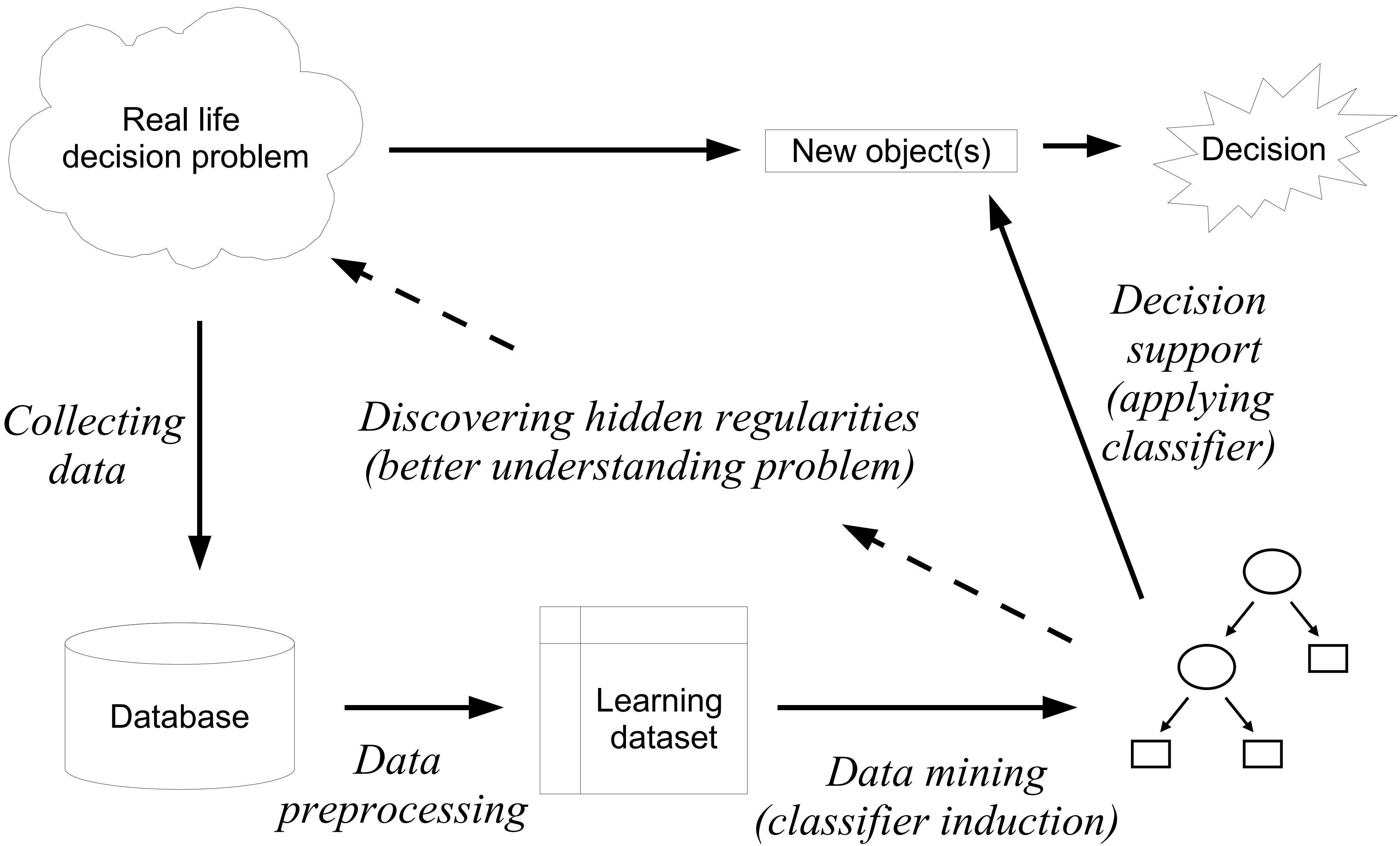
A Memetic Algorithm for Global Induction of Decision Trees

Marek Krętowski
Faculty of Computer Science
Białystok Technical University
Poland

