct selections from a total of 128 test instances, the runtime algorithm selection system recorded an accuracy of 80.46%. For the task of only selecting the algorithm with the best result, the labels matched decreased to 95 out of 128 instances. This leads to an accuracy of 74.21% on the test instances. The decrease can be explained by the distribution of the best performing algorithm on result selection which covers all five algorithms for different instance types, making the task more difficult. One final observation regarding this solution targets the need to incorporate more benchmark instances in order to offer more competitive performance.

2.4 Framework of algorithm portfolios for strip packing problem [4] *(Kamil Piechowiak, Maciej Drozdowski and Éric Sanlaville)*

This solution sheds light on a subtlety in the problem's formulation: the solving may proceed either with an algorithm or with a portfolio of algorithms **per instance**, the latter ending up in using the best combination constructed for the prediction stage. This method does not construct or make use of features/characteristics in the course of the machine learning task. Instead, a specific runtime limit is established and an ordered system composed of multiple algorithms (constrained to fit in time when adding the individuals costs for all algorithms) turns on, one by one, each of its components. Creating the system eventually comes down to solving a scheduling problem to do the match on the testing set, based on the central tendency in the training split for similar characteristics. Moreover, no algorithm is eliminated from the portfolio as long as there is an instance which the algorithm uniquely solves to the best (possible) objective value for some instance. Taking into account that the formalization of solutions such that a maximum time limit is included

in their representation, there are cases when a schedule for the unseen instances can be built easily, relying on the fact that we know the total work of the algorithms and no algorithm has runtime longer than the one in its definition. Moreover, if a portfolio comprises metaheuristics, then all these algorithms are simply suspended on the known limit. The metric this solution introduces for each algorithm is the number of *wins*, defined as the number of instances for which the algorithm built a solution of best quality. To give an insight on the performance of the current method, for validation instances of the same size as the training instances the smallest number of wins was 93 (out of 100). An immediate disadvantage of the *number of wins* metric is that some algorithm which is, e.g., the second best for all instances in the dataset, will be considered worse than some other algorithm which is always the worst with the exception of a single win. However, this situation is perfectly in line with the approach of building algorithm portfolios. Still, to comply with the classic methodologies, median and maximum relative distances from the best solution found are used as auxiliary algorithm performance measures. Unlike the other approaches, this solution feeds each instance into the prediction model and not common characteristics describing groups of instances, which might give potential to neural networks, for example, to extract particular, one-time, low-level features on their own.

3 Benchmark instances

Given the particular circumstances of a less intensively studied combinatorial problem, a quite large collection of benchmark instances is available at the following addresses: `https://www.euro-online.org/websites/esicup/data-sets/#1535972088188-55fb7640-4228`, `https://www.cs.put.poznan.pl/mdrozdowski/2sp-asp/` [6, 7]. As in the state-of-the-art methods, additional instances in order to artificially enlarge the dataset might be produced using specialized generators, such as **2DCPackGen** [5]. The second link already includes a couple of automatically generated instances. Both sources are the result of collecting and applying well known literature landmarks in packing problems, with respect to features such as the rectangle dimensions, the ratios given by their extremely small or large dimensions and the heterogeneity degree among the rectangles to frame, which are usually employed as specific input for the selection algorithms. The novel approaches presented in this paper (along with their parameter setting variations) make use of both literature instances and automatically generated instances, which

were found to be more conveniently organized for creating the training and testing environments in the second source mentioned.

The problem instances fed into the neural networks within the proposed solution (which is to be described in the rest of the current paper) are located in .txt files formatted as follows: the first row contains the total width of the bounding box, represented through an integer number, the second row depicts the number of rectangles to pack (which is obviously an integer), and the rest of the lines in the input files containing instances describe the width and the height, in this particular order, of each (smaller) rectangle, through integers. The solution described in the next sections proceeds with a dataset comprised of 1144 instances from the second source mentioned above, which were selected such that there is a high degree of diversity among the four characteristics proved to be essential to strip packing problems — the maximum aspect ratio of all items of an instance, the maximum area ratio of all pairs of items of an instance, the heterogeneity ratio, and the width ratio — but also such that there are at most 100 rectangles to pack and there are no instances exceeding the maximum width on any of the two dimensions (this last condition serves to the floorplanning and skyline heuristics which are detailed in Section 4.1) [2].

4 My novel approach: assembling neural networks in augmented feature spaces

This section details the main contribution in this paper, which consists of four implemented prediction systems, based on neural networks, among which two rely on assembling more classifiers belonging to the other two individual types [17]. Besides the machine learning algorithm, this paper contributes to the automatic algorithm selection research field by creating a dual algorithm portofolio, which correlates Strip Packing instances with both the best possible value for the height — obtained by at least one of the algorithms available in the portofolio — and (all the) the algorithms obtaining this value.

4.1 Algorithm portofolio

According to the way algorithm selection is adressed and universally perceived in dedicated literature, an algorithm portofolio is needed for the automated selection framework to be fitted on and then become capable of sustainable generalization. Since none of the existing studies on

the topic holds any references to a portofolio instance, a collection of unified, per-instance bridges between 1144 benchmark instances and 6 heuristic algorithms is created, under the following specifications.

The first stage assumes establishing a collection of algorithms for the instances to be in turn evaluated on. In the context of the present solution, this collection consists of the next 6 algorithms: priority guillotine constraint optimization algorithm, a genetic algorithm with previously finetuned parameters, the Steinberg heuristic algorithm, an heuristic system deciding upon flooplanning and skyline configurations, another heuristic system deceiding between hyper search and local search and a particle swarm optimization approach (also with previously finetuned parameters). Within the context of this paper, a heuristic system is defined as a very specific collection of algorithms derived from the same optimization principle, which are further considered as solely one technique during selection; among these similar optimization algorithm versions, the optimum selection within the collection is executed internally: for instance, the second mentioned heuristic system requires two individual runs for one instance, one for the hyper search method and one for the local search method, among which the minimum result is selected and appended to the portofolio under the tag of the heuristic system as a whole. The other heuristic system, internally deciding upon flooplanning and skyline configurations, chooses between 120 results, among which it was assesed that we rarely meet more than 2 configurations yielding the global minimum result in the case this technique as a whole is marked as optimum. A natural extension of the current solution would be to consider all 127 results individually, but for the frame of the nascent stage the current solution lies within, this approach is likely to introduce too much variance in the patterns we aim to infer.

