An experimental study on the complexity of left-deep join ordering problems for cyclic queries*

Birgitta König-Ries[1] *Sven Helmer*[2] *Guido Moerkotte*[3]

Fakultät für Informatik[1,2]
Universität Karlsruhe
Am Fasanengarten
D–76128 Karlsruhe
Germany
koenig|helmer@ira.uka.de

Lehrstuhl für Informatik III[3]
RWTH-Aachen
Auf der Hörn
D–52056 Aachen
Germany
moer@gom.informatik.rwth-aachen.de

Abstract

Not only in deductive databases, logic programming, and constraint satisfaction problems but also in object bases where each single dot in a path expression corresponds to a join, the optimizer is faced with the problem of ordering large numbers of joins. This might explain the renewed interest in the join ordering problem. Although many join ordering techniques have been invented and benchmarked over the last years, little is known on the actual effectiveness of the developed methods and the cases where they are bound to fail. The problem attacked is the discovery of parameters and their qualitative influence on the complexity of single problem instances and on the effectiveness of join ordering techniques including search procedures, heuristics, and probabilistic algorithms. Thus an extensive analysis of the search space is carried out, with particular emphasis on the existence of phase transitions in this space and on the influence the parameters have on these transitions.

Having these parameters on hand serves two important tasks. (1) For a given heuristic, parameter combinations can be identified where it performs nearly optimal and others where it performs badly. Hence, on the one hand, we can be confident about the results of a heuristic for well determined cases and, on the other hand, can avoid the application of a heuristic where it is bound to fail. (2) When benchmarking join ordering heuristics, one had—up to now—to choose a benchmark arbitrarily. Given the parameters which influence the complexity of a join ordering problem it becomes possible to consciously design challenging benchmarks.

Index Terms — databases, query optimization, join ordering, cyclic queries, complexity, phase transition, simulated annealing, iterative improvement, heuristics.

*This work was supported by the German Research Council (DFG) under contract number SFB 346/A1.

1 Introduction

A join ordering problem occurs whenever several joins occur. A join is the manifestation of an evaluation of a logical conjunction. Hence, whenever several explicit or implicit conjunctions occur between relations, predicates, or constraints an equal number of joins occurs. For example, consider rules in deductive databases or logic programming where the body mostly is a conjunction of predicates. Consider database queries with conjunctions between relations. Especially high numbers of conjunctions are seen in generated queries. Further, constraint satisfaction problems as well as object bases apply the notion of conjunction. Having several conjuncts, the question in which order to evaluate them — or, in which order to evaluate the corresponding joins — occurs immediatly since the costs of different join orders may differ vastly. The problem of finding an optimal join order is typically referred to as the *join ordering problem* which is more precisely defined in the subsequent section.

The standard, and maybe even today prevailing method to determine an optimal join order is dynamic programming [17]. In 1984, the proof for the NP-completeness of join ordering for cyclic queries was presented together with an algorithm ordering joins for tree queries optimally in $O(n^2 \log n)$ time [8]. This algorithm was subsequently improved to $O(n^2)$ time complexity [13]. A heuristic for join ordering applying this algorithm to the minimal spanning tree of the join graph started the investigation of non-trivial heuristics for join ordering [13].

Given the increasing relevance of ordering high numbers of joins, more join ordering techniques were developed. Most predominant are those based on stochastic optimization like simulated annealing, iterative improvement, and combinations of these or with heuristics [9, 12, 18, 19, 20]. While all these methods were benchmarked, almost nothing is known on the cases where the join ordering methods perform best and where they are bound to fail. Further, the relevance of the benchmarks is still not clear. Investigations in this direction are rare. Results are the NP-completeness of join ordering [8] and characterizations of the search spaces for left-deep and bushy join trees with and without cross products [10, 11, 16]. These investigations talk about classes of problems or search spaces rather than single problem instances. Nevertheless, observing the behavior of search procedures or heuristics, there are clearly problem instances on which they perform well and others on which they perform badly. Despite the obvious benefit of having this information on hand, further characterizations of these problem instances are missing.

This situation is different for other classes of problems known to be NP-complete (like satisfiability [1, 3, 5, 15], graph coloring [1], Hamiltonian circuits [1], traveling salesman [1], and constraint satisfaction [21]), where a whole bunch of experimental investigations on the hardness of problem instances was initiated by Cheeseman, Kanefsky, and Taylor [1].

This paper is a first step in filling this gap for the join ordering problem. Inspired by the above mentioned work, we designed experiments in order to detect the relevant parameters which influence the hardness of single join ordering instances. The obvious benefits for this case are:

1. The designer of an optimizer can classify queries such that heuristics are applied

where they guarantee success; cases where they are bound to fail can be avoided. Furthermore, taking into account the vastly different run time of the different join ordering heuristics and probabilistic optimization procedures, the designer of an optimizer can choose the method that achieves a satisfactory result with the least effort.

2. The developer of search procedures and heuristics can use this knowledge to design methods solving hard problems (as exemplified for graph coloring problems [7]).

3. The investigator of different join ordering techniques is able to (1) consciously design challenging benchmarks and (2) evaluate existing benchmarks according to their degree of challenge.

The rest of the paper is organized as follows. Section 2 introduces the join-ordering problem together with its parameters and specifies the benchmarks we have carried out. Section 3 thoroughly investigates the variations of the shape of the search space as the parameters vary. A phase transition will be discovered. The paper then continues with an investigation of the influence of the parameters on the performance of search procedures (Section 4), on the effectiveness of heuristics (Section 5) and the performance of probabilistic optimization techniques (Section 6). Section 7 concludes the paper.

2 General Remarks

2.1 The join-ordering problem

This subsection introduces the join-ordering problem. It can be skipped by the readers already familiar with it. Since we built on some knowledge of the relational model we refer the unacquainted reader to, e.g., [14].

An instance of a join-ordering problem is fully described by the following parameters. First, n relations $R_1, \ldots, R_n$ are given. Associated with each relation is its *size* $|R_i|$, also denoted by n_i. Second, a *query graph* whose nodes are the relations and whose edges connect two relations by an undirected graph constitutes the second parameter. The edges of the query graph are labelled by their according *selectivity*. Let (R_i, R_j) be an edge in the query graph. Then, the associated selectivity $f_{i,j}$ is an abstraction of a *join predicate* $p_{i,j}$:

$$f_{i,j} := \frac{|\{t_i \circ t_j | t_i \in R_i, t_j \in R_j, p_{i,j}(t_i, t_j)\}|}{|\{t_i \circ t_j | t_i \in R_i, t_j \in R_j\}|}$$

where $|\cdot|$ denotes the cardinality of a set and $\circ$ denotes tuple concatenation. Defining the join ($\bowtie$) and the cross product ($\times$) between two relations as follows

$$\begin{aligned} R_i \bowtie_{p_{i,j}} R_j &:= \{t_i \circ t_j | t_i \in R_i, t_j \in R_j, p_{i,j}(t_i, t_j)\} \\ R_i \times R_j &:= \{t_i \circ t_j | t_i \in R_i, t_j \in R_j\} \end{aligned}$$

the above definition of a selectivity can be rewritten to

$$f_{i,j} := \frac{R_i \bowtie_{p_{i,j}} R_j}{R_i \times R_j}$$

Since the join predicate $p_{i,j}$ is implied by the relations involved in a join, we will just write $R_i \bowtie R_j$ instead of $R_i \bowtie_{p_{i,j}} R_j$.

The goal of *join-ordering* is to give a permutation π of $\{1, \ldots, n\}$. This permutation then corresponds to a *join expression*

$$(\ldots(R_{\pi(1)} \bowtie R_{\pi(2)}) \bowtie R_{\pi(3)} \ldots) \bowtie R_{\pi(n)}.$$

Such a join expression is often graphically depicted as a *join graph*. For π being identity, a join graph for $n = 5$ is depicted in Figure 1.