Knowledge discovery process



Top-down versus global induction

- There exist dozens of DT systems (CART, ID3-family, AID-family, ...), but a wide diversity is somehow seeming
- Almost all approaches are based on top-down induction + post pruning
  - learning set is associated with the root node
  - the optimal test searches and data splitting are recursively repeated to consecutive subsets of the training data until the stopping condition is not met
  - fast, easy to implement & efficient in practical situations
  - however for many problems this approach fails (e.g. classical chessboard)
- Global approach
  - the whole tree (its structure and tests in non-terminal nodes) is searched at the time
  - more computationally complex, but it can reveal hidden regularities

GDT – Global Decision Tree system

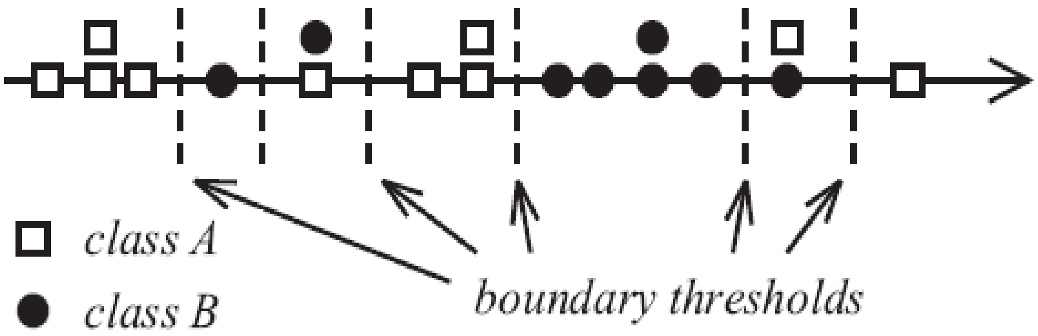
- *GDT* system is based on specialized evolutionary algorithms
  - different types of decision trees: univariate (axix-parallel), oblique and mixed
  - cost-senstive classification (different misclassification costs and feature costs)
- In contrast to classical top-down approaches *GDT* searches for the optimal tree in a global manner:
  - it learns a tree structure and splits in one run of EA
  - globally generated classifiers are generally less complex with at least comparable accuracy
- In this paper, for the first time a memetic algorithm for global induction of univariate decision trees is proposed by extending the GDT system

Memetic algorithms

- It is known that pure evolutionary methods are powerful and robust but not the fastest methods
  - a lot of effort is put into speeding them up
- One of the possible solutions is a combination of evolutionary approach with local search techniques, which is known as **Memetic Algorithms**
- However, designing a competent memetic algorithm for a given problem is not an easy task and a number of important issues have to be addressed
  - where and when local search should be applied during the evolutionary search
- In the proposed approach the local search component responsible of the optimal test search in internal nodes is introduced in the initialization and embedded into the mutation operator

Representation

- Decision trees are complicated structures as number of nodes, test types and number of test outcomes are not known *a priori* for the learning set
  - additional information in nodes is necessary during the induction (e.g. feature vectors associated with a node) => not especially encoded, represented in their actual form
- Test types:

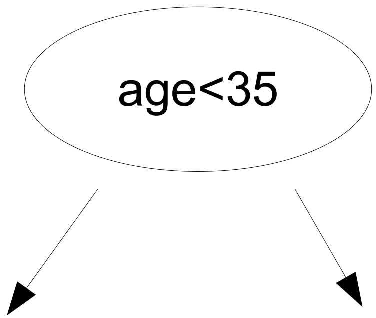


  - inequality tests with two outcomes for continuous-valued features
    - only boundary thresholds as potential splits; a boundary threshold => a midpoint between such a successive pair of examples in the sequence sorted by the increasing value of the attribute, in which examples belong to different classes
    - all boundary thresholds are calculated before starting the induction => it significantly limits the number of possible splits and focuses the search

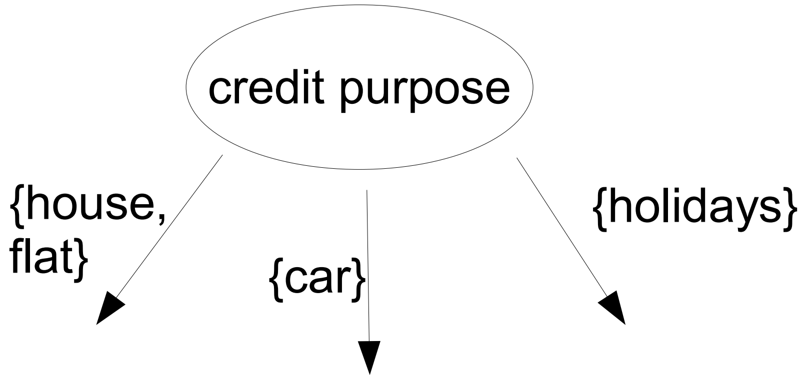
Representation (2)