The second stage of processing within the portofolio creation consists in automating the sequential execution of all algorithms for each instance, recording the height returned as integer output, on one side, and then postprocessing the results sequence for each instance in order to obtain the optimal algorithms as binary output, on the other side. The integer output is obtained by sequencing the results computed by each algorithm on a given instance, while the binary output encoding is the result of applying the min(imum) operator over the integer encoding and is similar to the traditional one-hot-encoding, except that multiple optimal algorithms are allowed for each instance. All the architectures described within Section 4.2 make exclusive use of the binary encoding in displaying predictions. Moreover, the ablation study in Section 5 will show

that neural networks perform poorly as means of automated selection if the integer encoding is employed. Section 4.3 will show a more statistically accurate methodology for handling the randomness of certain heuristics and distinguishing between accidental and consistent optimal behavior (Fig. 2).

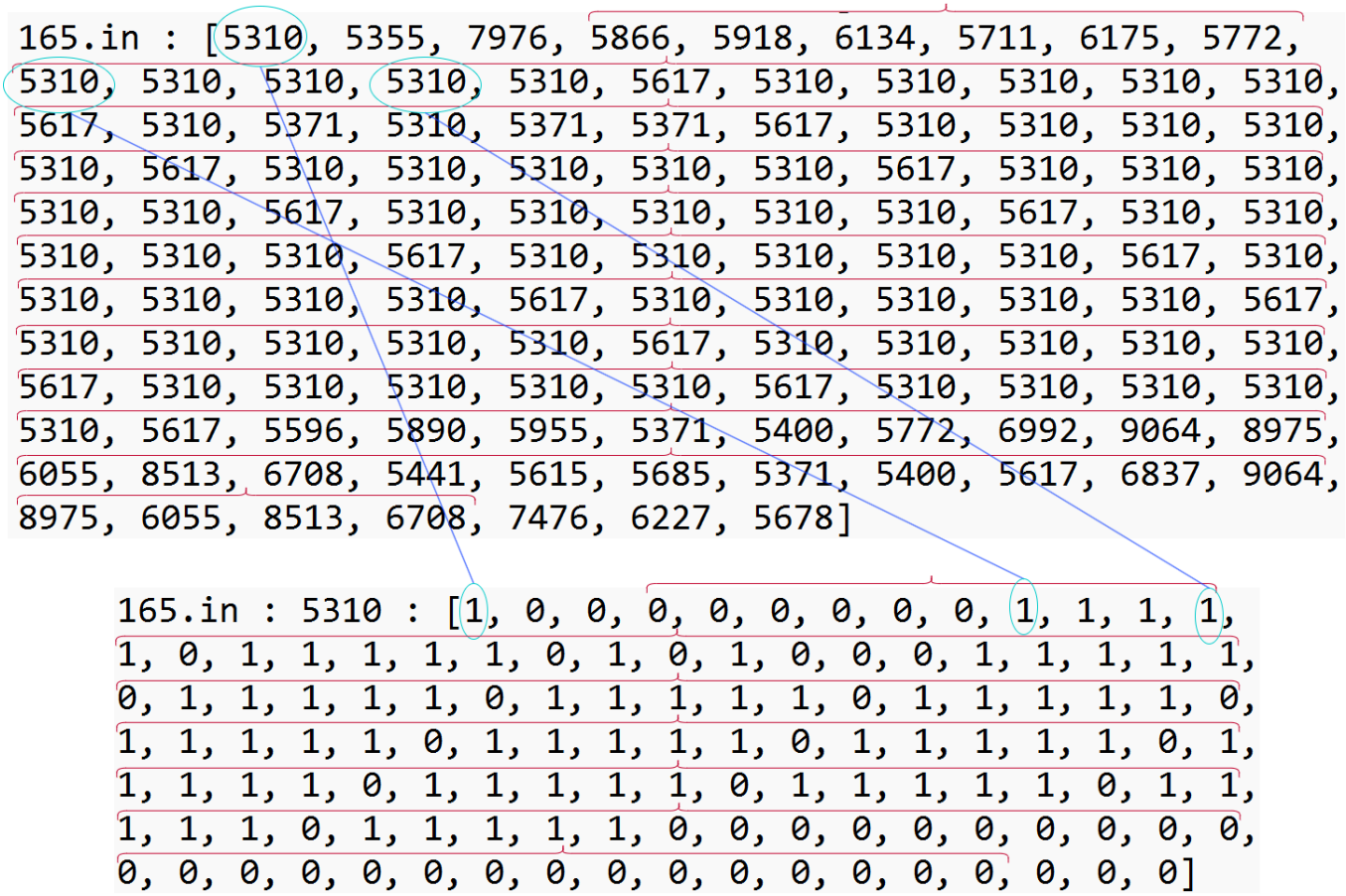

Figure 2: Illustrated equivalence between the ineteger output encoding and the binary output encoding.

All the mentioned algorithms are briefly described as follows, with regard to their corresponding GitHub implementations (and specific research papers, if that is the case).

The algorithm portofolio built as previously described represents the dataset for the machine learning applications which follow for presentation. In most of the test cases described within Section 5, the dataset is partitioned according to a $[0.87, 0.13]$ ratio, meaning a random subset comprising 87% of the instances is assigned for training and the rest of 13% is meant for validation / testing (within the machine learning framework we are building, there is no separation between validation and testing, since the parameter hypertuning and the accuracy prediction are performed

on the same data partition).