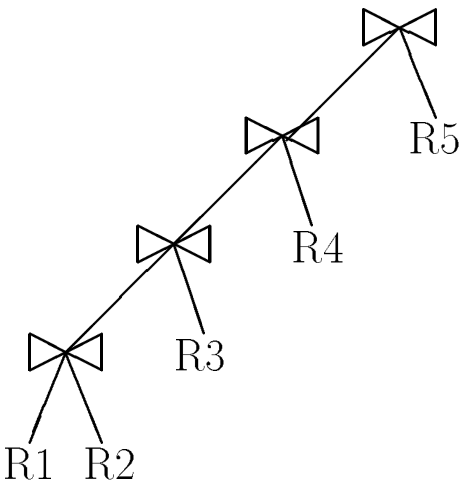

Figure 1: left-deep tree

Of course, there exist $n!$ different permutations and not any one is as good as any other one. Behind the join operators are real implementation doing the work and their corresponding costs. This leads to the third parameter of the join-ordering problem: the cost function. Since there exist several implementations for a join, there exist the several different according cost functions. The most common implementations of a join operator are

1. hash loop join
2. sort merge join
3. nested loop join

The according cost functions are:

$$\begin{aligned}
C_{hc}(R_i \bowtie R_j) &:= |R_i|1.2 + |R_i||R_j|f_{i,j} \\
C_{smc}(R_i \bowtie R_j) &:= [|R_i|\log(|R_i|) + |R_j|\log(|R_j|)] + |R_i||R_j|f_{i,j} \\
C_{nlc}(R_i \bowtie R_j) &:= |R_i||R_j| + |R_i||R_j|f_{i,j}
\end{aligned}$$

These deserve some comments. The first part of the sum results from the costs necessary to iterate over the relations and checking the join predicate for a pair of tuples. The second part, the same for all cost functions, accounts for the costs of constructing the (intermediate) result. The 1.2 of the C_{hc} cost function accounts for the average length of the collision list of the hash table.

Sometimes, only the costs of producing the intermediate results is counted for. This is especially the case, if the intermediate results must be written to disk, since then the costs of writing to disk clearly overpower the CPU costs for checking the join predicate. This cost function is referred to as C_{out}:

$$C_{out}(R_i \bowtie R_j) \;\; := \;\; |R_i||R_j|f_{i,j}$$

This is the simplest considerable cost function.

For all cost functions, we will assume a binary equivalent whose input are just the sizes n_i and n_j of the according relations R_i and R_j. For example, for C_{out}, we have

$$C_{out}(n_i, n_j) \;\; := \;\; n_i n_j f_{i,j}$$

Now, we are ready to state the *join-ordering problem.* Given n relations $R_1, \ldots, R_n$, a query graph with according selectivities $f_{i,j}$, and a cost function C_x , determine a permutation π such that the expression

$$C(\pi) := \sum_{i=2}^{n} C_x(\prod_{j=1}^{i-1}((\prod_{l=1}^{j-1} f_{\pi(j),\pi(l)})n_{\pi(j)}), n_{\pi(i)})$$

is minimal among all possible $n!$ permutations. Note that $\prod_{j=1}^{i-1}((\prod_{l=1}^{j-1} f_{\pi(j),\pi(l)})n_{\pi(j)})$ is the size of the (intermediate) relation resulting from joining the relations $R_{\pi(1)}, \ldots, R_{\pi(i-1)}$.

Why is the join-ordering problem a problem? First, the *evaluation costs* as given by the above sum of a *join order* π may vastly differ for two different π. Second, it is NP-complete, even for the simple cost function C_{out}:

Theorem 2.1 *The join-ordering problem with the cost model C_{out} is NP-complete.*

Since there exists only one proof of NP-completeness [8], where a very complex cost function taking disk accesses of a very special block wise nested loop join implementation is applied, we shortly sketch the proof for the C_{out} cost function.

Proof 2.2 *Obviously the join-ordering problem $\in$ NP. We will restrict the join-ordering problem to the Clique problem which is known to be NP-complete [4]. (The question asked in the Clique problem is, whether a graph G contains a clique of at least size K or not.) We will represent all n nodes in a graph G by relations of cardinality 1. If there is an edge between two nodes in G, then the according selectivity of the edge between the corresponding relations is set to $\frac{1}{2}$. Now, the following is obvious: G contains a clique of size K or more, iff $C_{out}(R_{\pi_{opt}(1)} \bowtie R_{\pi_{opt}(2)} \bowtie \ldots \bowtie R_{\pi_{opt}(K)}) = \sum_{i=2}^{K} \prod_{j=1}^{i} \prod_{l=1}^{j-1} f_{\pi_{opt}(j),\pi_{opt}(l)} = \frac{1}{2} + (\frac{1}{2})^3 + (\frac{1}{2})^6 + \ldots + (\frac{1}{2})^{\frac{K(K+1)}{2}}$ where π_{opt} is the optimal join-ordering of the n relations.* $\square$

The proof already indicates, that the query graph plays an essential role for the complexity of a join-ordering problem instance. This has been made explicit by showing that if the graph does not contain a cycle, i.e., is a tree, the join-ordering problem can be solved in polynomial time for all of the above cost functions [8, 13, 2]. Hence, in this paper we concentrate on an experimental study of the complexity of join-ordering problems for cyclic query graphs.

2.2 Parameters

With the above it is now easy to identify the parameters by which the complexity of the join-ordering problem is influenced:

1. the number of relations n,
2. the relation sizes n_i,
3. the query graph,
4. the selectivities $f_{i,j}$, and
5. the cost function.

We discuss each parameter in turn.

The number n of relations has a clear impact: the larger n, the more complex the problem becomes. Since we carried out experiments for which all $n!$ permutations were needed many times to do meaningful averaging of results, we could not choose an n larger than 10. For n smaller than 10, the qualitative results as presented in the next sections remained the same and we are convinced, that this is also true for larger n. Hence, we further restrict the number of relations to 10.

The relation sizes can be varied without such problems. Hence, we have choosen three different mean relation sizes: 2480, 24805, and 248050 since these reflect a wide range of in praxi occurring values. But not only the relation sizes can be varied from one join-ordering problem to the next, also the variances of the relation sizes within a given problem. Again, we have chosen to investigate three variances for relation sizes: 0.0, 6.594e+05, and 8.832e+06 subsequently denoted by Var 0, Var L, and Var XL. Analogously, we have investigated three different mean values for the selectivities: 0.05, 0.15, and 0.25, and three different variances of the selectivity values within a join-ordering problem: 0.0, $\frac{1}{1200}$, and $\frac{1}{300}$. Again, in the figures below we will refer to the different variances of the selectivities by Var 0, Var L, and Var XL. Whether we speak of relation sizes or selectivities will always be indicated. For Var XL for selectivity values, we have a problem: the value of 0.05 is too small to achieve a variance as high as $\frac{1}{300}$. Hence, curves corresponding to this parameter combination will be missing. We believe that choosing these values for mean relation sizes, relation size variances, mean selectivites, and selectivity variances covers a large portion of values occuring in practice. Nevertheless, we have carried out benchmarks with other parameters. The results will not be presented in this paper, but we assure the reader that they underline the conclusions that we will draw.

The next parameter we discuss is the query graph. Already from the NP-completeness result above and from the knowledge that acyclic queries can be solved in polynomial time, we see that this is one of the most critical parameters at hand if it comes to determining the complexity of a join-ordering problem instance. Since acyclic query graphs can be solved in polynomial time, we concentrate on cyclic query graphs. We construct query graphs by starting from a simple cycle involving each relation once (Fig. 2, left outermost graph) and then adding edges as indicated in Figure 2. This is done until we reach a certain average connectivity within the resulting graph. (The connectivity of a graph is defined as the average number of edges a node is involved in.) This connectivity then constitutes the most important parameter for our benchmarks. In fact, in all curves we will present, the connectivity varying from 2 to 9 (for 10 relations) will be the label for the x-axis. Nevertheless, it will become clear, that it is just a matter of representing numbers and that the information contents of all figures put together is not influenced by this choice, i.e., is complete considering our choices of the parameters for which we measure their influence on the complexity of a join-ordering problem.

Last not least, we have to fix our cost model. While we measured even more cost models than stated above, it became quite clear to us that the additional information and the conclusions which can be drawn from the observations from the experiments do qualitatively not depend on the choice of the cost model. Nevertheless, we have chosen to include at least (and at most) the experimental results for two different cost models. Once this decision was made, the choice of which representatives to take became easy. Remembering the definition of the cost functions one can see that they are more or less symmetric in their two arguments. Hence, we have chosen to give the results for C_{hc} as the most asymmetrical cost function and for C_{nlc} as the most symmetrical cost function, neglecting choices inbetween.

