Label Ranking in Case-Based Reasoning*

Klaus Brinker and Eyke Hüllermeier
Department of Mathematics and Computer Science
Marburg University, Germany
{brinker,eyke}@informatik.uni-marburg.de

Abstract

The problem of label ranking has recently been introduced as an extension of conventional classification in the field of machine learning. In this paper, we argue that label ranking is an amenable task from a CBR point of view and, in particular, is more amenable to supporting case-based problem solving than standard classification. Moreover, by developing a case-based approach to label ranking, we will show that, the other way round, concepts and techniques from CBR are also useful for label ranking. In addition to an experimental study in which case-based label ranking is compared to conventional nearest neighbor classification, we present an application in which label ranking is used for node ordering in heuristic search.

1 Introduction

As a generic problem solving methodology, case-based reasoning (CBR) has already been applied successfully for various types of problems [21]. An especially simple yet relevant problem class for CBR is *prediction*, including classification (predicting one among a finite set of class labels) and regression (predicting a numerical output) as special cases. In this context, CBR overlaps with the field of machine learning and is typically referred to as case-based, instance-based, or memory-based learning [20, 2, 1]. The core of case-based learning algorithms is built upon the nearest neighbor estimation principle [7].

From a CBR point of view, prediction is arguably one of the least complex problem types, mainly because the crucial subtask of *adaptation* is not an intricate issue. In

*Draft of a paper to appear in: M. Richter and R. Weber (eds.). Proceedings ICCBR–07, 7th International Conference on Case-Based Reasoning, pages 31–40, Belfast, 2007.

fact, the adaptation of previous solutions, retrieved from a case library, to the current problem at hand is still one of the most challenging steps of a CBR process and quite difficult to automate [9]. For prediction problems, however, adaptation can be done in a rather straightforward way: In the case of regression, one can hardly do better than deriving an average of the k nearest neighbors' outputs (in which the neighbors are weighted according to their similarity to the query [17, 4]). In the case of classification, the "solution space" is given by the finite set of class labels, and the classes of the nearest neighbors are typically combined through majority voting.

Despite the fact that (nearest neighbor) classification methods have been used extensively in the CBR field, e.g., for problems such as diagnosis, one may argue that the classification framework in its standard form is not fully satisfactory from a *problem solving* point of view, simply because problem solving usually goes beyond predicting a single solution:

- A first crucial problem is that a simple classification does not imply a possible course of action in the case where it fails: If the classification is wrong, the problem is not yet solved, so the question is how to continue.
- Besides, in the case of failure, conventional classification does not offer a means to properly *learn* from the unsuccessful trial, because a problem in conjunction with a suboptimal solution is not an "example" in the sense of supervised learning; and even if an optimal solution is eventually found, the corresponding problem/solution pair will usually not comprise the complete experience (e.g., differences in the quality of suboptimal solutions) that has been gathered in the course of the problem solving episode.

To avoid these problems, we propose a (case-based) approach to *label ranking* and elaborate on its application in case-based reasoning. Label ranking is an extended classification task that has recently been studied in machine learning. The goal in label ranking is to predict a complete ranking of all class labels instead of only a single class (top-label). Obviously, a prediction of that kind can be very useful in CBR. For example, it suggests a simple problem solving strategy, namely a trial and error search process which successively tests the candidate solutions until the problem has been solved. Needless to say, to be effective, this strategy presupposes a solution space in the form of a finite set of small to moderate size (just like classification learning itself).

To illustrate, consider a fault detection problem which consists of identifying the cause for the malfunctioning of a technical system. Moreover, suppose that a (case-based) learning system is used to predict the true cause, e.g., on the basis of certain sensor

measurements serving as input attributes (see, e.g., [3] for an application of that type). A prediction in the form of a label ranking then dictates the complete order in which the potential causes (which play the role of labels) should be searched.

The remainder of the paper is organized as follows: The problem of label ranking and its relation to CBR are discussed in Section 2. In Section 3, we present a case-based approach to label ranking which is an extension of nearest neighbor classification.[1] Section 4 is devoted to experimental studies in which our case-based label ranking method is compared to conventional nearest neighbor classification. Moreover, this section elaborates on the idea of using case-based label ranking for node ordering in heuristic search. Finally, Section 5 gives a summary and concludes the paper.

2 Label Ranking and CBR

In label ranking, the problem is to learn a mapping from instances $\mathbf{x}$ of an instance space $\mathcal{X}$ to rankings $\succ_x$ (total strict orders) over a finite set of labels

$$\mathcal{L} = \{\lambda_1, \lambda_2 \ldots \lambda_c\},$$

where $\lambda_i \succ_x \lambda_j$ means that, for the instance $\mathbf{x}$, label λ_i is placed ahead of λ_j; in this case, we shall also say that λ_i is preferred to λ_j (by $\mathbf{x}$). As mentioned earlier, in the context of CBR, instances correspond to problems and labels correspond to candidate solutions. This should be kept in mind, since we shall subsequently use both the CBR and the machine learning terminology.