4.2 Design and implementation of the solutions proposed

Despite being the current lead in artificial intelligence tasks for a while now, neural networks are quite uncommon with respect to the automated algorithm selection proposed solutions so far, not to mention convolutional neural networks, which are quite rare while working with non-image and non-audio data in the first place (essentially, while working with data which is not headed for defining neighborhood areas over it) [18, 19]. To our best knowledge, cutting and packing problems have never been approached via neural networks. Consequently, this paper explores the opportunity of fitting and assembling neural networks in the limited environment provided by the reduced dataset.

For each instance a number of 16 base features are extracted, consisting in mathematical and statistical properties of mutual dependency relationships between the (small) rectangles and between the (small) rectangles and the strip: *max_aspect_ratio*, *max_area_ratio*, *heterogeneity_ratio*, *width_ratio*, *median*, *median_width*, *median_height*, *mean*, *mean_width*, *mean_height*, *variance*, *variance_width*, *variance_height*, *max_all*, *max_width*, *max_height*, each holding the associated significance in Table 1.

According to R.G. Rakotonirainy in [2], only the first four features mentioned above are proven to be practically sufficient in order to fully describe the data instances, in the context of clusterization. However, neural networks are generally more hungry for data and features in exchange for enhanced results and the algorithm selection framework makes no exception [19]. The 16 features above, plus the features resulting from the feature space augmentation technique which is presented within Section 4.3, make up the input of the neural network architectures in the machine learning algorithm. In other words, this is the semantic format of the input which the neural networks aim to connect with the binary encoded output. It goes without saying that the input for the neural network is standardized and is different from the input of the second stage in the portofolio creation, since the former turn the implicit content of the latter, which is raw and irregular (see Section 3), into explicit features.

For the machine learning solution 4 neural network configurations were implemented, among which the first one is defined as a fully-connected network with hill topology. In the context of this paper, hill topology refers to a sequence of layers which visibly increase and then decrease the

Table 1: Base features in the machine learning algorithms' inputs.

Feature	**Description**
max_aspect_ratio	the maximum ratio between one (small) rectangle's length along the larger side dimension and its length along the smaller side dimension (both rectangles are part of the instance)
max_area_ratio	the maximum ratio between two distinct rectangle areas
heterogeneity_ratio	the ratio between the total number of rectangles and the number of distinct types of items in an instance; two items are of the same type if they have identical smaller and larger side dimensions
width_ratio	the ratio between the width of the strip and the mean value of all (smaller and larger) item dimensions
median	the median of all (smaller and larger) item dimensions
median_width	the median of the first dimension (width) across all items
median_height	the median of the second dimension (height) across all items
mean	the mean of all (smaller and larger) item dimensions
mean_width	the mean of the first dimension (width) across all items
mean_height	the mean of the second dimension (height) across all items
variance	the variance of all (smaller and larger) item dimensions
variance_width	the variance of the first dimension (width) across all items
variance_height	the variance of the second dimension (height) across all items
max_all	the maximum length of any larger dimension across all items
max_width	the maximum length of any width dimension across all items
max_height	the maximum length of any height dimension across all items

number of neurons; the layers increasing the number of neurons will be further called multiplication layers, whereas the ones decreasing the number of neurons will be calles reduction layers; the more neurons added for the peak coming after one multiplication layer, or the more neurons removed for the abyss coming after one reduction layer, the steeper the slope of the hill and hopefully the greater the chances of embedding significance in the fed features. Therefore, besides relying on the Softmax activation function, the Adam optimizer and the Mean Squared Error (MSE) loss, the architecture of the first neural network consists of one layer connecting the input features to 100 neurons and another layer connecting these 100 neurons to 6 output neurons (Fig. 3).

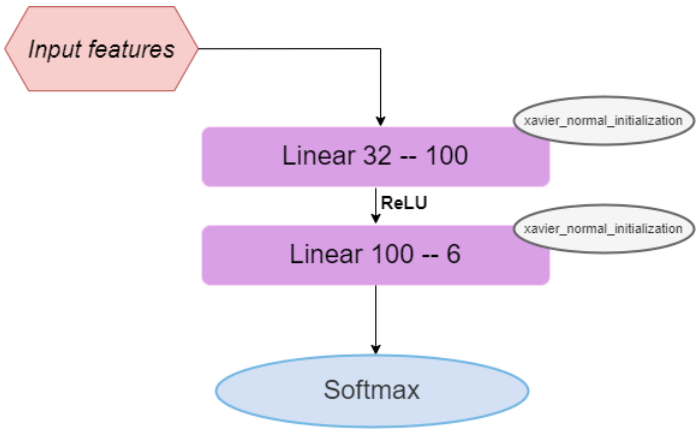

Figure 3: The first solution's architecture: simple fully-connected neural network with hill topology layer layout.

The second solution configuration belongs to a convolutional network defining a decoder-encoder architecture. Within the context of this paper, a decoder-encoder architecture refers to a sequence of two convolutional layers blocks, one for upsampling and one specifically dedicating to downsampling. Both hill topology and decoder-encoder architecture are newly introduces defining terms for artificially increasing the feature space via exponentially multiplying the number of neurons in order to reduce them to a nonlinear combination of essential (initial and supplementary) features afterwards. Therefore, the second neural network consists of four upsampling and four downsampling units, each potentially adding normalization or Dropout layers and ReLU or LeakyRelu activations, with a kernel size of $(4, 4)$ and a stride varying between 1 and 2, proportionally with the number of parameters in the layers traversal. Padding is only employed with a value of 1 for the units bridging the upsampling and the downsampling block, respectively (Fig. 4).