   - locally applied the optimal test search can find a split, which is not based on a pre-calculated threshold
- for nominal attributes group of feature values can be associated with each branch starting in the node (internal disjunction)

Example of inequality test



Example of nominal test with internal disjunction



Initialization

- An initial population should be created with emphasis on diversity of candidate solutions:
  - it is especially useful when large search space is penetrated => it can focus and significantly speed up the search process
- Initial trees are created by applying the top-down algorithm to randomly chosen sub-samples of the original data
  - 10% of training data, but not more then 500 examples
- Test search strategies:
  - 3 strategies come from the systems (CART and C4.5) and are based on *GiniIndex*, *InfoGain* and *GainRatio*
  - *dipolar* strategy - a test splitting randomly selected mixed dipole (a pair of feature vectors from different classes) is searched
  - a random combination of all the aforementioned strategies
- Stopping condition:
  - all training objects in a node belong to the same class
  - the number of objects is lower than the predefined value (default: 5)
- Finally, the resulting trees are post-pruned according to the fitness

Genetic operators

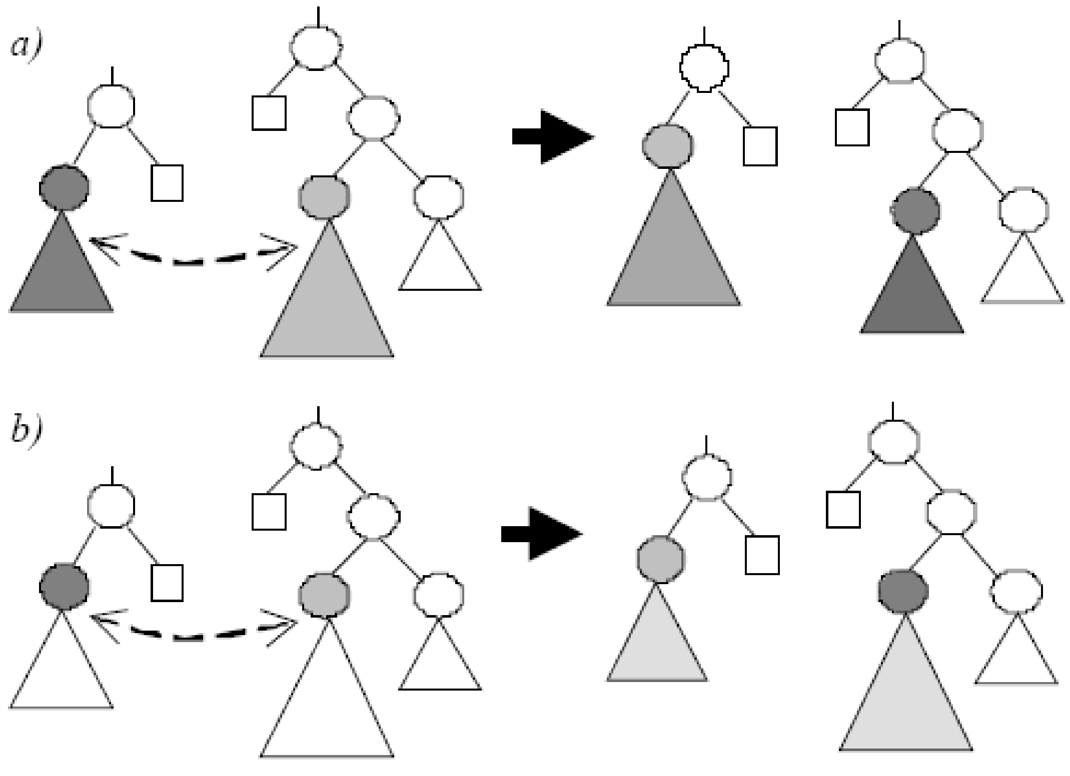
- There are two specialized operators corresponding to classical mutation and crossover
- Mutation-like operator is applied to randomly chosen node
  - first the node type is chosen and then a ranked list of nodes of the selected type is created (mechanism is similar to linear ranking selection)
- Possible modifications depend on the node type:
  - in an **internal** (non-terminal) **node** [ranking takes into account both accuracy of the subtree starting in the node and its level in the tree]:
    - a completely new test can be drawn:
      - with the user defined probability (default: 0.05) a new test can be locally optimized (*GiniIndex*, *InfoGain* or *GainRatio)* or can be chosen to split a randomly drawn mixed dipole from the learning subset associated with the node;