A ranking over $\mathcal{L}$ can be represented by a permutation τ of $\{1 \ldots c\}$, where $\tau(i)$ denotes the position of the label λ_i; thus, $\lambda_i \succ_x \lambda_j$ iff $\tau(i) < \tau(j)$. The target space of all permutations over c labels will subsequently be referred to as $\mathcal{S}_c$.

As training data, a label ranking algorithm can refer to a number of example instances together with different types of information regarding their preference for labels. Since this point is important for CBR, we will discuss it in this specific context.

2.1 Training Data in Label Ranking

Suppose that, in the course of a problem solving process in CBR, a subset of all candidate solutions has been tried to solve a query problem $\mathbf{x}_0$. What can be learned from these trials, and what experiences can be memorized? One type of experience concerns

[1] Parts of this section can also be found in the companion paper [6].

the suitability of individual candidate solutions λ_i: Such candidates may be feasible or acceptable as a solution or not. Subsequently, we shall refer to this kind of distinction as an *absolute preference*.

Another type of experience concerns *relative preferences*: As soon as two alternatives λ_i and λ_j have been tried as solutions for $\mathbf{x}_0$, these two alternatives can be compared and, correspondingly, either a preference in favor of one of them or an indifference can be expressed:

$$\lambda_i \succ_{x_0} \lambda_j \quad \text{or} \quad \lambda_i \prec_{x_0} \lambda_i \quad \text{or} \quad \lambda_i \sim_{x_0} \lambda_j.$$

Again, these preferences can be memorized and utilized for future problem solving. For example, a preference $\lambda_i \succ_{x_0} \lambda_j$ clearly holds if λ_j turned out to be unacceptable, while λ_i was found to be acceptable as a solution. However, even if both alternatives were acceptable, or both unacceptable, one may state that one of them is still better than the other one (e.g., because it is less expensive). Indeed, more often than not, there will be more than one acceptable solution, even though not all of them will be equally preferred.

To exploit experience of the above type, a CBR system may store a problem along with absolute and relative preferences in a case library. Thus, a *case* may look a follows:

$$\left\langle \mathbf{x}, ([\lambda_2, \lambda_4]^+, [\lambda_1, \lambda_6]^-, \lambda_4 \succ \lambda_2, \lambda_1 \sim \lambda_6) \right\rangle. \tag{1}$$

The meaning of this example is that λ_2 and λ_4 are acceptable solutions for the problem $\mathbf{x}$, λ_1 and λ_6 are unacceptable, λ_4 is preferred to λ_2, and indifference holds between λ_1 and λ_6. Note that a case such as (1) represents only partial preference information about labels; for example, nothing is known about λ_3, perhaps because it has not been tried as a solution for $\mathbf{x}$.

As will be explained in more detail later on, cases of the form (1) correspond to the (most general) type of examples that label ranking algorithms can learn from. With regard to gathering experiences from a problem solving episode, label ranking is hence more flexible and expressive than conventional classification. In fact, the standard type of example in classification learning is a tuple consisting of an instance and an associated class or, using CBR terminology, a problem with its "correct" solution. Comparing these two approaches, label ranking clearly exhibits the following advantages:

- Label ranking can also learn from unsuccessful trials; in contrast, the information that a certain label is not the correct class is not directly utilizable by standard classification methods (unless, of course, in the case of binary problems with only two classes).

- Even if the problem was eventually solved, label ranking can learn more than the optimal solution, because it additionally makes use of the preferences between the alternatives that have been tried before.

Even though conventional classification has already been extended in one way or the other, the aforementioned limitations essentially persist. For example, in *multi-label classification*, an instance may belong to more than one class or, stated differently, a problem may have more than one solution. So, an example is a tuple consisting of an instance $\mathbf{x}$ together with an associated subset $L_x \subseteq \mathcal{L}$ of class labels. In principle, it is hence possible to distinguish between acceptable and unacceptable solutions, as we have done above. Note, however, that L_x must indeed be known exactly, i.e., the representation of partial knowledge is still a problem. Besides, of course, a more refined discrimination between solutions in terms of relative preferences is not possible.

2.2 Prediction and Loss Functions on Label Rankings

Apart from increased flexibility with respect to the representation of experiences, label ranking has also advantages regarding the prediction of solutions for new problems. Conventional classification learning essentially allows a classifier to make a one shot decision in order to identify the correct label. A prediction is either correct or not and, correspondingly, is rewarded in the former and punished in the latter case. The arguably best-known loss function reflecting this problem conception is the misclassification or error rate of a classifier. However, in many practical applications, the problem is not to give a single estimation, but to make repeated suggestions until the correct target label has been identified. Obviously, this task is ideally supported by a label ranking.

To measure the quality of a predicted label ranking, a suitable loss function $\ell(\cdot)$ is needed. To this end, several meaningful metrics can be used, such as the sum of squared rank distances

$$\ell_2(\tau, \tau') \stackrel{\text{df}}{=} \sum_{i=1}^{c} (\tau(i) - \tau'(i))^2. \tag{2}$$

The linear transformation of the latter loss function into a $[-1, 1]$-valued similarity measure is well-known as the *Spearman rank correlation coefficient* [19].