The third solution consists in assembling 10 neural networks of the first configuration and the fourth solution embeds 10 neural networks of the second kind in a convolutional ensemble. Given

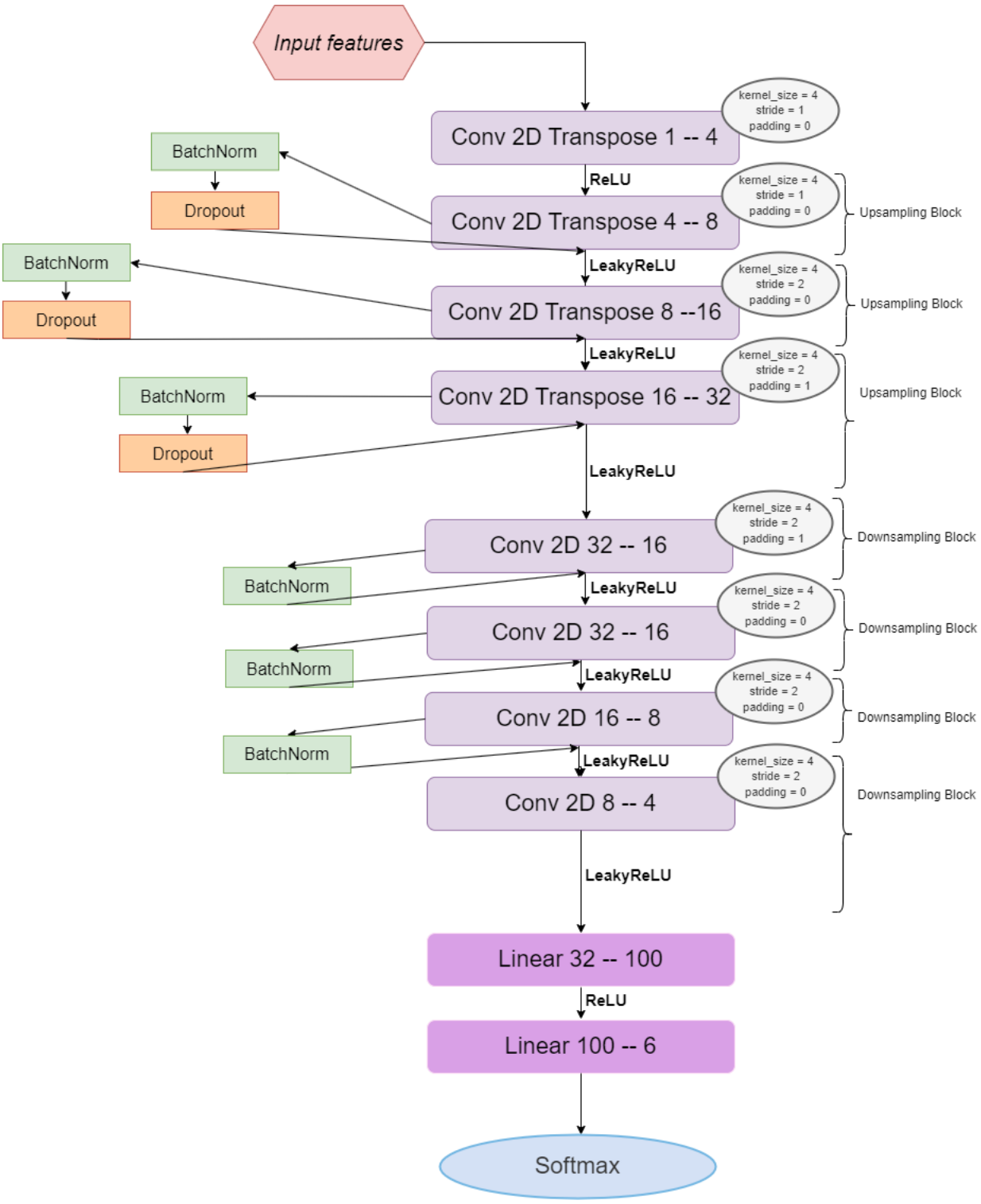

Figure 4: The second solution's architecture: convolutional neural network reversing the classical encoder-decoder architecture. Pay attention that the termination of this network is exactly the MLP network employed by the first solution.

that the neural networks' parameters are sufficient to provide diversity within the ensemble, the solution quality for these two configurations ultimately depends on the (un)biased and (un)stable behavior of each individual component [17].

4.3 Solution refining

Major improvements in the solution's performance are due to the feature space augmentation technique, which is inspired from image inpainting literature, as it is supposed to fill the void of some missing, even though anonymous, features [20]. The already existing features are complemented with new ones by sampling from a mixture of normal distributions, fitted on the base features. For this purpose the *GMM* algorithm is used, as its functioning mechanism lies exactly within the previously mentioned parameters: it is capable of arbitrarily sampling an additional block of features out of the mixture fitted. Since prospective refinements over the neural network's architectures are intended, investing in more features might represent a dramatic keypoint for pushing the accuracy score beyond 75%. Nevertheless, since in the *EM* space, which is the reverse of the original dataset space, the first dimension (equal to the number of instances) is a lot bigger than the second dimension (which is the number of features), few distributions need to be integrated in the gaussian mixture and their covariance matrices have to be diagonal, therefore corresponding to spherical (when all elements on the main diagonal have the same value) or elliptical distributions (when the elements on the main diagonal can take different values). Failing to follow these specifications would cause singular matrix errors wihtin the implementation stage. Moreover, special attention must be paid to the initialization procedure (so as to shift the mean value and the covariance matrix values away from the empirical mean and variance, but keep the parameters in line with them at the same time), otherwise it is likely that the implementation will lack bias at the latent variables' level throughout iterations, as the values computed with respect to the gaussian curves will be too close to 0 (or actual 0, depending on the numerical precision). For instance, one method could target initialization of the additional features (the number of additional features being established in advance) by randomly sampling from the mixture (with equal probability from each of the distributions), after the mean and covariance parameters have been initialized empirically for the initial variables only. Additionally, *GMM* was employed for a second time so that new instances are sampled from the clusters formed, this way fulfilling the purpose of complementing the database's structure itself instead of punctual features within the instances'

structure (Fig. 5).

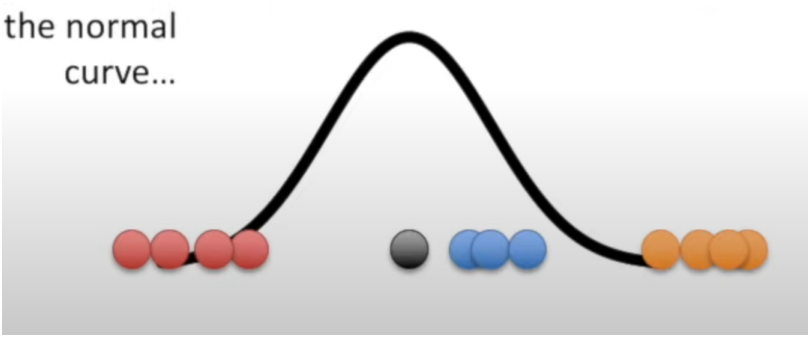

Figure 5: Sampling new features on the neural network's behalf using GMM.