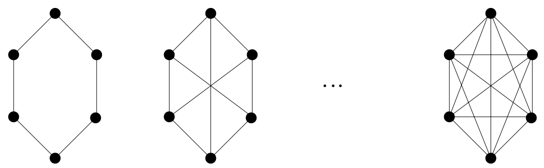

Figure 2: Increasing the connectivity

3 Search Space Analysis

The goal of this section is to determine the influence of the parameters on the shape or structure of the search space of left-deep join trees. We are particularly interested in the existence and position of phase transitions. More specifically, we are interested in answering two questions. First, we are interested in the cost difference of the worst and the best solution for the different possible parameter combinations. Second, we are interested in how a variation of the parameters impacts the phase transitions. In order to do so, we investigate the influence of the parameters on the percentage of good solutions among all solutions. The more good solutions exist, the easier the problem. Hence, within this section we will use this as the complexity measure of a problem. A phase transition then occurs where the complexity of a problem is highest.

The quality of a solution is measured by the factor its cost deviates from the optimal permutation. For each of these aspects there exists a separate subsection. The last subsection summarizes the investigations on the search space.

3.1 The Best and the Worst Case

Each picture in figures 3 to 6 shows the average optimum case cost and the average worst case cost for a specific setting of the parameters listed above. The x-axis corresponds the connectivity of the query graph, the y-axis to the costs.

On a first glance, all the pictures exhibit a similar pattern. We first describe and explain the reasons for this pattern and then describe and explain the influence of the different parameters in detail.

- Both curves descend steadily.
  Clearly, the main factor influencing the cost of a given join is the size of intermediate results. This size depends both on relation sizes and on selectivities. Thus, a join order will be the cheaper, the smaller the intermediate results are, which in turn will be the case the more selectivities are involved and the smaller these selectivities are. Thus the steady decrease in cost can be quite easily explained: The higher the connectivity of a join graph, the more selectivities are involved, the smaller the intermediate result sizes, the cheaper the join order.

- For low connectivities, the worst case and optimum cost lie very closely together.
  Since a low connectivity implies that only a few selectivities exist, no big cost differences can be obtained.

- Up to a certain (parameter depending) connectivity, the gap between the best and the worst case grows steadily.
  We assume that in order to obtain the optimal solution, the optimizer tries to minimize intermediate results. The selectivities have the main influence on these sizes, i.e. the optimizer will try to take advantage of the small selectivities as early as possible, joining the appropriate relations first. On the other side, in order to obtain the worst case, joining relations connected by an edge has to be deferred as long as possible. With increasing selectivities, the optimizer can improve the quality of the solution very fast by using the appropriate relations, that means, the cost for the optimal solution will shrink quite fast with increasing selectivities. On the other hand side, to obtain the worst case, the usage of small selectivities can be avoided for some time, accounting for the slower decrease in worst case cost.

- At some point the curve displaying the optimum cost exhibits a bend after which it falls slower.
  After a number of relations have been joined, the intermediate result tends to be so small that the joining of the last relations has nearly no influence on the total cost of the according join order. The bend represents exactly this point.

- After this connectivity is reached, the difference rests more or less constant or decreases slightly.

Now, with the general pattern being explained, let us have a look at the influence of the various parameters:

- Cost Model
  Obviously the two different cost models don't have an impact on the curves, compare for instance figures 3 and 4. Although the curves are not identical they don't show significant differences.

- Relation Sizes
  Figures 3 and 4 show the curves for different relation sizes for the hash loop and nested loop cost model, respectively. From left to right the mean relation size increases by one magnitude of order for each column.
  Obviously the difference in the cost increases with growing relation sizes. This is not surprising, since the absolute costs increase as well.

- Variance of Relation Sizes
  Within Figures 3 and 4 the relation size variances increase from top to bottom. Again, the cost deviation increases as well. Again, the reason is quite obvious: With highly differing relation sizes, extremer cost deviations can be reached, than with relations of equal size or of only a small difference.

- Selectivity Sizes
  Figures 5 and 6 show the curves for different selectivities for both cost models. From left to right, the mean selectivity sizes increase. As the figures show, the gap decreases with growing mean selectivity. At the same time, the absolute costs are increasing. The reason for both phenomena is, that the higher the mean selectivity is, the smaller is the difference in cost between a join and a cross product.

- Variance of Selectivities
  Figures 5 and 6 show differing variances in selectivity sizes as well. These variances increase from top to bottom. As explained in the previous section the bottom left picture is missing, as the combination of this very low average selectivity with the very high variance is not possible. Different variances in selectivity sizes don't seem to have an impact on the curves. The curves are more or less identical for the different variances in selectivity sizes.

3.2 The Shape of Search Space

In extensive benchmark experiments, we determined the cost of every possible permutation and compared it to the cost of the optimal solution. More specifically, we accumulated the number of permutations showing less than a 10%, 50%, 100% etc. deviation from the optimum. Figures 7-10 show the influence of the parameters on the shape of the resulting curves. Again, the x-axis is labelled by the connectivity of the query graph. The y-axis corresponds to the percentage of solutions. The different curves in one picture show the percentage of all permutations whose cost deviates from the optimum permutation by a factor smaller than the factor by which the curve is labelled. The lowest curve is labelled by the factor 1.1, the second lowest by a factor of 1.5, then the factors 2, 5, 10, 50, 100 and so on follow.

Global Picture At a first glance, we observe the following:

1. All pictures presented show the same global pattern: All curves have more or less a U-shape where some pictures only display some left part of the U.

2. All curves in one picture reach their minimum at the same connectivity.

3. The position of this minimum obviously differs depending on the value of the parameters.

Now note following. The minima of the U curves characterize problem instances where the percentage of good solutions to bad solutions is at a minimum. Hence, these minima reflect the problems of the highest complexity. Thus, the minima correspond to phase transitions.

We further observe that the number of curves displayed in each pictures strongly varies. The more curves are visible, the greater the variance in solution costs. Converting the percentage of good solutions to the notion of complexity of the problem, we can state that for each parameter combination, there exists a connectivity for which the resulting problem is the most difficult one since the percentage of good solutions reaches a minimum.

Besides the arguments already applied in the previous subsection, these findings can be illustrated as follows. First, let us bring the observation to a point: with increasing connectivity, the join ordering problem becomes more complex up to a certain point and then less complex again. To see how this might happen, consider the following special, though illustrative case. Assume equal relation sizes and equal selectivities. Then, the optimization potential *worst case/optimum* is clearly larger or equal to 1. Further, it is 1 for connectivity 0 (no edges in the join graph) and 9 (a clique). In between there exists a connectivity exhibiting the maximum optimization potential. This was already observed in Figures 3 to 6. If we assume an almost equal distribution of the costs of all alternatives between the costs of the best and the worst case, then the observation of the U-shapes follows immediately from the corresponding observation on the optimization potential.

Cost Model No clear influence can be perceived, as e.g. a comparison of figures 7 and 8 reveals. This is analogous to the results of previous subsection.

Influence of Relation Sizes Figures 7 and 8 show the influence of varying relation sizes. With growing mean relation sizes, the phase transition moves to the right, and the x-position of the phase transition coincides with the x-position of the bend of the curves in the previous section.

Influence of Variance in Relation Sizes The higher the variance in relation sizes, the more curves can be seen, i.e., the more difficult the problem becomes. The position of the phase transition is not influenced.

Influence of Selectivity Sizes Figures 9 and 10 show the influence of the selectivities. Changing average selectivity sizes influences the pictures in two ways:

- The smaller the selectivities, the more curves are visible, i.e., the more complex the optimization problem becomes.
- With growing selectivities, the position of the phase transition of the curves moves to the right. The x-position of the phase transition coincides with the x-value of the bend observed in the pictures in the previous subsection.

Both observations can be explained by the arguments already used in the previous subsection.

Influence of Variance of Selectivities The pictures show no influence of the variances in the average selecitivity sizes on the shape of the curves. Again, this is analogous to the observations of the previous section.

3.3 Summary of the Search Space Analysis

The experiments presented in this sections showed that the search space for left-deep join trees possesses a phase transition. The position of this transition as well as the general shape of the search space is clearly influenced by a number of parameters:

- connectivity of the query graph,
- average relation sizes,
- average selectivity sizes,
- variance in relation sizes (influences only the overall difficulty, and *not* the position of the phase transition).