Remark: Regarding the relation between classification and label ranking, ordering the class labels according to their probability of being the top-label (i.e., the "true" label in classification) as suggested, e.g., by a probabilistic classifier, does not usually yield a good prediction in the sense of a ranking error such as (2). To illustrate, suppose that $\mathbb{P}(1 \succ 2 \succ 3) = 0.5$, $\mathbb{P}(3 \succ 2 \succ 1) = 0.3$, $\mathbb{P}(2 \succ 1 \succ 3) = 0.2$, while the probability

of all other rankings is 0. The probability of being the top-label is, respectively, 0.5, 0.2, 0.3 for the three labels 1, 2, and 3, so sorting them according to these probabilities gives $1 \succ 3 \succ 2$. However, in terms of the sum of squared rank distances (2), this ranking is suboptimal as it has a higher expected loss than the ranking $2 \succ 1 \succ 3$. This result is not astonishing in light of the fact that, by only looking at the top-labels, one completely ignores the information about the rest of the rankings. In the above example, for instance, one ignores that label 2 is never on the lowest position.

In some applications, the quality of a ranking may not depend on the positions assigned to *all* the labels. For example, consider again a fault detection problem which consists of identifying the cause for the malfunctioning of a technical system. As mentioned earlier, a ranking suggests a simple (trial and error) search process which successively tests the candidates, one by one, until the correct cause is found. In this scenario, where labels correspond to causes, the existence of a single target label λ^* (the true cause) instead of a target ranking can be assumed. Hence, an obvious measure of the quality of a predicted ranking is the number of futile trials made before that label is found. To distinguish it from real *ranking errors* such as (2), a deviation of the predicted target label's position from the top-rank has been called a *position error* [3, 16]. Needless to say, various generalizations of a position error thus defined are conceivable, depending on the type of search or problem solving process used to find the target label λ^*. For instance, imagine a process which tries exactly k solutions and then takes the best among these candidates, $\lambda^{(k)}$. In this case, a reasonable loss function is given by the quality of λ^* minus the quality of $\lambda^{(k)}$.

3 Case-Based Label Ranking

For the time being, let us make the idealized assumption that the training data submitted to the learning algorithm consists of a set of examples

$$\mathcal{D} = \{(\mathbf{x}_1, \tau_1), (\mathbf{x}_2, \tau_2) \dots (\mathbf{x}_m, \tau_m)\},$$

where each example contains a *complete* label ranking. As discussed in Section 2.1, the preference information being available in practice will usually be much weaker. However, by reducing the technical complexity, this assumption will allow us to focus on the main conceptual elements of case-based label ranking. In Section 3.2, we will discuss how to handle more general scenarios that relax the above assumption and allow for dealing with examples such as (1).

In the following, we will introduce a general case-based framework for learning label rankings. The k-nearest neighbor algorithm (k-NN) is arguably the most basic

case-based learning method [7]. In its simplest version, it assumes all instances to be represented by feature vectors $\mathbf{x} = (x_1 \dots x_N)^\top$ in the N-dimensional space $\mathcal{X} = \mathbb{R}^N$ endowed with a distance measure $d(\cdot)$ such as the Euclidean metric. Given a query input $\mathbf{x}_0$, the k-NN algorithm retrieves the k training instances closest to this point in terms of $d(\cdot)$. In the case of classification learning, k-NN estimates the query's class label by the most frequent label among these k neighbors. As mentioned in the introduction, it can be adapted to the regression learning scenario by replacing the majority voting step with computing the (weighted) mean of the target values.

A unified view of both classification and label ranking as *discrete-valued* learning problems suggests a straightforward generalization of the k-NN algorithm which predicts the most common label ranking as a target object. However, on second thought, several obvious problems make this approach seem inappropriate in general:

- The cardinality of the target space in label ranking is $|\mathcal{S}_c| = c!$, a number exceeding the typical cardinality in classification learning abundantly clear. Therefore, if the local distribution of label rankings does not have sharp peaks, equal votes statistics are much more likely (except for $k = 1$). Random tie-breaking, a standard technique in k-NN learning, will hence be used rather frequently, resulting in randomly selecting a label ranking among the k nearest neighbors.

- In contrast to classification learning, where only the discrete metric (0/1 loss) is given on the target space, meaningful *non-trivial* metrics can be defined on label rankings (cf. Section 2.2), a property shared with regression learning. The conventional k-NN algorithm does not exploit this property in the aggregation step, which is typically realized as a simple majority vote among the neighbors instead of any sort of averaging.

To avoid these problems, a more sophisticated algorithm should incorporate the structured nature of the space of label rankings. Our approach, recently put forward in [6], considers aggregation techniques for label ranking which are conceptually related to averaging in k-NN regression learning. To this end, we incorporate a common rank aggregation model to combine the k nearest neighbors into a single ranking. Even though this model has already been used in a variety of applications, such as in combining meta-search results [8], it is a novel component in a label ranking algorithm. The *consensus label ranking* is computed such that it minimizes the sum of pairwise disagreement measures with respect to all k rankings, as will be detailed below.

