# Automated Algorithm Selection: Experiments on 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 all and every instance, does not exist. This paper investigates state-of-the-art approaches in automated algorithm selection, while tacking the 2-dimensional strip packing problem (2SP) as experimental framework. 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 State-of-the-Art

### 1.1 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.

## 1.2 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 rectange, 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.

## 1.3 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.

## 1.4 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.

## 2 Benchmark instances

A large collection of benchmark instances is available at the following addresses: `https://www.euro-online.org/websites/esicup/data-sets/#1535972088188-55fb7640-4228`, `ttps://www.cs.put.poznan.pl/mdrozdowski/2sp-asp/`. 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.

# 3 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 1: Features' definition in solution 1.

| | 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 2: Confusion matrix and performance metrics for BL and BLF in solution 1.

# 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. Raschip, " 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.

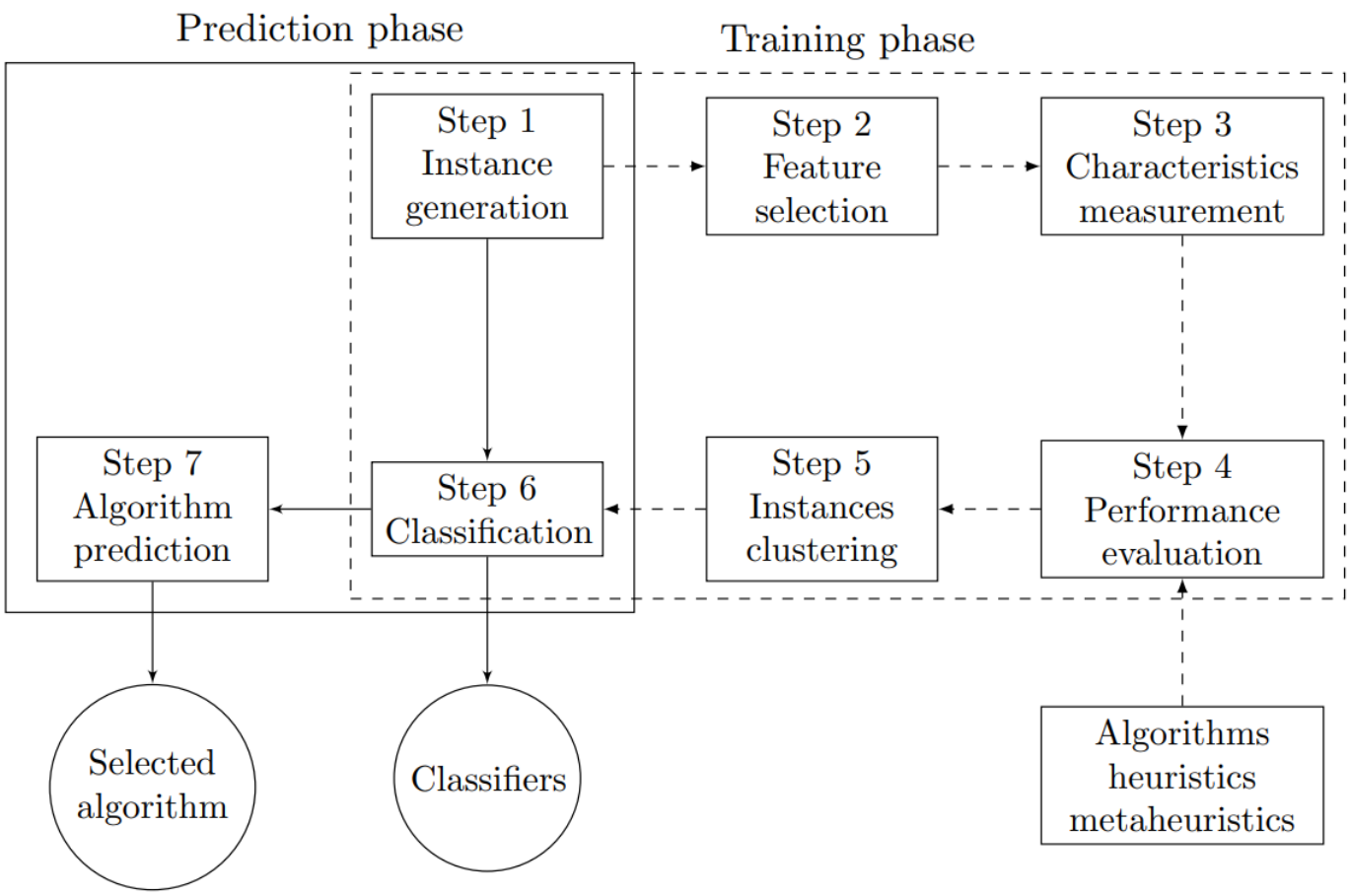


Figure 3: The framework for automated packing algorithm selection in solution 2.

| **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 4: Classification accuracy of the various decision tree methods when applied to the training data sets in solution 2.