Title: Algorithm selection

URL: https://en.wikipedia.org/wiki/Algorithm_selection

PageID: 50773876

Categories: Category:Constraint programming, Category:Machine learning

-----

Algorithm selection (sometimes also called per-instance algorithm selection or offline algorithm selection) is a meta- algorithmic technique to choose an algorithm from a portfolio on an instance-by-instance basis. It is motivated by the observation that on many practical problems, different algorithms have different performance characteristics. That is, while one algorithm performs well in some scenarios, it performs poorly in others and vice versa for another algorithm. If we can identify when to use which algorithm, we can optimize for each scenario and improve overall performance. This is what algorithm selection aims to do. The only prerequisite for applying algorithm selection techniques is that there exists (or that there can be constructed) a set of complementary algorithms.

Definition

Given a portfolio P {\displaystyle {\mathcal {P}}} of algorithms A ∈ P {\displaystyle {\mathcal {A}}\in {\mathcal {P}}} , a set of instances i ∈ I {\displaystyle i\in {\mathcal {I}}} and a cost metric m : P × I → R {\displaystyle m:{\mathcal {P}}\times {\mathcal {I}}\to \mathbb {R} } , the algorithm selection problem consists of finding a mapping s : I → P {\displaystyle s:{\mathcal {I}}\to {\mathcal {P}}} from instances I {\displaystyle {\mathcal {I}}} to algorithms P {\displaystyle {\mathcal {P}}} such that the cost ∑ i ∈ I m (s (i) , i) {\displaystyle \sum _{i\in {\mathcal {I}}}m(s(i),i)} across all instances is optimized.

Examples

Boolean satisfiability problem (and other hard combinatorial problems)

A well-known application of algorithm selection is the Boolean satisfiability problem . Here, the portfolio of algorithms is a set of (complementary) SAT solvers , the instances are Boolean formulas, the cost metric is for example average runtime or number of unsolved instances. So, the goal is to select a well-performing SAT solver for each individual instance. In the same way, algorithm selection can be applied to many other N P {\displaystyle {\mathcal {NP}}} -hard problems (such as mixed integer programming , CSP , AI planning , TSP , MAXSAT , QBF and answer set programming). Competition-winning systems in SAT are SATzilla, 3S and CSHC

Machine learning

In machine learning , algorithm selection is better known as meta-learning . The portfolio of algorithms consists of machine learning algorithms (e.g., Random Forest, SVM, DNN), the instances are data sets and the cost metric is for example the error rate. So, the goal is to predict which machine learning algorithm will have a small error on each data set.

Instance features

The algorithm selection problem is mainly solved with machine learning techniques. By representing the problem instances by numerical features f {\displaystyle f} , algorithm selection can be seen as a multi-class classification problem by learning a mapping f i ■ A {\displaystyle f_{i}\mapsto {\mathcal {A}}} for a given instance i {\displaystyle i} .

Instance features are numerical representations of instances. For example, we can count the number of variables, clauses, average clause length for Boolean formulas, or number of samples, features, class balance for ML data sets to get an impression about their characteristics.

Static vs. probing features

We distinguish between two kinds of features:

Static features are in most cases some counts and statistics (e.g., clauses-to-variables ratio in SAT). These features ranges from very cheap features (e.g. number of variables) to very complex features (e.g., statistics about variable-clause graphs).

Probing features (sometimes also called landmarking features) are computed by running some analysis of algorithm behavior on an instance (e.g., accuracy of a cheap decision tree algorithm on an ML data set, or running for a short time a stochastic local search solver on a Boolean formula). These feature often cost more than simple static features.

Feature costs

Depending on the used performance metric m {\displaystyle m} , feature computation can be associated with costs.

For example, if we use running time as performance metric, we include the time to compute our instance features into the performance of an algorithm selection system.

SAT solving is a concrete example, where such feature costs cannot be neglected, since instance features for CNF formulas can be either very cheap (e.g., to get the number of variables can be done in constant time for CNFs in the DIMACs format) or very expensive (e.g., graph features which can cost tens or hundreds of seconds).