3.1 Aggregating Label Rankings

Let $\tau_1 \dots \tau_k$ denote rankings of the c alternatives (labels) $\lambda_1 \dots \lambda_c$. A common method to measure the quality of a ranking

$$\tau = \mathrm{AGGR}(\tau_1 \dots \tau_k)$$

as an aggregation of the set of rankings $\tau_1 \dots \tau_k$ is to compute the sum of pairwise loss values with respect to a loss (distance) function $\ell : \mathcal{S}_c \times \mathcal{S}_c \to \mathbb{R}_{\geq 0}$ defined on pairs of rankings:

$$L(\tau) \stackrel{\mathrm{df}}{=} \sum_{i=1}^{k} \ell(\tau, \tau_i)$$

Having specified a loss function $\ell(\cdot)$, this leads to the optimization problem of computing a ranking $\hat{\tau} \in \mathcal{S}_c$ (not necessarily unique) such that

$$\hat{\tau} \in \arg\min_{\tau \in \mathcal{S}_c} \sum_{i=1}^{k} \ell(\tau, \tau_i). \tag{3}$$

For the sum of squared rank distances as a loss function, a provably optimal solution of (3) is obtained by ordering alternatives according to the so-called *Borda count* [15], a voting technique well-known in social choice theory. The Borda count of an alternative is the number of (weighted) votes for that alternative in pairwise comparisons with all remaining options. This voting rule requires computational time on the order of $\mathcal{O}(kc + c \log c)$ and thus can be evaluated very efficiently.[2]

In the experimental section, we will use the Borda-count ordering technique as it is computationally efficient and has a sound theoretical basis. However, as the aggregation component is an isolated module within our case-based framework, alternative aggregation techniques which may be suitable for the particular application at hand may be integrated easily (such as aggregation techniques which minimize loss functions focusing on correct top ranks rather than distributing equal weights to all positions).

3.2 Extensions of Label Ranking

Practical applications of (case-based) label ranking suggest several generalizations of the framework that we introduced above. Essentially, these generalizations concern the target space, that is, the set $\mathcal{S}_c$ of all rankings over $\mathcal{L}$. As an appealing property of the

[2] More technical details can be found in [6], where the aggregation problem is also considered for other loss functions.

case-based framework, replacing $\mathcal{S}_c$ by any more general space, say, $\mathcal{S}_c^{ex}$ can be done quite easily without changing the framework itself, provided that $\mathcal{S}_c^{ex}$ can be endowed with a suitable distance measure. In the following, we give a brief overview of some important extensions, though without going into much technical detail.

3.2.1 Rankings with Ties.

So far, we assumed rankings in the form of *strict* total orders, which means that, for any pair of alternatives λ_i, λ_j, either $\lambda_i \succ \lambda_j$ or $\lambda_j \succ \lambda_i$. More generally, as mentioned in Section 2.1, it might be reasonable to allow for the case of indifference ($\lambda_i \sim \lambda_j$), that is, to consider *rankings with ties*. A ranking of that kind is also referred to as a *bucket order* [8]. More precisely, a bucket order is a transitive binary relation $\succ$ for which there exist sets $B_1 \ldots B_m$ that form a partition of the set of alternatives $\mathcal{L}$ such that $\lambda_i \succ \lambda_j$ if and only if there exist $1 \leq k < l \leq m$ such that $(\lambda_i \in B_k) \wedge (\lambda_j \in B_l)$. A bucket order induces binary preferences among labels and, moreover, forms a natural representation for generalizing various metrics on strict rankings to rankings with ties. To this end, we define a generalized rank $\sigma(i)$ for each label $\lambda_i \in \mathcal{L}$ as the average overall position $\sigma(i) = \sum_{l<j} |B_l| + \frac{1}{2}(|B_j| + 1)$ within the bucket B_j which contains λ_i. Fagin *et al.* [8] proposed several generalizations of well-known metrics such as Kendall's tau and the Spearman footrule distance.

3.2.2 Calibrated Rankings.

A particularly interesting generalization of label ranking is *calibrated label ranking* as introduced in [5]. Roughly speaking, a calibrated ranking is a ranking with an additional neutral label which splits a ranking into two parts, say, a positive and a negative one. This way, it becomes possible to combine absolute and relative preference information as introduced in Section 2.1. For example, in a CBR context, the positive part may consist of those alternatives (solutions) λ_i which are feasible, while the alternatives in the negative part are not acceptable as solutions for the current problem. The ranking further refines this crude distinction, e.g., one alternative can be better than another one, even though both are feasible.

An elegant extension of distance measures for rankings to measures for calibrated rankings was proposed in [6]. The basic idea is to define the distance between two calibrated rankings by the distance between the associated extended rankings which include the neutral label. Moreover, the neutral label can be duplicated to broaden the gap between the positive and the negative part. This way, a deviation of a label's estimated position

from its true position is punished more strongly if it furthermore leads to putting the label on the wrong side.