On the other hand, the search space is obviously **not** influenced by the following parameters:

- cost model,
- variance in selectivity sizes.

Further, the minimal connectivity of the most difficult problem for a given combination of the remaining parameters is not influenced by the variance of the relation sizes.

4 Search Procedures

For evaluating the influence of the parameters on the performance of a search procedure, we analyzed the behavior of the best-first search procedure (denoted by *bf*).

The *bf* algorithm keeps a queue of left-deep trees ordered by their respective costs. The first tree, i.e. the one with the lowest costs, is discarded. It is expanded by all relations which are not yet joined and these expansions are inserted into the queue, if there does not already exist an alternative within the queue joining the same set of relations and

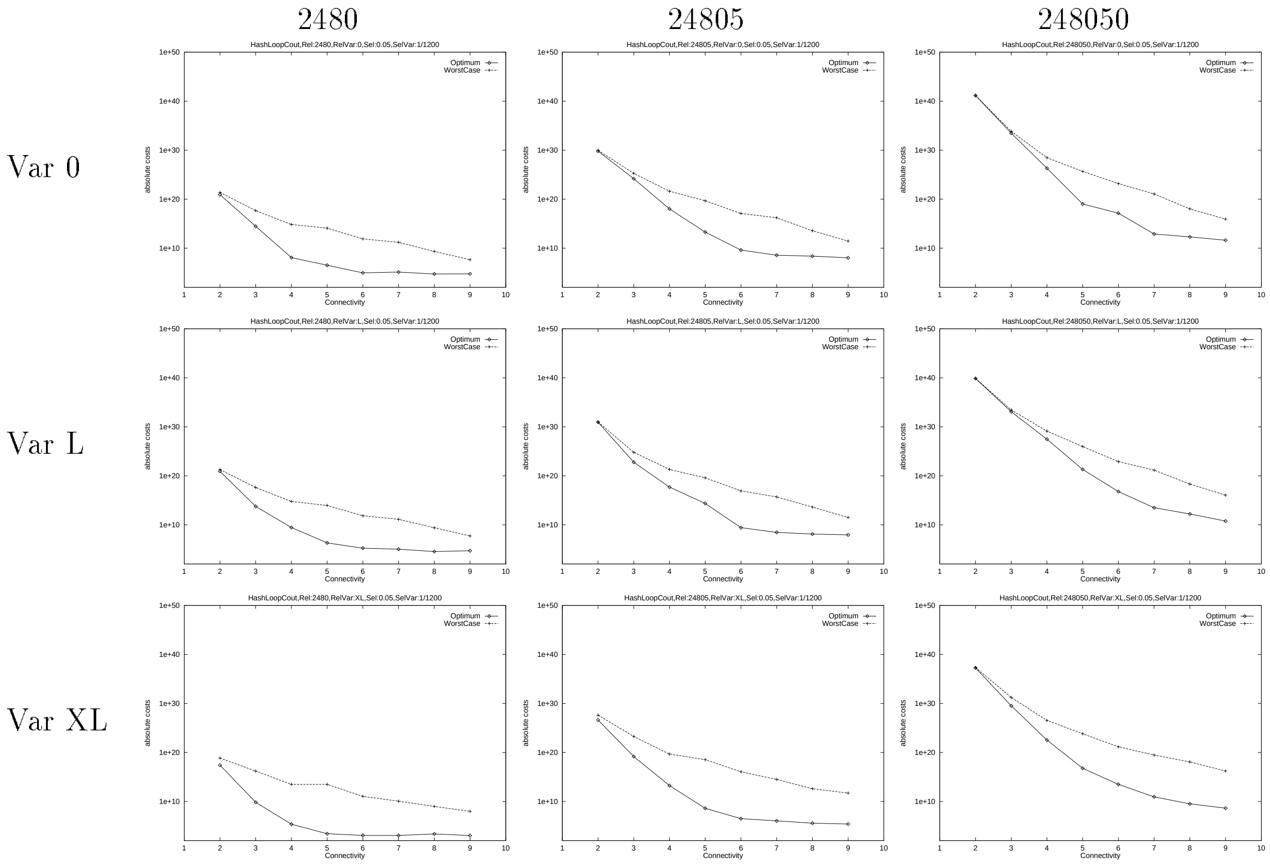

Figure 3: Search space analysis for varying relation parameters (HL)

producing cheaper cost. The queue is initialized with all the single relations ordered by their size.

There exists an alternative of *bf* which keeps track of *all* already computed alternatives and uses them for pruning. Obviously, this alternative will produce smaller numbers of expansions. We tested this alternative as well, but apart from the anticipated reduction of expansions, the pattern of the generated curves didn't differ much from the pattern of the curves generated by *bf*. Thus we present only the results of the latter in this section. The same argumentation holds for different cost models, here, too, not much change in the curves could be observed. Hence, we limit the presentation to the hash loop cost model.

Figure 11 shows the results for varying relations sizes together with their variances and the selectivities together with their variances. The x-axis depicts as usually the connectivities, the y-axis depicts the number of expansions produced by *bf*.

Influence of Relation Sizes The left column of Figure 11 gives the results for different mean relation sizes and different relation size variances. The relation size variances increase from top to bottom. Each picture contains three curves corresponding to the

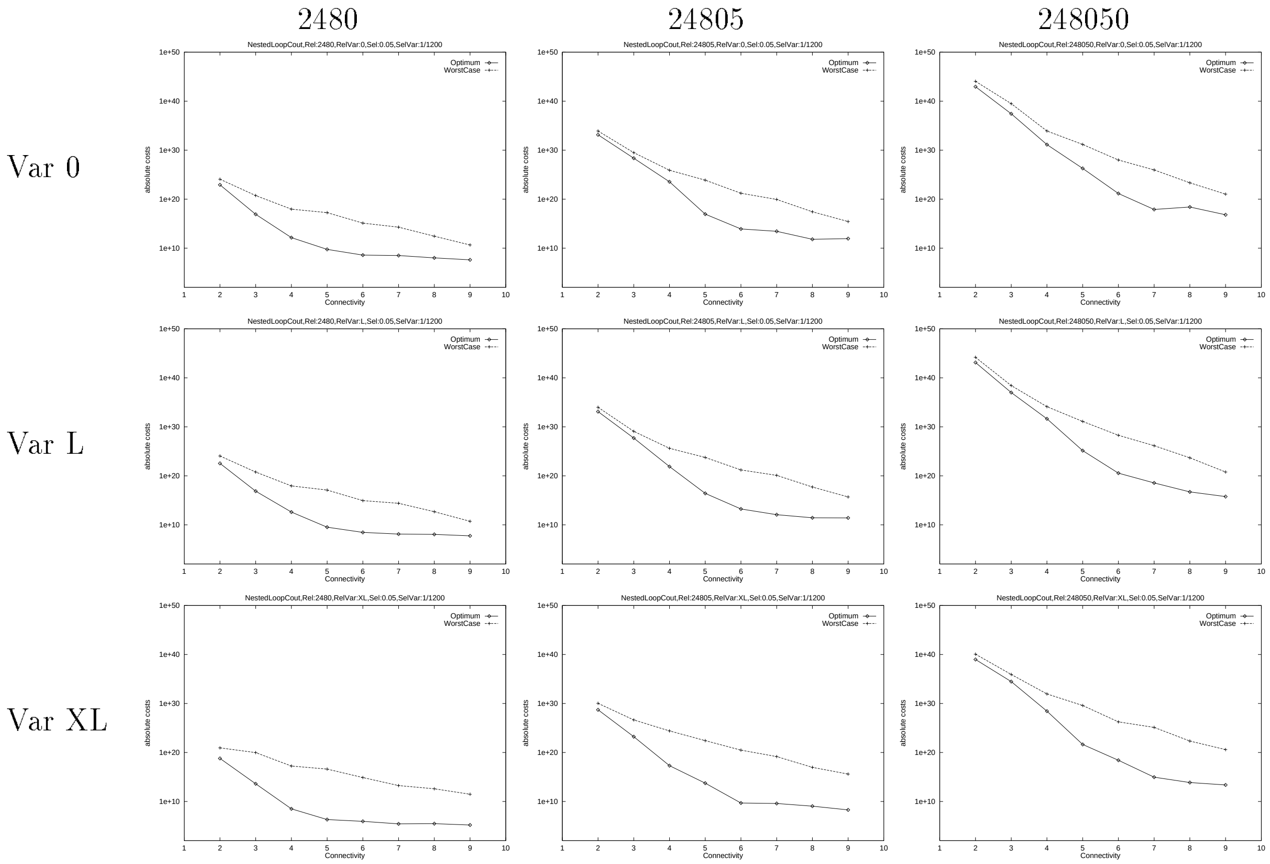

Figure 4: Search space analysis for varying relation parameters (NL)

different mean relation sizes.

