4. Mixed Integer Problems for Finding Optimal Tree-Based Index Structures

The best way to escape from a problem is to solve it. Alan Saporta

This chapter describes an approach for finding optimal index structures by mapping the problem into a *M*ixed *I*nteger *P*roblem (MIP) and then solving the MIP using standard algorithms.

4.1. Introduction

If all data is known a priori, an approach for constructing a good index structure is to create bottom-up structures from the leaves to the root [Finkel and Bentley, 1974]. Other examples for structures using this technique are the STR-tree [Leutenegger et al., 1997], packed-R-tree [Roussopoulos and Leifker, 1985] and the Hilbert R-tree [Kamel and Faloutsos, 1994]. Section 3.5.7 discusses these techniques. These approaches apply heuristics to cluster the multidimensional points according to some rule. These techniques use only the relations between single points (local optimization) and they do not reach generally a global optimum. An idea to find a global optimum with high probability is using simulated annealing [Pagel, 1995].

In contrast to heuristic approaches, this chapter describes an approach that guarantees finding an optimal index structure.

4.2. Optimization problem parameters

The problem of finding a good clustering of points into clusters (rectangles) is mapped to a MIP and then processed with a MIP-solver. Every tuple is a point in a d-dimensional data space and every leaf node is mapped to a rectangle. Therefore, the terms tuple and points respectively leaf nodes and rectangles have similar meanings and are used interchangeable. If a tuple belongs to a certain node/cluster, its point is in the rectangle of the leaf node it belongs to. Since overlaps between rectangles are allowed, a point lying in a rectangle does not necessarily imply that its tuple must belong to the node of that rectangle. We define now all parameters needed for our approach.

- Dimensionality of data d. Examples presented in this chapter assume the dimensionality $d = 2$. In general we assume $d \in \mathbb{N}$.
- Set P of d-dimensional points, which are to be clustered. $P = \{(p_{11}, \cdots, p_{1d}), \cdots, (p_{t1}, \cdots, p_{td})\}$.
- The cardinality of P is represented by t, ($|P| = t$). In the examples small sets with 4, 8, 12, and 16 points are used. We define $T = \{1, \cdots, t\}$.
- The number of leaf nodes or clusters is denoted by n. The number n is given as an input parameter. The set P of t tuples has to be clustered into n nodes. In the example $n \in \{1, 2, 3, 4\}$, We define $N = \{1, \cdots, n\}$.
- B_{leaf} is the maximal number of tuples per leaf node. This is the capacity of a leaf node. (In the example: $B_{leaf} = 5$).
- b_{leaf} is the minimum number of points per leaf node. (In the example: $b_{leaf} = 2$, minimum usage of nodes $40\,\%$).

The values of B_{leaf} and b_{leaf} are constraint. There cannot be more tuples than the capacity of all leaf pages and there must be enough tuples to fill the leaf pages with at least b_{leaf} tuples ($b_{leaf} * n \leq t \leq B_{leaf} * n$).

4.3. Mapping into a mixed integer problem

The task is to cluster t tuples into n leaf nodes. Each leaf node contains between b_{leaf} and B_{leaf} tuples. The affiliation of a tuple to a leaf node is modeled by binary variables x_{ij}. Variable x_{ij} is set to 1 if and only if tuple j belongs to leaf node i. If tuple j does not belong to leaf node i x_{ij} is set to 0. Leaf nodes are represented by the minimum bounding boxes (rectangles) of all points belonging to the cluster. l_{ik} is the lower limit of the bounding box of cluster i in dimension k, and u_{ik} is the upper border of the bounding box of cluster i in dimension k. Figure 4.1 shows an example. Based on the above parameters we define a corresponding MIP for the leaf node level. In order to extend the model to other levels of the tree, all equations have to be applied to all other levels, too. Due to its similarity of the other relations, we do not present this here.

Objective: Minimize the sum of *margins* of all clusters:

$$\min \sum_{i \in N} \sum_{k \in \{1, \cdots, d\}} (u_{ik} - l_{ik}) \tag{4.1}$$

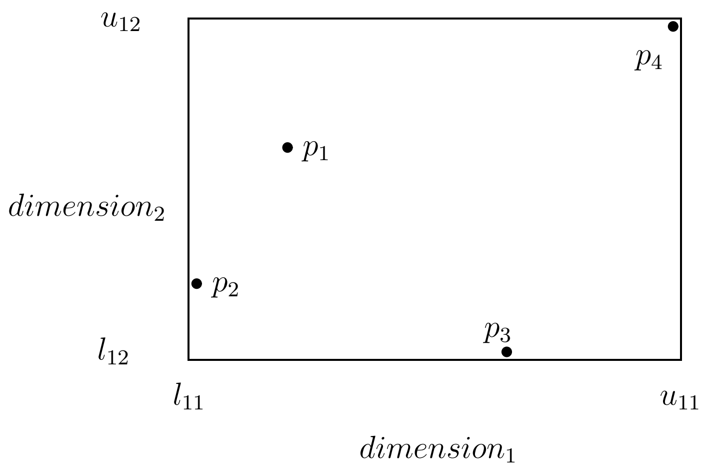

Figure 4.1.: Upper and lower bounds for MIP

We mention that all u_{ik}, l_{ik}, and x_{ij} are variables of the MIP, but only the u_{ik} and l_{ik} occur in the objective function. With *margin* we denote half of the perimeter. The minimization of the sum of margins of all clusters is chosen as the objective function because this yields rather quadratic rectangles. Other objectives like *minimize overlaps of all clusters* or *minimize the area of all clusters* do not generate quadratic rectangles and have the additional drawback that they cannot be expressed as linear functions. The complexity of MIPS with non linear objectives is higher than the complexity of MIPS with linear objectives.