It is important to take the overhead of feature computation into account in practice in such scenarios; otherwise a misleading impression of the performance of the algorithm selection approach is created. For example, if the decision which algorithm to choose can be made with perfect accuracy, but the features are the running time of the portfolio algorithms, there is no benefit to the portfolio approach. This would not be obvious if feature costs were omitted.

Approaches

Regression approach

One of the first successful algorithm selection approaches predicted the performance of each algorithm m ^ A : I → R {\displaystyle {\hat {m}}_{\mathcal {A}}:{\mathcal {I}}\to \mathbb {R} } and selected the algorithm with the best predicted performance a r g min A ∈ P m ^ A (i) {\displaystyle arg\min _{{\mathcal {A}}\in {\mathcal {P}}}{\hat {m}}_{\mathcal {A}}(i)} for an instance i {\displaystyle i} .

Clustering approach

A common assumption is that the given set of instances I {\displaystyle {\mathcal {I}}} can be clustered into homogeneous subsets

and for each of these subsets, there is one well-performing algorithm for all instances in there.

So, the training consists of identifying the homogeneous clusters via an unsupervised clustering approach and associating an algorithm with each cluster.

A new instance is assigned to a cluster and the associated algorithm selected.

A more modern approach is cost-sensitive hierarchical clustering using supervised learning to identify the homogeneous instance subsets.

Pairwise cost-sensitive classification approach

A common approach for multi-class classification is to learn pairwise models between every pair of classes (here algorithms)

and choose the class that was predicted most often by the pairwise models.

We can weight the instances of the pairwise prediction problem by the performance difference between the two algorithms.

This is motivated by the fact that we care most about getting predictions with large differences correct, but the penalty for an incorrect prediction is small if there is almost no performance difference.

Therefore, each instance i {\displaystyle i} for training a classification model A 1 {\displaystyle {\mathcal {A}}_{1}} vs A 2 {\displaystyle {\mathcal {A}}_{2}} is associated with a cost | m (A 1 , i) − m (A 2 , i) | {\displaystyle |m({\mathcal {A}}_{1},i)-m({\mathcal {A}}_{2},i)|} .

Requirements

The algorithm selection problem can be effectively applied under the following assumptions:

The portfolio P {\displaystyle {\mathcal {P}}} of algorithms is complementary with respect to the instance set I {\displaystyle {\mathcal {I}}} , i.e., there is no single algorithm A ∈ P {\displaystyle {\mathcal {A}}\in {\mathcal {P}}} that dominates the performance of all other algorithms over I {\displaystyle {\mathcal {I}}} (see figures to the right for examples on complementary analysis).

In some application, the computation of instance features is associated with a cost. For example, if the cost metric is running time, we have also to consider the time to compute the instance features. In such cases, the cost to compute features should not be larger than the performance gain through algorithm selection.

Application domains

Algorithm selection is not limited to single domains but can be applied to any kind of algorithm if the above requirements are satisfied.

Application domains include:

hard combinatorial problems: SAT , Mixed Integer Programming , CSP , AI Planning , TSP , MAXSAT , QBF and Answer Set Programming

combinatorial auctions

in machine learning, the problem is known as meta-learning

software design

black-box optimization

multi-agent systems

numerical optimization

linear algebra, differential equations

evolutionary algorithms

vehicle routing problem

power systems

For an extensive list of literature about algorithm selection, we refer to a literature overview.

Variants of algorithm selection

Online selection

Online algorithm selection refers to switching between different algorithms during the solving process. This is useful as a hyper-heuristic . In contrast, offline algorithm selection selects an algorithm for a given instance only once and before the solving process.

Computation of schedules

An extension of algorithm selection is the per-instance algorithm scheduling problem, in which we do not select only one solver, but we select a time budget for each algorithm on a per-instance base. This approach improves the performance of selection systems in particular if the instance features are not very informative and a wrong selection of a single solver is likely.

Selection of parallel portfolios

Given the increasing importance of parallel computation,

an extension of algorithm selection for parallel computation is parallel portfolio selection, in which we select a subset of the algorithms to simultaneously run in a parallel portfolio.

External links

Algorithm Selection Library (ASlib)

Algorithm selection literature

References