Mutation (cont.)

    - for nominal features only tests with the maximal number of outcomes are analyzed due to the computational complexity constraints
  - existing test can be altered by shifting the splitting threshold (continuous-valued feature) or re-grouping feature values (nominal feature);
    - modifications can be purely random or can guided by dipolar principle of splitting mixed dipoles and avoiding to split pure
  - the test can be replaced by another test from the tree,
  - the node can be transformed into a leaf
- a **leaf** [only if it is not homogeneous and leaves which are worse in terms of classification accuracy are mutated with higher probability]:
  - is transformed into an internal node and a new test is drawn
  - this can be repeated recursively repeated for descendants

Crossover

- When crossover is applied the randomly chosen parts of two trees are swapped:
  - subtrees or only tests in the nodes can be exchanged,
  - there are a few variants of this exchange which are randomly drawn taking into account structures of two subtrees



Additional operations

- The application of any genetic operator can result in a necessity of relocation of the input vectors between the parts of the tree rooted in the modified node
  - it can lead to non-effective tests and empty subtrees, which are eliminated,
  - additionally local **maximization of the fitness** function is performed by pruning lower parts of the subtree on condition it improves the value of the fitness
- Enlarging margins => improves classification accuracy
- **Centering** is applied to the best decision tree found
  - only for inequality tests - thresholds are shifted to half-distance between the corresponding feature values
  - it does not change the value of the fitness

Fitness function

- The goal of any classification system is the correct prediction of class labels of new objects => such a target function cannot be defined directly
- Accuracy on the training data is often used => their direct optimization leads to over-fitting problem
  - in typical top-down induction, the problem is mitigated by defining a stopping condition and by applying a post-pruning
- In the presented approach a complexity term is introduced and the fitness function (maximized) is defined as follows:

$$Fitness(T) = Q_{Reclass}(T) - \alpha \cdot (S(T) - 1)$$

  - where: $Q_{Reclass}(T)$ is the re-classification quality, $S(T)$ - the size of the tree (number of nodes) , α - relative importance of the complexity term (default:0.001) and a user supplied parameter
  - subtracting 1.0 eliminates the penalty when the tree is composed of only one leaf (in majority voting)

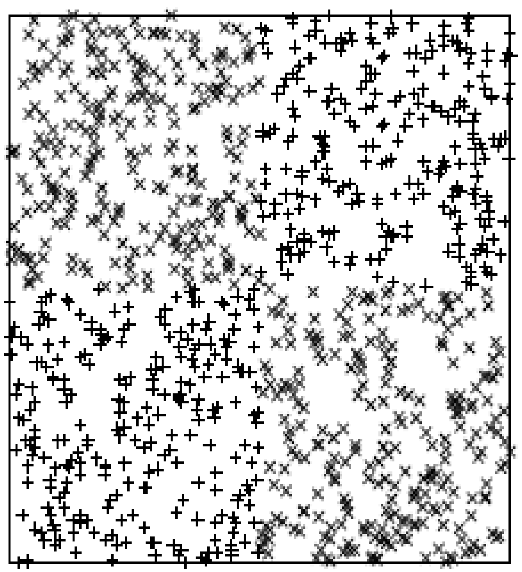
Experimental validation

- The proposed memetic system (denoted as *GDT-MA)* is compared with:
  - the well-known *C4.5* system by J. R. Quinlan
  - the standard version of *GDT (denoted as GDT-AP)*
- Performance of all systems is assessed on two types of problems:
  - artificial datasets with analytically defined decision borders
  - real-life datasets publicly available from *UCI ML Repository*
- All systems are tested with a default set of parameters
- Results correspond to averages of 10 runs and were obtained by 10-fold stratified cross-validation or measured on the testing set (when provided)
  - number of nodes (internal nodes and leaves) as the complexity measure („size” in the tables)