The second refining resolution is applied to the algorithm portofolio creation stage and makes use of parametric and nonparametric statistical tests in order to sharpen the dataset's reliability. The first approach eliminates inconsistently winning algorithms during the second processing phase based on the ANOVA parametric test, while the second one filters the most statistically relevant ones for optimality gain using the Friedman nonparametric test. Both approaches are considered reliable themselves on a background of at least 30 repetitions of the second processing phase in the algorithm portofolio creation. This means that, for each instance, a sufficient number of runs for each algorithm (at least 30, according to the previous specification) is performed such that the algorithm set for that instance can be divided into reliable and unreliable algorithm subsets. Consequently, the realiable subset will be marked with 1 values and the unreliable one will be annotated with 0 values in the binary output sequence indicating the algorithms preferred. The way these two subsets are decided stands upon the execution of the statistical tests just mentioned. Multiple tests are applied with respect to each block of output sequences (which means multiple tests for each instance): firstly, one test is performed in order to conclude if there is statistical difference between the results mean (results has the meaning of heights here) associated with the algorithms (among the six) which recorded at least one optimal occurence during the runs. If the null hypothesis is rejected and, hence, there is statistical difference among the at least one-time optimal algorithms, the next tests concentrate on post-hoc analysis in order to establish which algorithms are on top. The total number of tests cannot be prescribed in advance, since the inequality relations between the algorithms' performance levels can take various outlines. Going deeper into the methodology of performing the tests remained, the pairwise instances identified as disimilar during post-hoc analysis are tested again for deciding upon the most optimal results

mean within each pair. These best algorithms are regrouped together and, again, one test is performed on them all so as to determine if there are statistical differences; if there are, post-hoc analysis is applied and the greatest heuristics are filtered. These three actions are repeated until the remaining algorithms are qualified as equally optimal by the common test. It is important to note that this refining procedure was only envisioned for one small partition of data in the experimental context of the current project, due to obvious time limitations [Algorithm 1].

Algorithm 1: A simplified pseudocode version of the second refining resolution, concerning the execution of statistical tests within the output composition phase.

```
Run all 6 algorithms 30 times for each instance and obtain optimal_outputs;
for each instance do
    repeat
        H_0 ← ANOVA (optimal_outputs);
        if H_0_status == rejected then
            optimal_outputs ← post-hoc_filtering(H_0_params, optimal_outputs);
        end
    until H_0_status == not rejected;
end
```

5 Ablation study: operating principles of the solutions proposed

This section studies the influence of modifying different architectural details and parameter values over the accuracy score, time, stability and over the natural interpretability of the system. Limited computational resources need to be taken into account as well, since the runs destined to the dataset creation process were performed on the personal machine, therefore they had to fit in the time windows alloted by the current term's schedule. The measured scores also depend on the stage of optimization at the moment the evaluation of a particular aspect was performed, but additional evidence has been gathered in the subsequent runs they are still topical for the problem considered anyway. Comparative evidence with respect to deep-rooted classification algorithms or other experimental algorithm selection strategies can also be gathered within the current section.

In this note, the creation of the portofolio took around 102 hours of running on the personal machine, which is a Lenovo Legion 5 Pro 16ARX8 laptop, equipped with AMD Ryzen 9 7945HX processor, 32GB of RAM memory and NVIDIA GeForce RTX 4070 graphical card.

As Table 3 illustrates, all solutions have comparable performance in terms of accuracy, the only difference residing in the number of epochs necessary to achieve this performance. However, as Table 2 shows, the accuracy is highly dependent on the output encoding and contrary to other machine learning approaches, neural network solutions perform better on the binary encoding and tend to be rather unprecise and unstable on the integer encoding version. Thus, a more diverse range of accuracy and loss metrics should be considered for the integer encoding, such as weighting the unit differences towards the optimal result with respect to a reasonable treshold.

Table 4 and Table 5 motivate the use of the feature space augmentation technique, as employing more features leads to significant improvements in the accuracy score. Certainly not all features can be mathematically or statiscally inferred, which is why it is relevant that artificial expansion at the feature level triggers performance spikes without needing to be in control of the generated features themselves.

As opposed to the features' significant influence on the output, the dataset size's evolution curve is way too smoth, as it can be deduced from Table 7. Consequently, this paper states that there is empirical evidence we should focus on solving the task for small datasets, as larger ones might be difficult to obtain or might not bring meaningful intakes.

As observed in Table 6, the linear configuration and the convolutional configuration reach similar performance in terms of solution quality and, even more, get confused about roughly the same instances. The two configurations owe their similarity to their exact same operating principle (upsampling and downsampling performed on linear features), only modeled differently, but also to the vague notion of neighborhood, barely existing on such features. Adding more epochs to the training stage, at least until 1000 is reached, does not add much to the models' performance, as suggested by Table 8. In fact, the maximum capacity of the two models is always attained during the first (maximum 200) epochs.

In order to place the present solution among landmarks in the field, the results are compared to the operating mode of two preeminent classifiers in working with reduced datasets: Random Forest and XGBoost, the latter being chosen as an ensemble classifier so as to fit the scenario of the last two configurations.

Table 2: Accuracy score comparison based on the output encoding for selection.

Criterion	**Integer output encoding**	**Binary output encoding**
Accuracy	33.54%	71.65%

Table 3: Accuracy and runtime comparison for single and ensemble neural networks.

Configuration	**Single**	**Ensemble**
Accuracy	71.65%	71.65%
Epochs	50%	20%

Table 4: Accuracy score improvement if employing more base features.