3.2.3 Partial Preference Information.

The assumption that a *complete* ranking is given for every training example will generally not be satisfied in practice. Instead, only partial preference information will be available, e.g., a ranking of only a subset of the labels $\mathcal{L}$. The problem of extending distance measures to partial preference relations was studied by Ha and Haddawy [11]. Here, the basic idea is to consider the set of all consistent extensions of such rankings (to complete rankings), and to measure a distance between these extensions. Again, this is a quite elegant approach, even though it may become computationally complex.

4 Experiments

As already mentioned earlier, label ranking essentially assumes a finite label set $\mathcal{L}$ of small to moderate size, a property it shares with conventional classification. If this property is fulfilled for a solution space in CBR, label ranking can be applied immediately. A related experimental study, which deals with predicting a rational, decision-theoretic agent's ranking of actions in an uncertain environment, is presented in Section 4.1.

Moreover, in Section 4.2, we consider an interesting alternative, showing how label ranking can be usefully applied as a sub-component in the context of a search-based problem solving strategy operating on a complex solution space. More specifically, by determining the order of successor states in heuristic search, label ranking will be used for guiding a heuristic search process.[3] Roughly speaking, the idea is to assume that, if a certain ordering of successor nodes in a search state A turned out to be useful, the same or a similar ordering will also be useful in a similar state B.

4.1 Case-Based Decision Making

In our first experiment, we replicate a setting that has been used in the context of label ranking in [10]. The problem is to learn the ranking function of an expected utility maximizing agent. More specifically, we proceed from a standard setting of

[3] The idea of using CBR to support heuristic search has already been put forward by several authors; see e.g. [18] for a very recent and closely related approach.

expected utility theory: $A = \{a_1 \ldots a_c\}$ is a set of actions the agent can choose from and $\Omega = \{\omega_1 \ldots \omega_m\}$ is a set of world states. The agent faces a problem of *decision under risk* where decision consequences are lotteries: Choosing action a_i in state ω_j yields a utility of $u_{ij} \in \mathbb{R}$, where the probability of state ω_j is p_j. Thus, the *expected utility* of action a_i is given by

$$\mathbb{E}(a_i) = \sum_{j=1}^{m} p_j \cdot u_{ij}. \tag{4}$$

Expected utility theory justifies (4) as a criterion for ranking actions and, hence, gives rise to the following preference relation:

$$a_i \succ a_j \Leftrightarrow \mathbb{E}(a_i) > \mathbb{E}(a_j). \tag{5}$$

Now, suppose the probability vector $\mathbf{p} = (p_1 \ldots p_m)$ to be a parameter of the decision problem (while A, Ω and the utility matrix matrix $\mathbf{U} = (u_{ij})$ are fixed). A vector $\mathbf{p}$ can be considered as a description of the "problem" that the agent has to solve, namely as a characterization of the uncertain environment in which the agent must take an action.

The above decision-theoretic setting can be used for generating synthetic data for label ranking. The set of instances (problems) corresponds to the set of probability vectors $\mathbf{p}$, which are generated at random according to a uniform distribution over $\{\mathbf{p} \in \mathbb{R}^m \mid \mathbf{p} \geq \mathbf{0}, p_1 + \ldots + p_m = 1\}$. The ranking associated with an instance is defined by the pairwise preferences (5). Thus, an experiment is characterized by the following parameters: The number of actions/labels (c), the number of world states (m), the number of examples (n), and the utility matrix which is generated at random through independent and uniformly distributed entries $u_{ij} \in [0, 1]$.

In this study, we applied our case-based approach to label ranking (CBLR), using the aggregation technique described in Section 3.1. For comparison, we used the standard k-NN classification method, which simply orders the class labels according to the number of votes they receive from the neighbors (ties are broken at random). The main goal of this study is to show the benefit of the additional information contained in the comparison of suboptimal solutions, which is exploited by CBLR but not by the simple k-NN classifier.

In the experiments, we chose the problem dimensions to be $m \in \{5, 10, 15, 20\}$, $c \in \{5, 15, 20\}$, and fixed the number of training and test examples to $1,000$ each. For each value of the input dimension, we generated 10 different label ranking problems originating from independently sampled utility matrices. As evaluation measures, we

Table 1: Results of the first experimental study. In each horizontal block, the first line shows the accuracy values for CBLR, the second line for the simple k-NN classifier.