We observe the following:

- All curves have a peak at a certain connectivity.
- This connectivity does not depend on the relation size variances,
- and only slightly depends on the absolute relation size where
- the peak moves to higher connectivities for higher relation sizes.
- All curves decline steeply after the peak and then flatten.

While the influence of the relation sizes is the only one observed, it is not strong. Nevertheless, comparing the results with the U-curves of the previous section, one observes that there exists a correspondence between the connectivity where the U-curves exhibit the phase transition and the point where the curves of the search procedure start flattening.

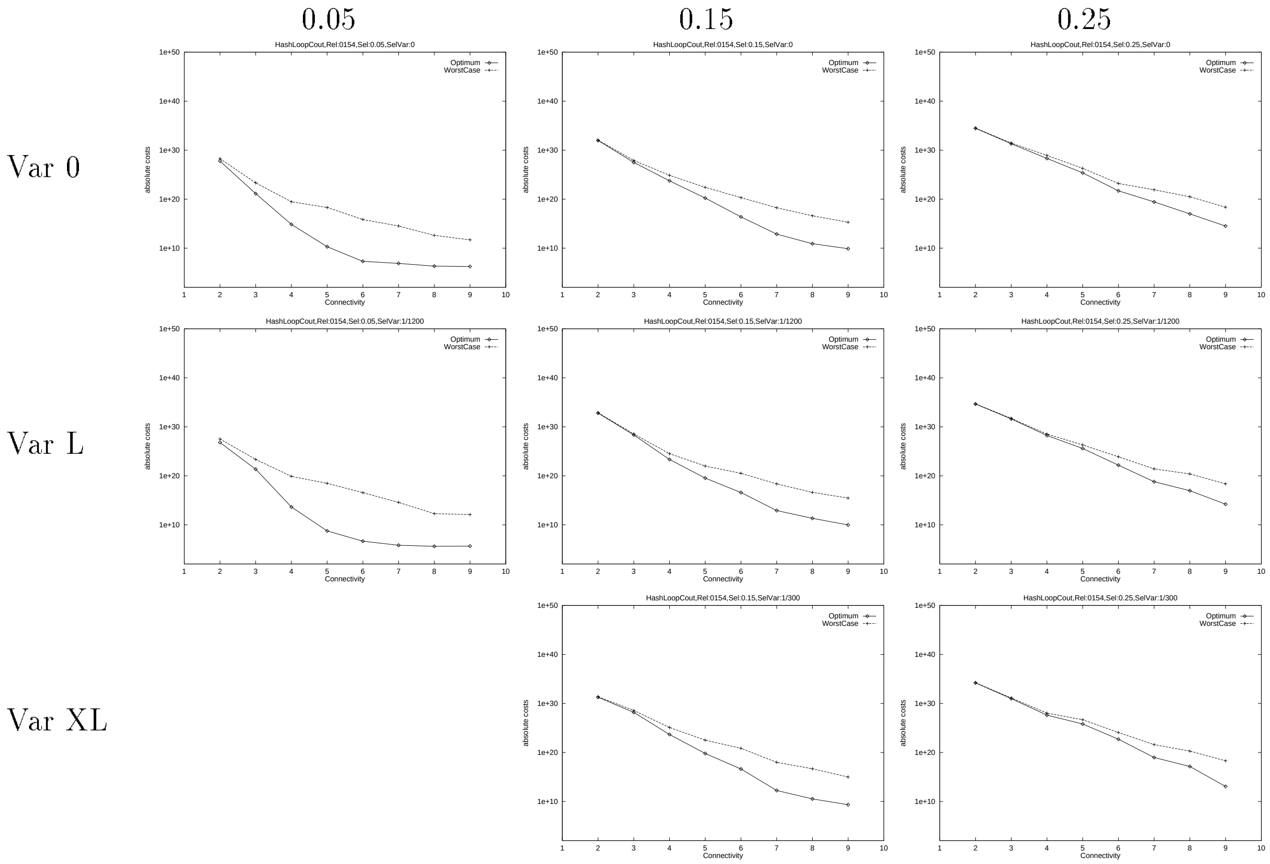

Figure 5: Search space analysis for varying selectivity parameters (HL)

Influence of Selectivities The right column of Figure 11 gives the results for different mean selectivities and different selectivity variances. The selectivity variances increase from top to bottom. Each picture, except for the last one, contains five curves corresponding to different mean selectivities. Again, for the lowest mean selectivity, the hightest variance could not be reached.

Obviously all the curves have some characteristics in common:

- All curves have a peak at a certain connectivity.
- All curves decline steeply after the peak and then flatten.
  (This point does not exist for those curves, where the peak is close to the right border of the picture).

The changes in variance does not have much impact on the shape of the curves. They differ somehow, but no clear pattern can be observed. Further, the connectivity at which the maximum is reached, is clearly independent of the variance, matching the corresponding observation of the previous section.

In contrary, the influence of the average selectivity sizes is quite obvious: with growing selectivities, the peak moves to the right. Moreover, we observe that the peak conincides

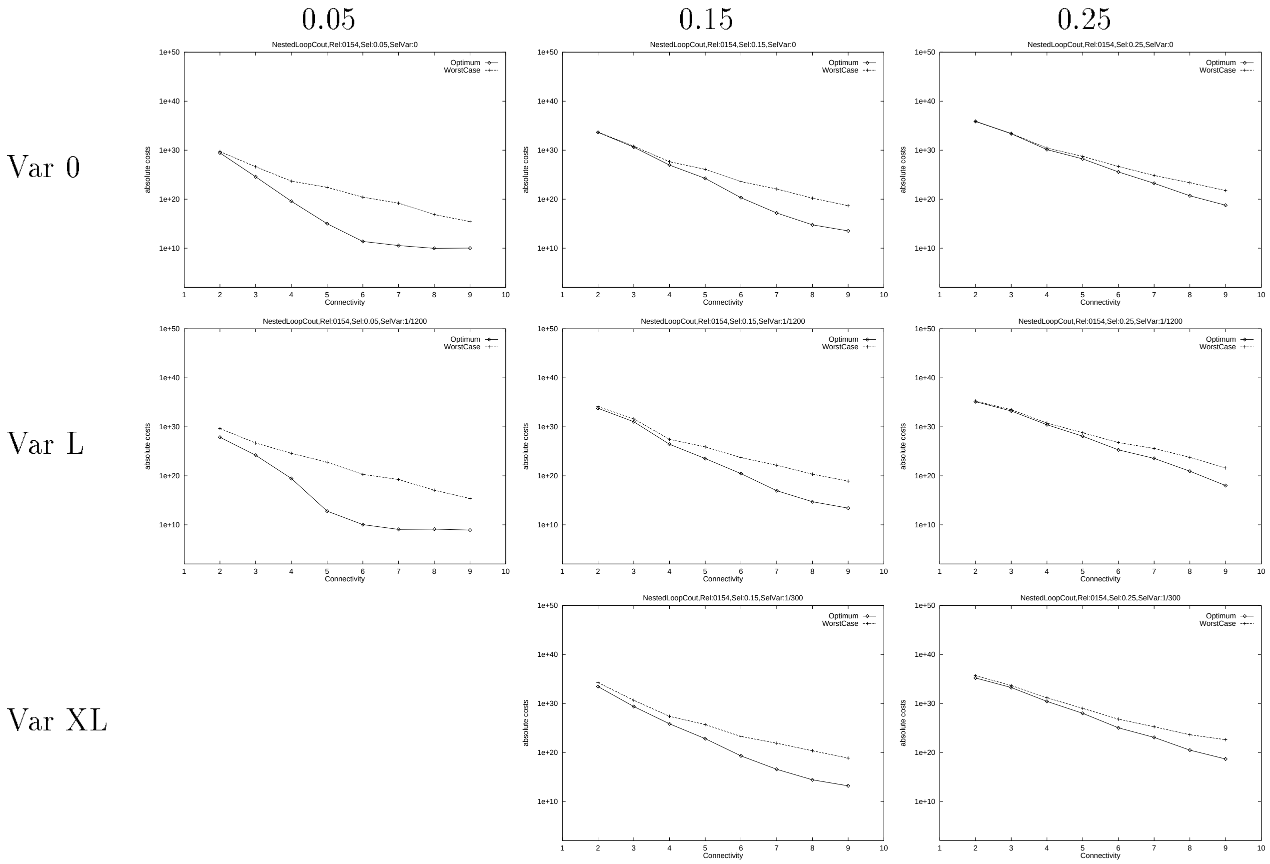

Figure 6: Search space analysis for varying selectivity parameters (NL)

with a point where the U-curves of the previous sections start flattening, typically shortly before the U-curves reach the point of the phase transition.