Configuration	4 **base features**	16 **base features**
Accuracy	62.11%	71.65%

Table 5: Accuracy score improvement if performing feature space augmentation.

Configuration	**Without feature expansion**	**With feature expansion**
Accuracy	71.65%	75.49%

Table 6: Accuracy and runtime comparison for the linear and convolutional configurations.

Configuration	**Linear**	**Convolutional**
Accuracy	75.49%	75.22%
Epochs	50%	150%

Table 7: Accuracy evidence on the dataset size.

Configuration	**Less instances**	**More instances**
Accuracy	62.11%	62.67%

Table 8: Accuracy evidence on the number of training epochs established.

Configuration	300 **epochs**	1000 **epochs**
Accuracy	75.49%	75.49%

Table 9: Accuarcy mean comparison with Random Forest and XGBoost classification engines.

Algorithm	**Neural networks**	**Random Forest**	**XGBoost**
Accuracy	70.37%	63.49%	67.24%

6 Discussion on results

As far as the current configuration joining multiple algorithms into one prediction thread is considered to be fixed and stable, it looks like the keypoint resides in producing cross-instance and cross-algorithm features for the neural network to generate more substantial connections between the instances and the optimal algorithms. It is also preferred for the feature expansion technique to show robustness in front of changes within the algorithm portofolio (such as adding heuristics to it or removing heuristics from it), since it does not depend on it in any way, but relates to the collected algorithms in order to measure its effectiveness when integrated within the proposed framework. Needless to say, augmenting the feature space visibly improves the accuracy for all the four solution candidates, in all versions experienced depending on the parameter setup, which makes the need for more features the obstructing obstacle standing in the way of the solution's simplicity; it seems that a simple neural network, comprising one or two linear layers, is able to perform really well on the algorithm selection task if provided with the right amount of features. Unfortunately, "per se" features for instances of many combinatorial problems are hard to deduce "with the naked eye" and no information background in the problem's (discovered or undiscovered) underlying premises, which is why the algorithm selection task would certainly benefit from the feature creation process being automated via generative techniques such as GMM or more modern approaches in the field.

Since the newly proposed decoder-encoder network architecture relies on the premise of having a (too) small number of features at the input layer's disposal, artificially augmenting the feature space would eliminate the need for such a specific architecture and expand the layer composition options amongst more generic and arbitrary combinations of layers. It is also quite straightforward that the ensembles of neural networks show the same quality performance as the neural networks standalone, which makes them a valid investment only in terms of running time and prediction stability, as previosuly stated. However, this leads us to the valuable conclusion that there is no bias in prediction; bias could have theoretically been introduced by any of the neural network's pa-

rameters, but ensembles showing no difference compared to the individual networks demonstrate that the networks' structure and functionality is resilient and consistent to the potential above mentioned changes in the portofolio.

7 Further work

Keeping the spirit of effective space exploration, future work might revolve around more modern techniques to sample new features in order to augment and expand the neural networks' inputs. Some of the ideas might involve generative adversarial networks, adapted to linear feature spaces with very light neighborhood dependencies, or an adaptation of the latest State-of-the-Art in generation of content, Stable Diffusion, vanilla model and hybrids. Moreover, a future version of the framework may encompass connections between best matching constructive heuristics and the preferred, not mutually exclusive, portofolio heuristics, by employing the former as additional features [1, 5]. This way a double heuristic examination would be applied to the problem data and perhaps the constructive heuristics would turn into valuable features with respect to the same selection target as before [9]. Again, feature expansion can be applied to these last features, separately (which would result in two generative techniques to be run, this is useful in case this particular type of features behaves better in the context of a different generative framework than the rest of the features) or together with the mathematical ones we worked with up to this point.

As far as the portofolio creation stage is concerned, the candidate algorithm space could be enlarged by adding new heuristics, since this paper leaves at least a couple of constraint programming and combinatorial optimization algorithms unexplored [11, 13]. This might further lead to an in-depth study on the main method's robustness about changes in the portofolio configuration, such as replacing, adding or removing algorithms. Studying the impact on accuracy of the dataset size could also be instructive as further practice, so as to spot more generic tendencies in the connections between algorithms and features and evaluate the joint classification pattern (which includes all algorithms as output in the same neural network traversal for every instance) in a more informed and reliable setting.

Lastly, the present solution may benefit from a runtime prediction appendix, in order to be able to weight algorithms as trade-offs between accuracy and time performance and choose the most suitable method accordingly. Since most of the current State-of-the-Art systems employ

fairly interesting quality measures, such as the number of wins — which is conditioned by the actual numerical result prediction — the present solution can be changed to work with custom loss functions so that its performance is evaluated from multiple perspectives and assigned to potential practical applications[1].

8 Conclusions

Taking into account the benefits and limitations of each variant presented, we are able to understand why there are multiple approaches out there for a unique proposed problem to be solved. While neural networks have definitely taken the lead in the artifficial intelligence research fields, there is a lot of effort to put in the configuration details and overtaking limitations of the decision framework the creator decides upon, especially when the dataset is quite limited itself, both in terms of features and instances. Given the results of all experiments performed, there is enough emipirical evidence to conclude that dataset preprocessing is the insufficiently praised backbone of the proposed algorithms' performance.

In complete agreement with the ensembles being built upon the principle which states that "unity is strength", experiments have shown the last two schemas among the four described demonstrate desirable behavior in fewer running epochs than the individual neural networks taken aside. Since a universal set of features, sensitive to all heuristics chosen within the portofolio, is crucial for the accuracy of the built system, the whole working framework described as a contribution in this paper shows sensitivity to additional parameters, specific to gaussian mixture models, such as the number of distributions in the mixture generating features and the shape of the covariance matrix. Given the fact that Random Forest and XGBoost form a trustworthy benchmark for comparing performance on small datasets and neural networks have been evaluated to reach higher accuracy than these decision trees approaches in a variety of parameter setup and data processing scenarios, it can be boldly concluded that a favorable progress track in the research to come might push neural networks to a future State-of-the-Art status for the task of automated algorithm selection.