m	*pos. err.*	*rank. err.*	*pos. err.*	*rank. err.*	*pos. err.*	*rank. err.*
5	$.967 \pm .020$	$.969 \pm .011$	$.932 \pm .026$	$.929 \pm .019$	$.878 \pm .0628$	$.880 \pm .046$
	$.967 \pm .020$	$.625 \pm .080$	$.923 \pm .025$	$.697 \pm .071$	$.866 \pm .0670$	$.706 \pm .084$
10	$.980 \pm .008$	$.979 \pm .005$	$.959 \pm .017$	$.951 \pm .007$	$.924 \pm .0333$	$.909 \pm .023$
	$.977 \pm .008$	$.437 \pm .067$	$.950 \pm .016$	$.484 \pm .102$	$.907 \pm .0373$	$.536 \pm .114$
15	$.987 \pm .005$	$.985 \pm .002$	$.966 \pm .011$	$.964 \pm .007$	$.954 \pm .0196$	$.926 \pm .006$
	$.982 \pm .006$	$.302 \pm .065$	$.957 \pm .015$	$.414 \pm .094$	$.940 \pm .0213$	$.425 \pm .109$
20	$.988 \pm .003$	$.988 \pm .002$	$.972 \pm .007$	$.968 \pm .006$	$.948 \pm .0204$	$.936 \pm .019$
	$.984 \pm .002$	$.266 \pm .043$	$.958 \pm .008$	$.337 \pm .055$	$.926 \pm .0232$	$.419 \pm .099$
	$c = 5$		$c = 10$		$c = 20$	

considered the position error (i.e., the position assigned to the true top-label) and the Spearman rank correlation. In order to simplify the comparison, the position error was re-scaled into a similarity measure on $[-1, +1]$ in a straightforward way. For each learning problem and algorithm, the neighborhood parameter $k \in \{1, 3 \ldots 19, 21\}$ was determined based upon the performance (with respect to the particular evaluation measure at hand) on a random 70/30 split of the training data. The performance results on the test sets were averaged over all 10 runs.

The results in Table 1 show that CBLR clearly outperforms the simple k-NN classifier. As it was to be expected, the differences in performance are indeed dramatic for the rank correlation which takes the complete ranking into account. However, CBLR is also superior for the position error, which is essentially a type of classification error. This result shows that exploiting information about suboptimal solutions can also improve the standard classification performance.

4.2 Label Ranking for Controlling Heuristic Search

Resource-based configuration (RBC) is a special approach to knowledge-based configuration [13]. It proceeds from the idea that a (technical) system is assembled from a set of primitive *components*. A resource-based description of components is a special property-based description in which each component (e.g. a lamp) is characterized by some set of *resources* or *functionalities* it provides (e.g. light) and some other set of resources it demands (e.g. electric current). The relation between components is

modeled in an abstract way as an exchange of resources. A configuration problem consists of minimizing the price of a configuration while satisfying an external demand of functionalities. In its simplest form it corresponds to an integer linear program $\mathbf{A} \times \mathbf{z} \geq \mathbf{x},\ \mathbf{c}^\top \mathbf{z} \rightarrow min$, where the matrix $\mathbf{A} = (a_{ij})$ specifies the quantities of functionalities offered and demanded by the components (a_{ij} = quantity of the i-th functionality offered by the j-th component, demands are negative offers), the vector $\mathbf{x}$ quantifies the external demand, and the vector $\mathbf{c}$ contains the prices of the components. A configuration is identified by the vector $\mathbf{z}$, where the j-th entry is the number of occurrences of the j-th component. In practice, it is reasonable to assume that different problems share the same *knowledge base* $\langle \mathbf{A}, \mathbf{c} \rangle$ while the external demand $\mathbf{x}$ changes. Thus, the instance (problem) space $\mathcal{X}$ can be identified by all possible demand vectors.

Since an RBC problem, in its basic form, is equivalent to an integer linear program, one could think of using standard methods from operations research for solving it. However, this equivalence is already lost under slight but practically relevant generalizations of the basic model (such as non-additive dependencies between components). Realizing a heuristic search in the *configuration space*, i.e., the set $\mathcal{Z}$ of possible configurations (identified by integer-valued vectors $\mathbf{z}$), seems to be a reasonable alternative which is more amenable toward extensions of the model. Besides, this approach is better suited for incorporating (case-based) *experience* from previously solved problems [14].

In fact, there are different ways of realizing the idea of *learning* from a set of (optimally) solved problems in connection with heuristic search. Here, we consider the possibility of employing (case-based) label ranking to guide the search process, i.e., to control the choice of search operators: By starting with the empty configuration (root of the search tree) and adding basic components one by one, every node η of the search tree can be associated with an (intermediate) configuration $\mathbf{z}(\eta)$ and a corresponding demand $\mathbf{x}(\eta) = \mathbf{x} - \mathbf{A} \times \mathbf{z}(\eta)$ which still remains of the original demand $\mathbf{x}$; the search process stops as soon as $\mathbf{x}(\eta) \leq \mathbf{0}$. The key idea of our approach is to use label ranking to predict a promising order τ in which to explore the successors of a search state, that is, the order in which adding the basic components is tried (see Figure 1); the latter hence correspond to the class labels, while the demand $\mathbf{x}(\eta)$ serves as an instance. As mentioned above, label ranking thus implements the heuristic (CBR) assumption that, to find a good solution for a problem $\mathbf{x}(\eta)$, the next component to be added to the current configuration should be one that turned out to be a good choice for a similar problem $\mathbf{x}'$ as well.