Further, the connectivity of the end of the steep decrease in the number of expansions for *bf* coincides with the connectivity of the minimum in the permutation curves.

Summary and Explanation We conclude that for search procedures high mean selectivities and relation sizes as well as lower connectivities are challenging. The influence of variances thereof remains neglectable. If the connectivity is high enough (higher than the peak or departure connectivity), then the problem will be solved with less expanded nodes than needed by dynamic programming (about 1000 for 10 relations).

Thus it seems that the increasing optimization potential spreads the costs of the single permutations apart enough before all relations have been joined, such that the best-first procedure has the possibility to prune early. Hence, the increasing optimization potential of the search space — as documented by the steep descend to the minimum in the U-curves — makes it easier for the search procedure to find the optimal solution. That the number of left-deep trees considered by the best-first procedure does not increase afterwards can be explained by an argument of the previous section. After the minimum

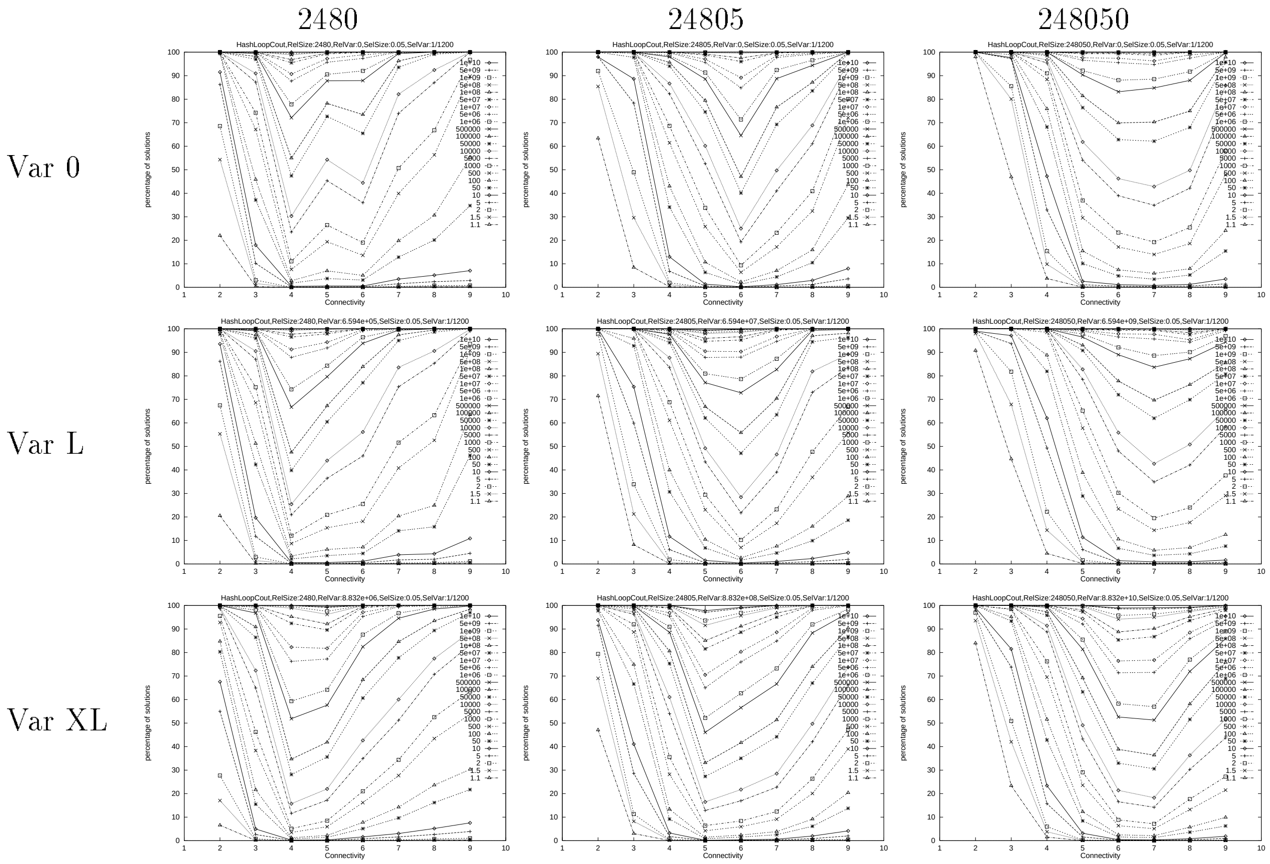

Figure 7: Search space analysis for varying relation parameters (HL)

of the U-curves, the intermediate result becomes that small, that the cost of joining subsequent relations remains neglectable. Hence, once a good deal of the bottom part of the left-deep processing tree has been constructed — which can be done fast after the minimum of the U-curves — the rest can be considered without the harm of invalidating the bottom part.

5 Heuristics

For analyzing the influence of the parameters on the performance of heuristics, we analyzed a couple of different simple and advanced heuristics proposed in the literature. Since the curves do not differ much in what can be said about the behavior of heuristics, we exemplify our investigations by giving the figures only for one simple heuristic which can easily be understood even by readers not familiar with the field. Intuitively, it seems a good idea, to choose the relation next, which has the smallest selectivities to already joined relations. For the start, we chose the smallest relation. Let us call this heuristic *MinSel*.

The results of the experiments are presented in Figure 12. The x-axis is labelled with