Eventually, there is room for experiencing with methodical parameter variations and in-

[1]For instance, one can measure performance by matching all the optimal algorithms for an instance, where there is more than one such algorithm, instead of only aiming at predicting one algorithm among the preferred ones. This change might give an insight into the framework's versatility and its potential to be implemented in a dynamic industry scenario.

genious hybridizations in order to find even more effective systems suiting the already known properties and behavior of the problem statement — towards finding optimal solutions throughout variously shaped search spaces — but also more accurate systems serving for the discovery of new significant influences in the feature space which the problem statement provides.

9 Appendix: illustrating keypoints in the state-of-the-art algorithms

Feature	Definition	Feature	Definition
Area-based group			
areacomp	The ratio between the projected strip area and rectangles areas, influenced by the ratio (between percentiles or quartiles measures) and composition (between the sum of larger and smaller measures) variables.	*areastats*	The ratio between the projected strip area and the rectangles' area, based on variables characterized by classical statistical measures (mean, median, and standard deviation).
Perimeter-based group			
perimcomp	The ratio between the strip perimeter and rectangles perimeters, influenced by ratio and composition variables.	*perimstats*	The ratio between the strip perimeter and rectangles perimeters, based on variables characterized by classical statistical measures.
Dimensions-based group			
dimcomp	The average dimension of rectangles compared to the strip width, influenced by ratio and composition variables.	*dimstats*	The average dimension of rectangles compared to the strip width, based on variables characterized by classical statistical measures.
Width dimension-based group			
widthdimcomp	Size of the largest rectangles dimension compared to the strip width, influenced by ratio and composition variables.	*widthdimstats*	Size of the largest rectangles dimension compared to the strip width, based on variables characterized by classical statistical measures.
Proportion-based group			
propcomp	The proportion between the strip and rectangle's dimensions, influenced by ratio and composition variables.	*propstats*	The proportion between the strip and rectangles dimensions, based on variables characterized by classical statistical measures.
Other group			
n^{pi}	The total number of rectangles.	*objdimratio*	The number of times that the strip lower bound is bigger than the strip width.
coefficient	Average rectangles' dimensions values.	*itdimratio*	The number of times that the maximum rectangles' dimension is bigger than the minimum rectangles' dimensions.
heterog	The proportion of different rectangles.	*maxcoefficient*	10 % larger rectangles dimensions values.
heterognt	The proportion of different rectangles with more than one rectangle.	*mincoefficient*	10 % smaller rectangles dimensions values.
difcoefficient	The total number of different rectangles dimensions.		

Figure 6: Features' definition examples.

	BL						BLF					
	CR	**SA**	**DR**	**RW**	**GA**	**TS**	**CR**	**SA**	**DR**	**RW**	**GA**	**TS**
CR	0	0	0	0	0	0	0	0	0	0	0	0
SA	0	0	0	0	0	0	0	0	0	0	0	0
DR	0	0	2	1	0	0	0	0	0	0	0	0
RW	2	0	2	12	4	5	1	1	6	10	4	5
GA	0	0	0	0	0	0	0	0	0	0	0	0
TS	24	22	470	263	178	1763	19	72	411	245	175	1779
sensitivity	0.000	0.000	0.004	0.043	0.000	0.997	0.000	0.000	0.000	0.039		0.997
specificity	1.00	1.00	1.00	0.995	1.00	0.023	1.00	1.00	1.00	0.993	1.00	0.023
prevalence	0.009	0.008	0.172	0.100	0.066	0.643	0.007	0.027	0.153	0.093	0.066	0.654
DR	0.000	0.000	0.001	0.004	0.004	0.642	0.000	0.000	0.000	0.004	0.000	0.652
DP	0.000	0.000	0.001	0.009	0.009	0.990	0.000	0.000	0.000	0.010	0.000	0.990
BA	0.500	0.500	0.502	0.519	0.519	0.510	0.500	0.500	0.500	0.516	0.500	0.510

Figure 7: Confusion matrix and performance metrics for BL and BLF in State-of-the-Art no 1, targeted for further work.