To guarantee that the calculated solution satisfies all necessary conditions of a tree-based index structure, the following constraints have to hold:

Constraints:

Each point p_j belongs to exactly one cluster:

$$\sum_{i \in N} x_{ij} = 1 \ \ \forall j \in T \tag{4.2}$$

The number of tuples per leaf node is limited by the maximum fanout. Each cluster stores at most B_{leaf} points:

$$\sum_{j \in T} x_{ij} \leq B_{leaf} \ \ \forall i \in N \tag{4.3}$$

Each cluster contains at least b_{leaf} points:

$$\sum_{j \in T} x_{ij} \geq b_{leaf} \ \ \forall i \in N \tag{4.4}$$

All tuples of a leaf node are inside the bounding box of the leaf node. We check the upper limits:

$$x_{ij}p_{jk} \leq u_{ik} \;\; \forall k \in \{1, \cdots, d\} \, \forall j \in T \, \forall i \in N \tag{4.5}$$

If $x_{ij} = 1$, the point j belongs to cluster i and the coordinates of point j are lower or equal the upper bounds of cluster i. If $x_{ij} = 0$, point j does not belong to cluster i and nothing is checked. In this case the left hand side equals 0 and the constraints are true for all reasonable p_{jk} and l_{ik}. We check the lower limits:

$$(1 - x_{ij})H_k + p_{jk} \geq l_{ik} \;\; \forall k \in \{1, \cdots, d\} \, \forall j \in T \, \forall i \in N \tag{4.6}$$

where $H_k \geq \max_{j \in T} p_{jk}$. If $x_{ij} = 1$, the point j belongs to cluster i and the coordinates are checked. Then $(1 - x_{ij})$ becomes 0 and the above equation remains to $p_{ij} \geq l_{ik}$. If point j does not belong to cluster i, x_{ij} is set to 0 and a "large" constant value is added to the left side of the equation and the equation is true for all p_{jk} and l_{jk}.

The fact that the lower bounds are always less than or equal to the upper bounds can be followed from Equation 4.4, Equation 4.5 and Equation 4.6. The next four equations define the domain of the used variables. All x_{ij} are binary variables:

$$x_{ij} \in \{0, 1\} \;\; \forall j \in T \, \forall i \in N \tag{4.7}$$

All p_{jk} are not negative reals:

$$p_{ij} \in \mathbb{R}_0^+ \;\; \forall k \in \{1, \cdots, d\} \, \forall j \in T \tag{4.8}$$

All l_{jk} are not negative reals:

$$l_{ij} \in \mathbb{R}_0^+ \;\; \forall k \in \{1, \cdots, d\} \, \forall j \in N \tag{4.9}$$

All u_{jk} are not negative reals:

$$u_{ij} \in \mathbb{R}_0^+ \;\; \forall k \in \{1, \cdots, d\} \, \forall j \in N \tag{4.10}$$

Equation 4.2 through Equation 4.10 define the constraints and the variables of a MIP. A solution defined by the these equations guarantees to be a valid solution. If, in addition to the constraints, the objective function defined in Equation 4.1 is minimized, an optimal solution is found.

4.4. Problem complexity

The number of variables and constraints depends on the number of tuples t, the number of leaf nodes n, and the number of dimensions d. The time complexity of a MIP depends mostly on the number of integer variables that are defined in the MIP. This MIP contains only the binary variables x_{ij}. There are $t * n$ binary variables x_{ij}. Scientific standard algorithms solve MIPs with a branch and bound algorithm where one more integer variable increases the height of the branch and bound tree by one. Therefore, the time complexity is $O(2^{tn})$. This exponential growth implies that the algorithm cannot be used for real sized problems but only for small examples.