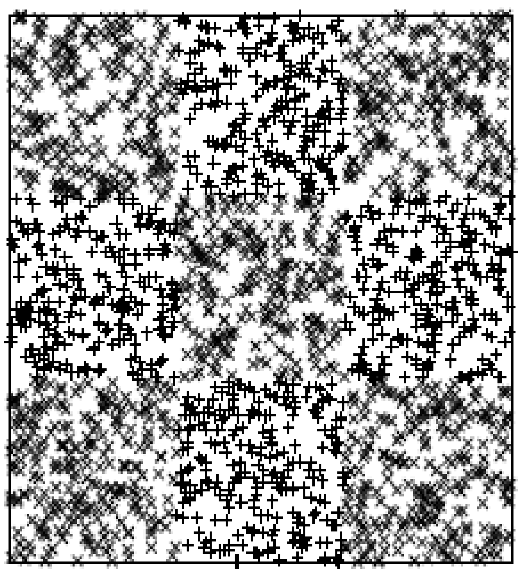
Artificial dataset

- For all domains global systems performed very well, both in terms of accuracy and complexity
- Compared to *C4.5* both global inducers were able to find proper solutions when top-down system failed and returned a default class

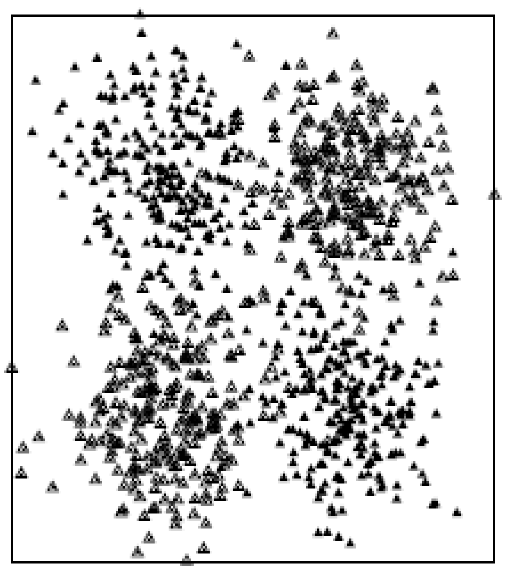
Dataset	C4.5		GDT-MA		GDT-AP	
	size	quality	size	quality	size	quality
chess2x2	1	50	4	99.9	4	99.8
chess2x2x2	1	50	8	99.8	8	99.7
chess3x3	9	99.7	9	99.8	9	99.7
chess3x3x3	54	99.3	27.2	99.0	27.1	98.9
house	21	97.4	12.1	96.4	13.3	96.6
normchess	1	50	4.1	95.5	4.2	95.5
normwave	15	94	8.8	92.6	9.1	93.5