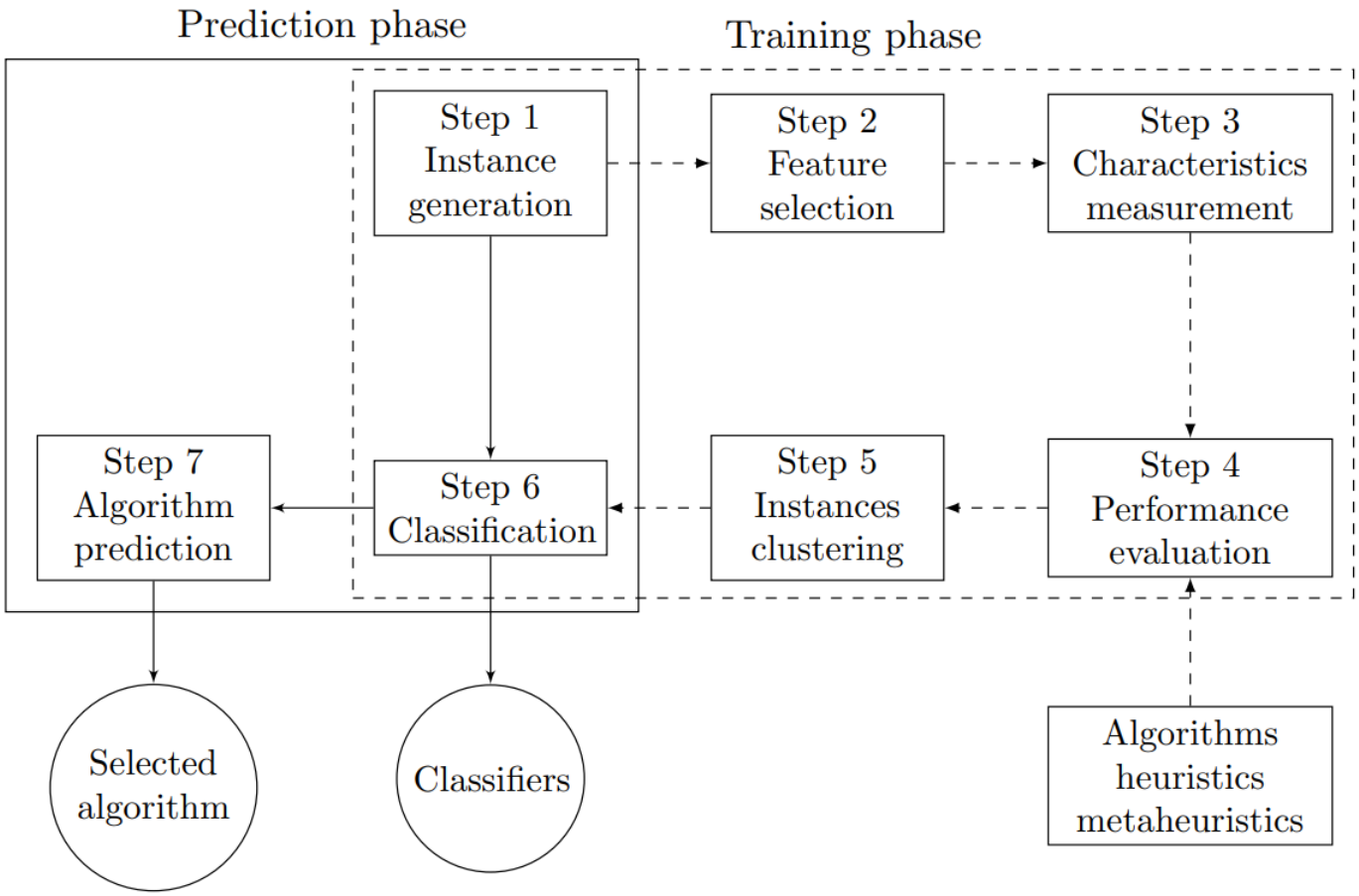

Figure 8: The base framework for automated packing algorithm selection, inspired from State-of-the-Art no 2.

Decision tree methods with	Gini index	Information gain
Maximum depth of the tree = 5	70%	69.6%
Maximum depth of the tree = 8	80%	75.4%
Maximum depth of the tree = 10	91%	90%

Figure 9: Classification accuracy of the various decision tree methods when applied to the training datasets, aimed for comparison with the proposed contribution.

References

[1] A. N. Júnior, J. Siluk, M. Francescatto, G. Stieler and D. Disconzi, "A framework to select heuristics for the rectangular two-dimensional strip packing problem," Expert Systems with Applications, vol. 213, Part C, March 2023.

[2] R. G. Rakotonirainy, "A machine learning approach for automated strip packing algorithm selection," Orion, vol. 36, December 2020.

[3] I. Sitaru and M. Răschip, " Algorithm selection for combinatorial packing problems," 2022 IEEE Congress on Evolutionary Computation, July 2022.

[4] C. Piechowiak, M. Drozdowski and É. Sanlaville, "Framework of algorithm portfolios for strip packing problem," Computers & Industrial Engineering, vol. 172, Part A, 2022.

[5] E. Silva, J.F. Oliveira and G. Wascher, "2DCPackGen: A problem generator for two-dimensional rectangular cutting and packing problems. European Journal of Operational Research," European Journal of Operational Research, vol. 237, September 2014.

[6] The Association of European Operational Research Societies, "ESICUP: Cutting and Packing," Available: `https://www.euro-online.org/websites/esicup/data-sets/#1535972088188-55fb7640-4228` [Accessed: May 4, 2024].

[7] K. Piechowiak, M. Drozdowski and É. Sanlaville, "Resources for Algorithm Selection for 2D Strip Packing Problem," Available: `https://www.cs.put.poznan.pl/mdrozdowski/2sp-asp/` [Accessed: May 4, 2024].

[8] RipperJ, "2DStripPacking", GitHub, Available: https://github.com/RipperJ/2DStripPacking [Accessed: May 28, 2024].

[9] D. Zhang, L. Shi, S. C. H. Leung and T. Wu, "A priority heuristic for the guillotine rectangular packing problem," Information Processing Letters, vol. 116, January 2016.

[10] Mxbonn, "About Strip-Packing," GitHub, Available: https://github.com/Mxbonn/strip-packing [Accessed: May 28, 2024].

[11] G. Wäscher, H. Haußner and H. Schumann, "An improved typology of cutting and packing problems," European Journal of Operational Research, vol. 183, December 2007.

[12] S. L. Riccardo, "Solving Strip Packing Problem with Genetic Algorithms," GitHub, Available: https://github.com/SebaRiccardo/Strip_Packing_GA [Accessed: May 28, 2024].

[13] A. Steinberg, "A Strip-Packing Algorithm with Absolute Performance Bound 2," SIAM Journal on Computing, vol. 26, 1997.

[14] yzdxdydz, "Steinberg's algorithm for the Strip Packing Problem with two heuristics," GitHub, Available: https://github.com/yzdxdydz/StripPacking?tab=readme-ov-file [Accessed: May 28, 2024].

[15] A. Aleiferis, "A hyper-heuristic for solving variants of the 2D bin packing problem," GitHub, Available: https://github.com/AlkiviadisAleiferis/hyperpack?tab=readme-ov-file [Accessed: May 28, 2024].

[16] maramire, "SP_PSO," GitHub, Available: https://github.com/maramire/SP_PSO [Accessed: May 28, 2024].

[17] T. Abe, E. K. Buchanan, G. Pleiss, R. Zemel and J. P. Cunningham, "Deep Ensembles Work, But Are They Necessary?," NeurIPS 2022, March 2022.

[18] E. Zvornicanin, "Convolutional Neural Network vs. Regular Neural Network," Baeldung, March 2024.

[19] I. Dagan, R. Vainshtein, G. Katz and L. Rokach, "Automated algorithm selection using meta-learning and pre-trained deep convolution neural networks," Information Fusion, vol. 105, May 2024.

[20] S. Ciobanu and L. Ciortuz, "Semantic Image Inpainting via Maximum Likelihood," 2020 22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), September 2020.