Table 4.1.: Calculated margins of R^*-tree and MIP for different configurations, R^*-tree on Sun Sparc 10 and MIP solver cplex 3.0 on Sun Sparc 4

Cluster	Points	fanout		Margin		CPU-time [sec]	
n	t	B_{leaf}	b_{leaf}	R^*	MIP	R^*	MIP
1	4	5	2	118	118	0.00	0.01
2	8	5	2	119	118	0.01	0.34
3	12	5	2	149	147	0.02	38.64
4	16	5	2	176	173	0.02	14127.84

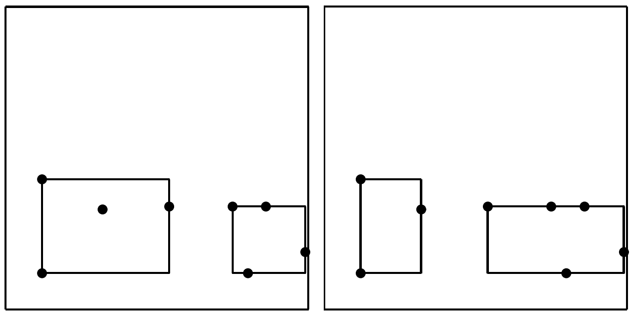
R^* margin=119 MIP margin=118

Figure 4.2.: Clustering for $t = 8$, $n = 2$

4.5. Model evaluation

Although the time complexity is high, evaluation of the MIP is done for small data sets. We run experiments in order to check how the solutions calculated by the above described MIP differ from solutions found by heuristics. We apply the MIP and the R^*-tree for different numbers of tuples and clusters. We use the R^*-tree as a reference structure throughout this thesis. New techniques applied in this thesis are tested against the widely used R^*-tree.

Table 4.1 shows the results of experiments with four sets of points which are clustered into one to four leaf nodes. The results in column five and column six show that the clusterings found by the R^*-tree have approximately the same quality in terms of minimum margin as the MIP-clusterings. Therefore, heuristics as the R^*-tree are quite good in clustering data for small problems.

The costs for computing the MIP-clustering grow exponentially with the number of clusters and the number of points. Experiments show that this approach is not feasible for practical problems. The MIP is solved with cplex 3.0 running on a Sun Sparc Station-4. Better hardware and software could make the experiments faster, but for real problems, it is still slow. The R^*-tree implementation runs on Sun Sparc Station-10. The Sun Sparc Station-10 is about four times faster than the Sun Sparc Station-4. This constant factor in speed does not change the meaning of the results.

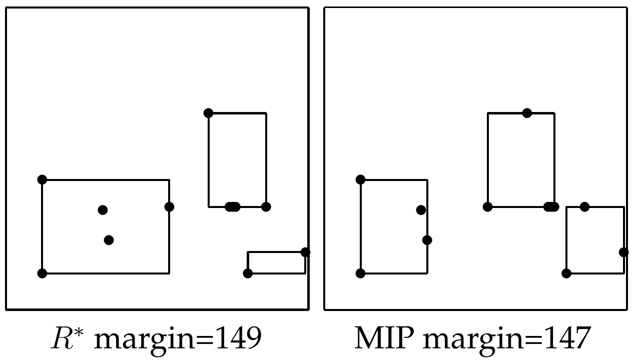

Figure 4.3.: Clustering for $t = 12$, $n = 3$

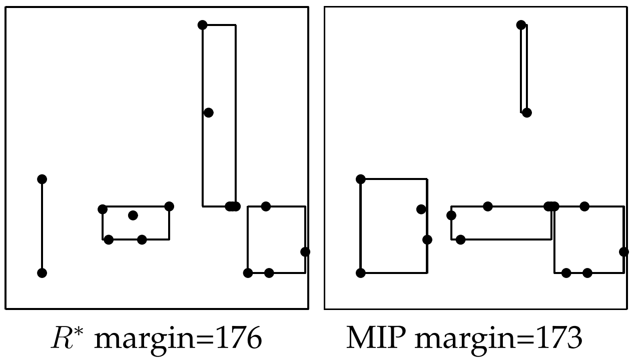

Figure 4.4.: Clustering for $t = 16$, $n = 4$

The clusterings of the first experiments with $n = 1$ and $t = 4$ found by the MIP and R^*-tree are the same and, therefore, are not graphically presented.

The left sides of Figure 4.2, Figure 4.3, and Figure 4.4 show the clusterings generated by the R^*-tree. The figures on the right side show the clusterings calculated by the MIP approach. The R^*-tree [Beckmann et al., 1990] is an heuristic approach which is applied to cluster points to the rectangles on the lowest level of the tree-structure. It calculates solutions that are nearly as good as the *optimal* solutions calculated by the the MIP approach.

More experiments with the optimizer MOPS [Suhl, 1998] show that the gap between the solution of the relaxed LP (linear problem without integer constraints) and the solution with the integer constraints is high. Therefore, the branch and bound tree becomes large and the execution time for solving the MIP increases. More advanced techniques like column generation, could be applied to speed up moderately this technique.

4.6. Summary

This chapter investigates the creating of optimal tree-based index structures by mapping the problem of finding an optimal index structure into a MIP. The MIP is solved and the solution represents an optimal index structure according to an objective function. Experiments show that the solutions found by the MIP are only slightly better than solutions found by the heuristic R^*-tree. The time complexity of the MIP grows exponentially in the size of the input. Because of this time complexity this approach cannot be applied to real world databases. However, for small data sets this technique evaluates how closely the heuristic approaches attain its optimum. In the next chapters we will apply heuristic techniques to organize multidimensional data. Heuristic approach scale much better with the problem size than the MIP approach.