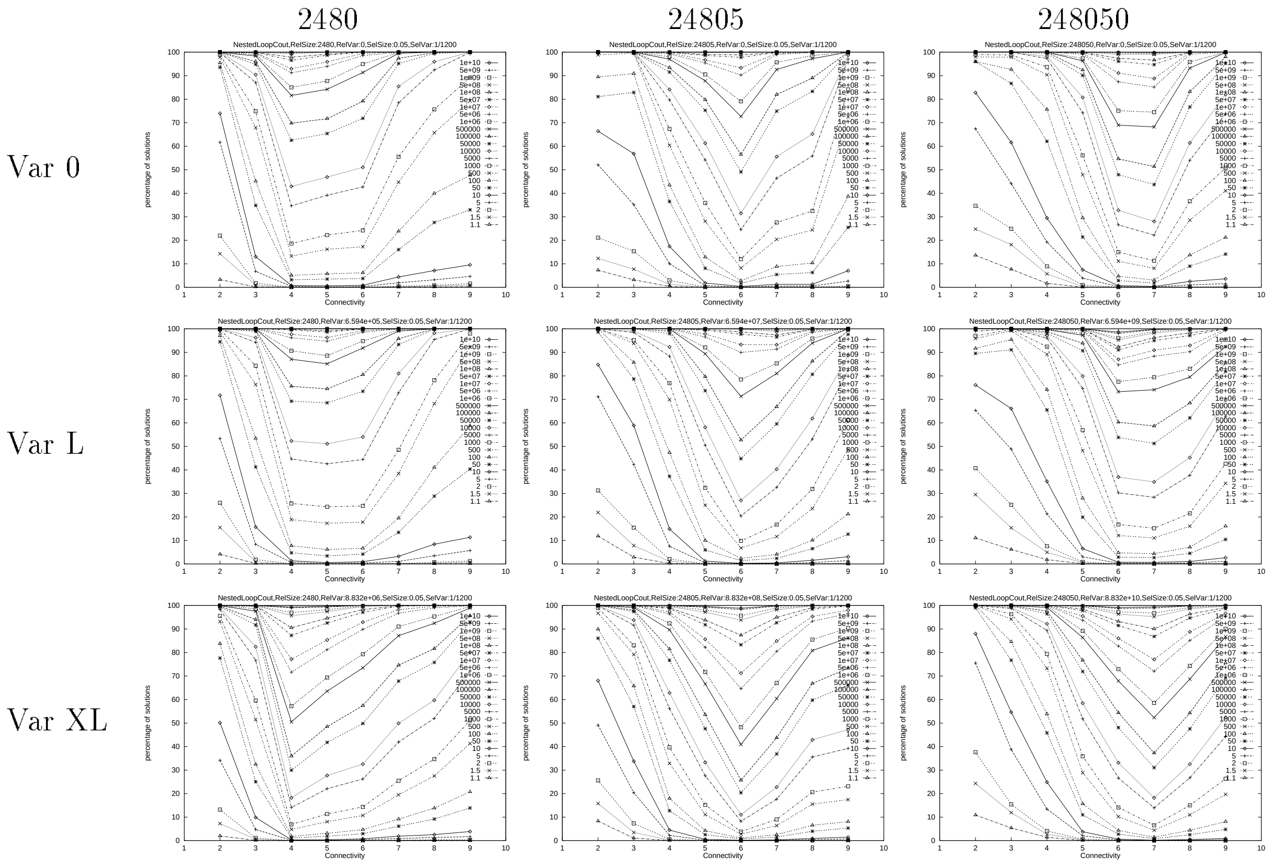

Figure 8: Search space analysis for varying relation parameters (NL)

the connectivity, the y-axis with the deviation of the heuristic's solution from the optimum. The pictures of the left column correspond to the different variances of the relation sizes. Each of the three curves within a picture corresponds to a different relation size. The pictures of the right column correspond to the different variances of the selectivities. Each of the curves within a picture corresponds to a mean selectivity value.

At a first glance, the pictures look far less regular (if not chaotic) than those presented so far. This clearly indicates the unstabibilty of the heuristic. Nevertheless, we can extract the following correspondingly rough observations. If one idealizes the curves one can guess a certain connectivity where each of the idealized curves reaches a maximum, i.e., a connectivity where the heuristic performs worst. This connectivity roughly corresponds to the connectivity where the U-curves reach display the phase transition. Then, one can also see that the peak connectivity is dependent on the selectivity size but not as regular as in the previous curves. Further, higher selectivities flatten the curves, that is, heuristics perform better at higher selectivities. An analogous observation for relation sizes cannot be drawn.

Despite the unstability of the heuristic, we can conclude that there is a correspondence between the performance of the heuristic and the shape of the search space.

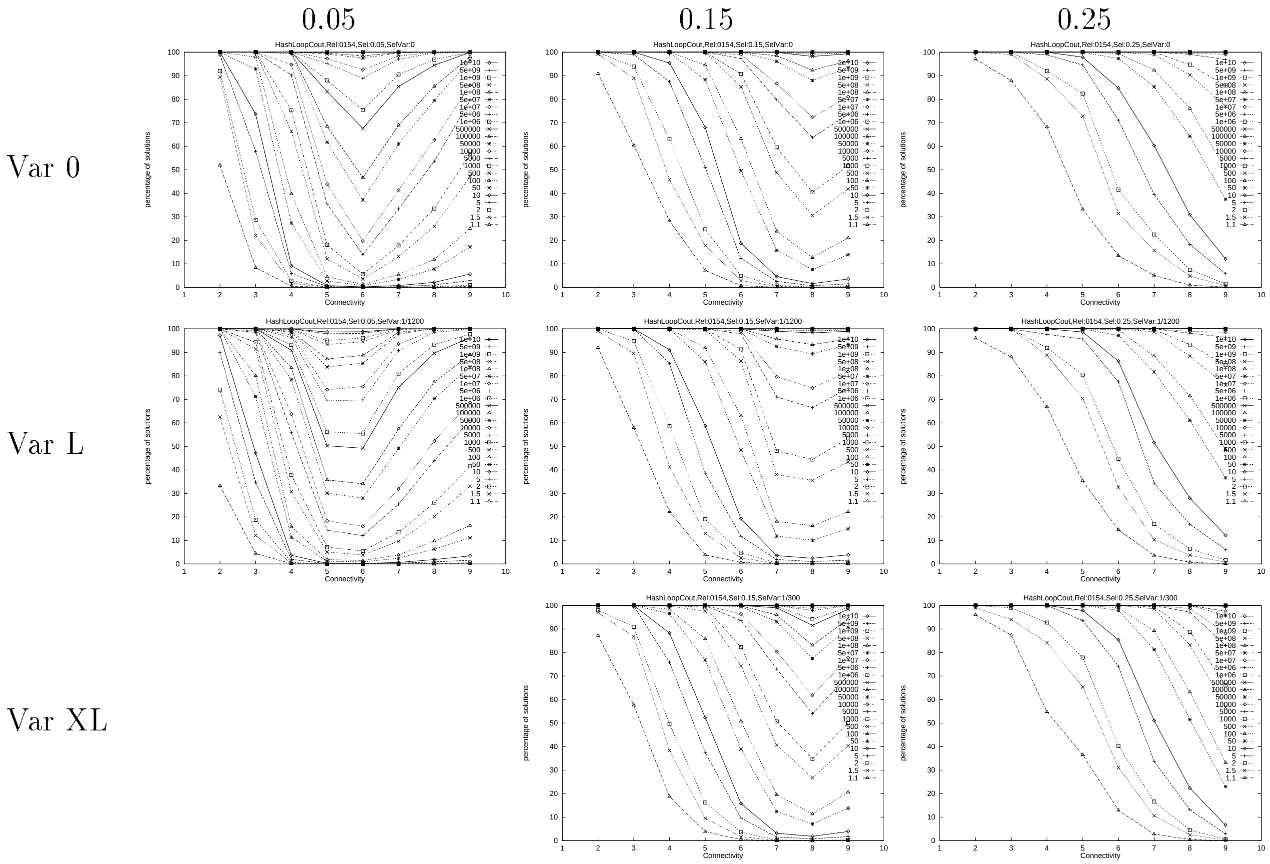

Figure 9: Search space analysis for varying selectivity parameters (HL)

6 Probabilistic Optimization Procedures

To optimization problems for which no polynomial time algorithm is known, probabilistic procedures are often applied with some success. Among the predominant algorithms used are simulated annealing (SA) and iterative improvement (II). Among the optimization problems to which these are applied are problems from many different areas including artificial intelligence and databases. In the latter area, probabilistic procedures have been applied to the join ordering problem [6, 9, 12, 18, 19].

Before we start our investigation let us fix our notion of complexity. We define the complexity of an optimization problem for a probabilistic algorithm by the number of alternatives the algorithm has to consider in order to solve the optimization problem. Since we consider probabilistic algorithms, the solution should be determined absolutely, but is to be defined in terms of some deviation of the cost of the solution found from the cost of the (global) optimum. Hence, for probabilistic optimization algorithms, an optimization problem is an instance of an ordinary optimization problem together with a percentage q of allowed deviation from the global optimum. The complexity of the problem with respect to a probabilistic algorithm is then defined by the number of nodes