In our experiments, we generated synthetic configuration problems as follows: The components of a 5×5-matrix $\mathbf{A}$ were generated at random by sampling from a uniform distribution over $\{-1, 0, 1, 2, 3\}$ (the sampling process was repeated until a feasible

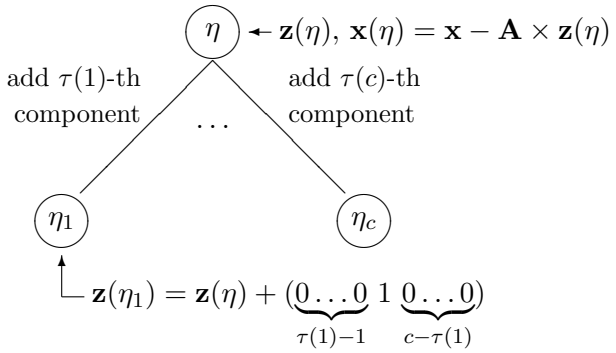

Figure 1: Configuration as heuristic search: Every node η of the search tree is associated with an intermediate configuration $\mathbf{z}(\eta)$. Each successor of η is obtained by adding one of the c available components, and the order in which components are tried is determined by the ranking (permutation) τ. The search stops if $\mathbf{x} \leq 0$.

solution with a cost ≤ 25 existed). Likewise, the components of the cost vector $\mathbf{c}$ and the demand vector $\mathbf{x}$ were sampled, respectively, from a uniform distribution over $\{1, 2, 3\}$ and $\{1, 2, 3, 4, 5\}$.

We constructed a case library as follows: Given a matrix $\mathbf{A}$, a cost vector $\mathbf{c}$ and a demand vector $\mathbf{x}$, we used iterative deepening search to determine a solution with minimum cost. Then, the search was repeated with a cost bound of twice the optimum value. In this "exploration phase", each node in the search graph corresponds to an intermediate configuration and a remaining demand vector. Moreover, each successor configuration can be associated with the minimum cost of all solutions within that subtree (which is set to infinity if no solution with bounded costs of twice the optimum value exists). Hence, from a label ranking point of view, we can see the remaining demand vector as an instance, the associated ranking of which is obtained by ordering the 5 possible successor configurations according to the subtree solution quality. We also used a calibration label (cf. Section 3.2) which corresponds to the maximum finite cost value, i.e., this label separates subtrees with finite-cost solutions from subtrees with no feasible solution.

An initial case library containing all remaining demand vectors and the associated rankings (with ties) we created for the search graph up to the maximum limit of twice the optimal cost. To reduce the number of ties and hence to increase the number of meaningful examples, we considered only those vectors for which the labels were associated with at least 2 different cost values. Fixing the matrix $\mathbf{A}$, the process was repeated 10 times for randomly sampled initial demand vectors $\mathbf{x}$ (see above). Finally, a subsample of constant size S was randomly selected from the the complete case library

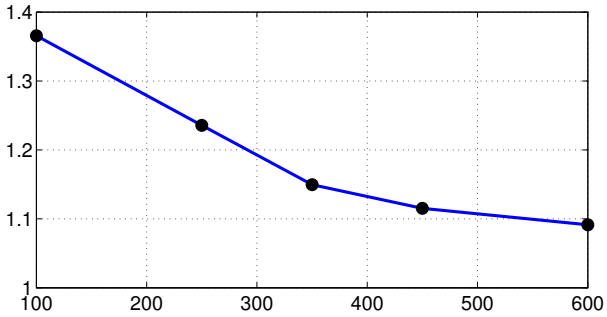

Figure 2: Performance curve for case-based heuristic search: Average ratio between the cost of the solution found and the true optimum (minimal cost), as a function of the size of the case library (number S of stored cases).

to equalize the knowledge base size for different matrices $\mathbf{A}$.

We gave this final case library to our CBLR approach (with $k = 3$) and used it as the main building block for a heuristic search strategy: Each search node corresponds to a remaining demand vector and the predicted ranking among the successor configuration can be interpreted as a qualitative ordering of the utility for searching a particular subtree. Hence, we traverse the search graph in the order dictated by the predicted ranking, where the calibration label is associated with a back-tracking step. To evaluate this strategy, we sample a new initial demand vector, conduct 100 search steps, and store the solution with minimum cost found in the course of this process. Additionally, we determined an optimal solution using the iterative deepening strategy.

The overall process of sampling $\mathbf{A}$ and $\mathbf{c}$, building the case library, and testing on a new demand vector was repeated 100 times. Figure 2 shows the average ratio between the cost of the solution found by our CBLR search heuristic and the optimal cost obtained by iterative deepening, depending on the size S of the case library. As can be seen, for a large enough case library, the solution quality comes close to the optimum. For example, for $S = 600$, the quality of the heuristic approach (which is faster by at least one order of magnitude) deviates by not more than 10% on average, showing the effectiveness of the approach.