chess2x2



chess3x3



normchess

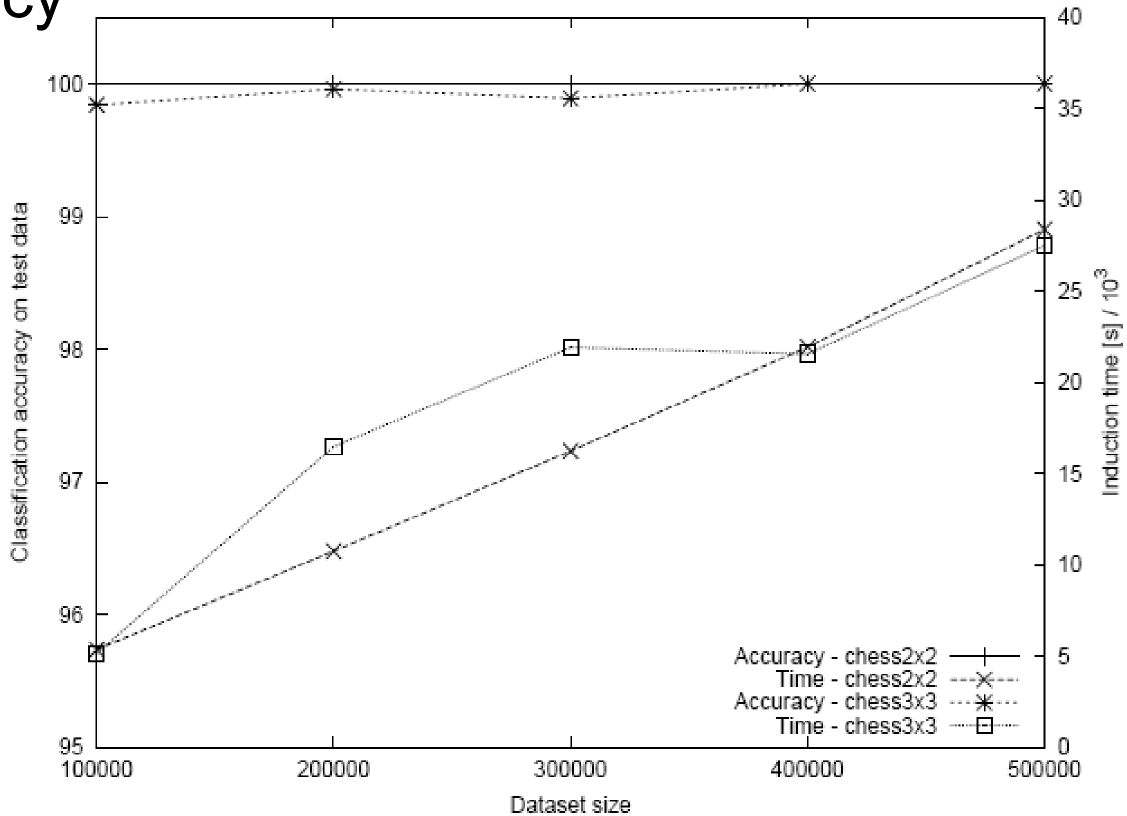
Real-life datasets

- In terms of the classification accuracy *GDT-MA* performed comparable to *C4.5* (for certain datasets it is slightly better for other is slightly worse)
- In terms of the simplicity of the solution, the proposed memetic algorithm is significantly better than *C4.5*
- *GDT-MA* was more accurate than its pure evolutionary predecessor for 12 out of 15 analyzed real-life datasets

Dataset	C4.5		GDT-MA		GDT-AP	
	size	quality	size	quality	size	quality
balance-scale	57	77.5	20.8	79.8	32.8	78.2
bcw	22.8	94.7	5.7	95.6	6.6	95.8
bupa	44.6	64.7	33.6	63.7	69.3	62.8
cars	31	97.7	3	97.9	4	98.7
cmc	136.8	52.2	19.2	55.7	13.1	53.8
german	77	73.3	18.4	74.2	16.5	73.4
glass	39	62.5	35.3	66.2	40.4	63.6
heart	22	77.1	29	76.5	44.9	74.2
page-blocks	82.8	97	7.4	96.5	7.5	96.4
pima	40.6	74.6	14.8	74.2	14.3	73.8
sat	435	85.5	18.9	83.8	19.2	83
vehicle	138.6	72.7	43.2	71.1	45.1	70.3
vote	5	97	10.9	96.2	13.5	95.6
waveform	107	73.5	30.7	71.9	36.2	72.3
wine	9	85	5.1	88.8	5.2	86.3

Performance on large datasets

- The experiment was performed on two variants of the *chess* dataset with increasing number of observations:
  - starting from 100 000 learning vectors up to 500 000
- *GDT-MA* can deal with large datasets:
  - optimal trees were found, both in terms of the accuracy and the tree size
  - acceptable time - 7 hours as measured on a typical machine (Xeon 3.2GHz, 2GB RAM)
  - induction time scales almost linearly with the dataset size



Conclusion

- A specialized memetic algorithm was developed for global induction of decision trees
  - the local search for optimal tests in nonterminal nodes based on the classical optimality criteria is embedded into the evolutionary search process
- Even preliminary results show that such a hybridization is profitable and improves the evolutionary induction
- The presented approach is still under development:
  - the influence of the local search operator on the performance must be studied in more details
  - additional optimality criteria (e.g. *TwoingRule*) are planned
- For data mining application of evolutionary algorithms there is always a strong motivation for speeding them up:
  - EA are well suited for parallel architecture => we are re-implementing the *GDT* system in a distributed environment