5 Summary and Conclusions

The aim of this paper is to establish a connection between CBR and the label ranking problem that was recently introduced in the field of machine learning. In fact, our claim is that this connection can be beneficial for both sides: Firstly, as we showed in [6], a case-based approach to the label ranking problem offers an interesting alternative to hitherto existing model-based methods [12, 10]. Secondly, we have argued here that, for various reasons, label ranking can be useful in the context of case-based problem solving, especially when being compared to conventional classification learning. In particular, label ranking can better exploit the pieces of experience that accumulate in the course of a problem solving episode, and provides predictions that are potentially more helpful in finding a solution to a new problem.

To substantiate our claims, we presented experimental studies which, in absence of existing benchmark problems, are based on artificial scenarios and synthetic data. Despite the usefulness of these settings (e.g., for conducting controlled experiments), an obvious next step is to put the ideas outlined in this paper into practice, that is, to use label ranking in conjunction with CBR methods for solving real problems.

References

[1] D.W. Aha. Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies*, 36:267–287, 1992.

[2] D.W. Aha, D. Kibler, and M.K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991.

[3] C. Alonso, JJ. Rodríguez, and B. Pulido. Enhancing consistency based diagnosis with machine learning techniques. In *Current Topics in Artificial Intelligence: 10th Conference of the Spanish Association for Artificial Intelligence*, volume 3040 of *Lecture Notes in Artificial Intelligence*, pages 312–321. Springer, 2004.

[4] C.G. Atkeson, A.W. Moore, and S. Schaal. Locally weighted learning. 11:11–73, 1997.

[5] K. Brinker, J. Fürnkranz, and E. Hüllermeier. A unified model for multilabel classification and ranking. In *Proceedings* ECAI–2006*, 17th European Conference on Artificial Intelligence*, Riva del Garda, Italy, 2006.

[6] K. Brinker and E. Hüllermeier. Case-based multilabel ranking. In *Proc.* IJCAI–07*, 20th International Joint Conference on Artificial Intelligence*, pages 701–707, Hyderabad, India, January 2007.

[7] B.V. Dasarathy, editor. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, Los Alamitos, California, 1991.

[8] R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, and E. Vee. Comparing and aggregating rankings with ties. In *Proc. PODS–04, 23rd ACM Symposium on Principles of Database Systems*, pages 47–58. 2004.

[9] B. Fuchs, J. Lieber, A. Mille, and A. Napoli. Towards a unified theory of adaptation in case-based reasoning. In K.D. Althoff, R. Bergmann, and L. Branting, editors, *Case-Based Reasoning Research and Development, Proceedings* ICCBR-99, number 1650 in LNAI, pages 104–117, 1999.

[10] J. Fürnkranz and E. Hüllermeier. Pairwise preference learning and ranking. In *Proc.* ECML–2003*, 13th European Conference on Machine Learning*, Cavtat-Dubrovnik, Croatia, September 2003.

[11] Vu Ha and P. Haddawy. Similarity of personal preferences: theoretical foundations and empirical analysis. *Artificial Intelligence*, 146:149–173, 2003.

[12] S. Har-Peled, D. Roth, and D. Zimak. Constraint classification: a new approach to multiclass classification. In *Proceedings 13th Int. Conf. on Algorithmic Learning Theory*, pages 365–379, Lübeck, Germany, 2002. Springer.

[13] M. Heinrich. Ressourcenorientiertes Konfigurieren. *Künstliche Intelligenz*, 1/93:11–14, 1993.

[14] E. Hüllermeier. Focusing search by using problem solving experience. In *Proceedings* ECAI–2000*, 14th European Conference on Artificial Intelligence*, pages 55–59, Berlin, Germany, 2000. IOS Press.

[15] E. Hüllermeier and J. Fürnkranz. Ranking by pairwise comparison: A note on risk minimization. In FUZZ-IEEE–04*,* IEEE *International Conference on Fuzzy Systems*, Budapest, Hungary, 2004.

[16] E. Hüllermeier and J. Fürnkranz. Learning label preferences: Ranking error versus position error. In *Proceedings IDA–05, 6th International Symposium on Intelligent Data Analysis*, number 3646 in LNCS, pages 180–191, Madrid, 2005. Springer-Verlag.

[17] D. Kibler, D.W. Aha, and MK. Albert. Instance-based prediction of real-valued attributes. *Computational Intelligence*, 5:51–57, 1989.

[18] T. De la Rosa, A. Garcia Olaya, and D. Borrajo. Case-based recommendation of node ordering in planning. In *Proc. 20-th International FLAIRS Conference*, Key West, Florida, 2007.

[19] C. Spearman. The proof and measurement for association between two things. *Amer. Journal of Psychology*, 15:72–101, 1904.

[20] C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communications of the ACM*, pages 1213–1228, 1986.

[21] I. Watson. Case-based reasoning is a methodology not a technology. In R. Mile, M. Moulton, and M. Bramer, editors, *Research and Development in Expert Systems XV*, pages 213–223. London, 1998.