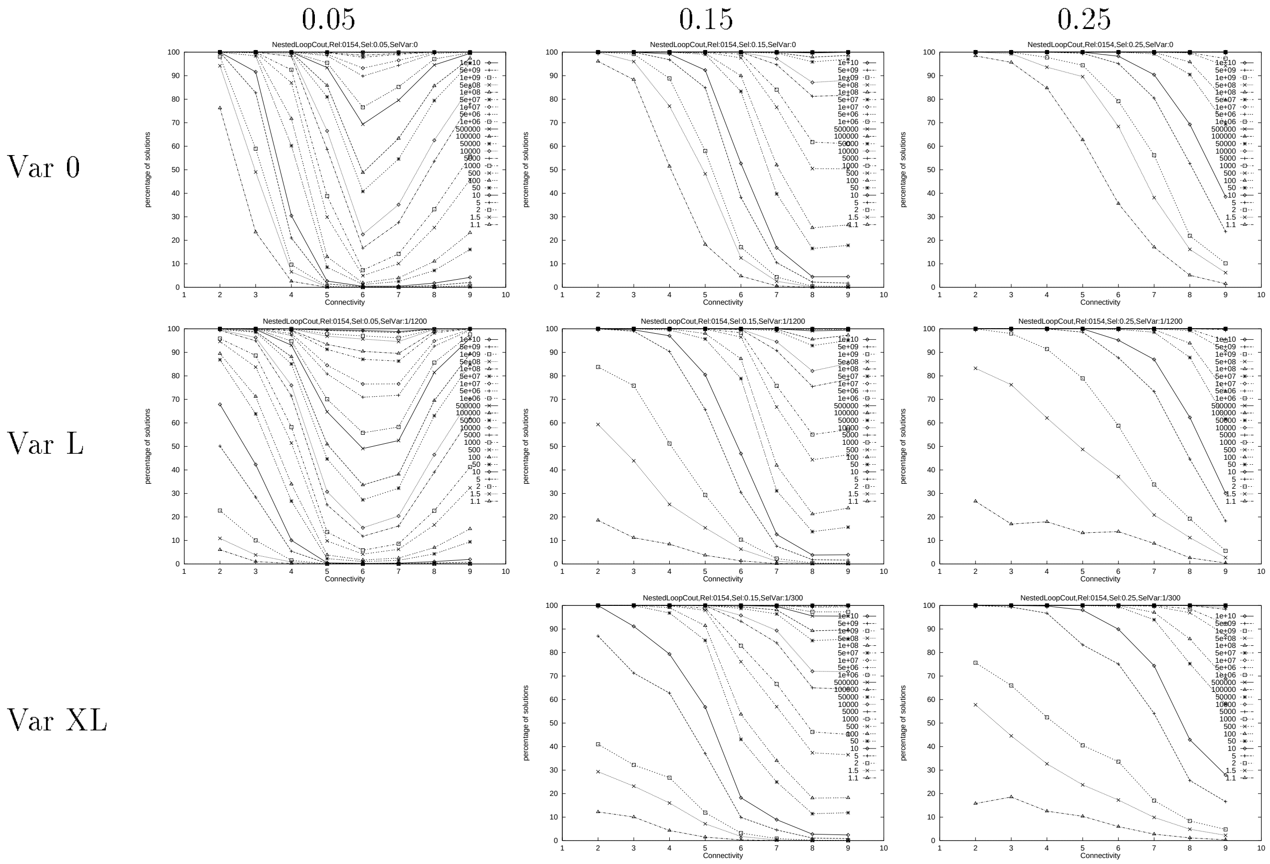

Figure 10: Search space analysis for varying selectivity parameters (NL)

the algorithms has to expand in order to find a solution whose cost is not more than q% higher than the global optimum.

Given this notion of complexity of a problem instance for a certain probabilistic optimization algorithm, we can again ask for the complexity of a given problem for a probabilistic optimization procedure and for the parameters influencing this complexity.

It has been pointed out by Ioannidis and Kang, that the structure of the search space plays an essential role for the complexity [10, 11]. If the search space forms a *well*, then the optimization is easier for probabilistic procedures. A well is essentially characterized by two features:

1. The difference in cost of two local minima is small.

2. There exists a path between two local minima whose elements have not too high a cost.

As also pointed out by Ioannidis and Kang, the search space of left deep trees is a little less appropriate than the search space of bushy trees.

However, no more details are known on the parameters which influence the complexity of a join ordering problem for SA or II. In this Section, we investigate the influence of the

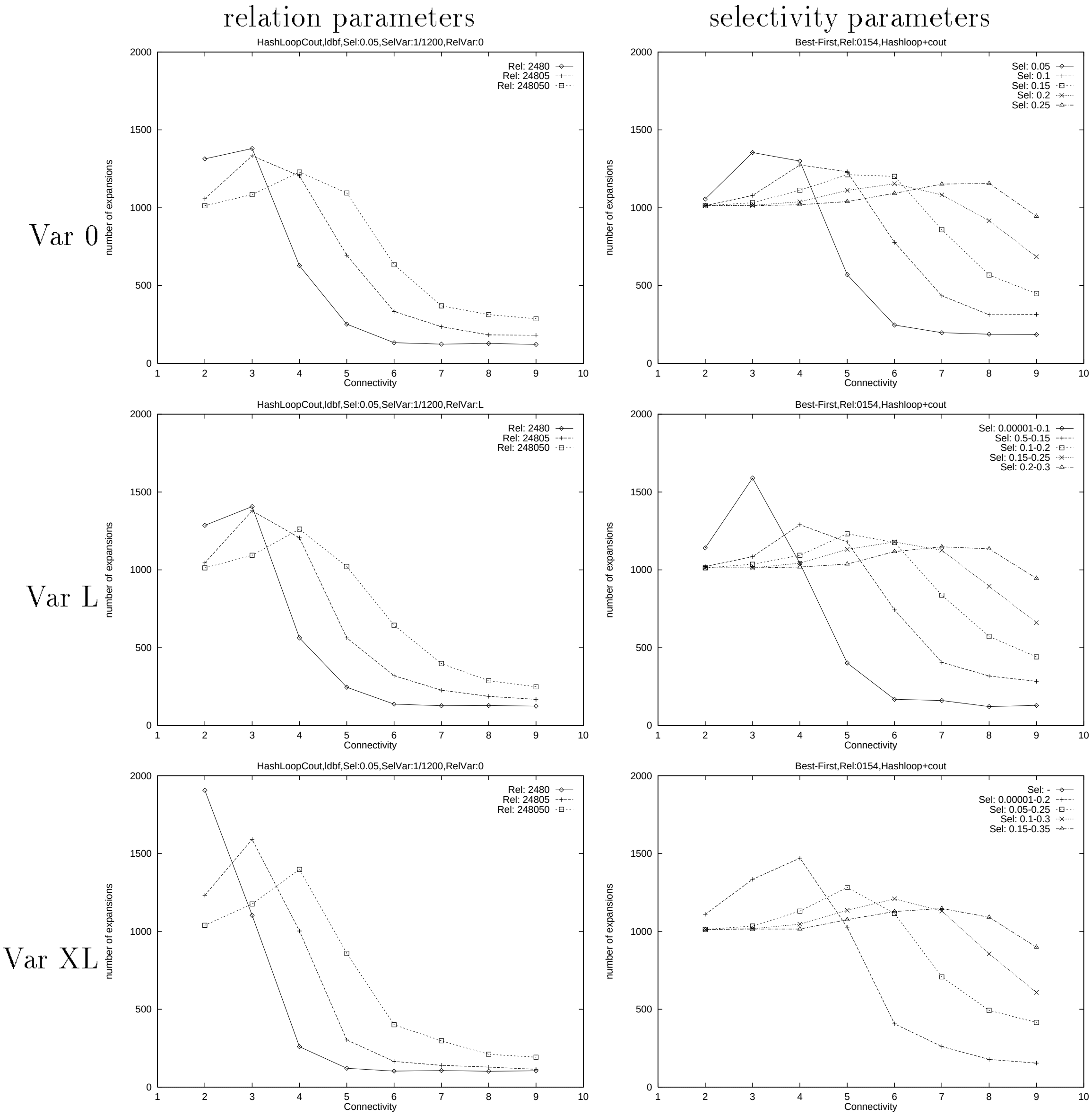

Figure 11: Impact of parameters on the search procedure

average connectivity and average selectivity on the complexity of SA and II. We assume that the readers are familiar with both, SA and II, and can imagine their application to the join ordering problem. If not, we refer to the literature [6, 9, 12, 18, 19].

Our experimental results are exemplified in Figure 13. In all cases, the algorithms had to produce a solution, which deviates at most 10% from the optimal solution. In order to get this number, we computed the optimal solution together with its cost before hand and stopped the probabilistic algorithms as soon as a first plan was encountered whose cost was within the 10% allowed deviation from the optimum. In order to get within the 10% range, the SA and II had to be carefully tuned. SA starts with a random join sequence with a starting temperature which averages the standard deviation of the costs

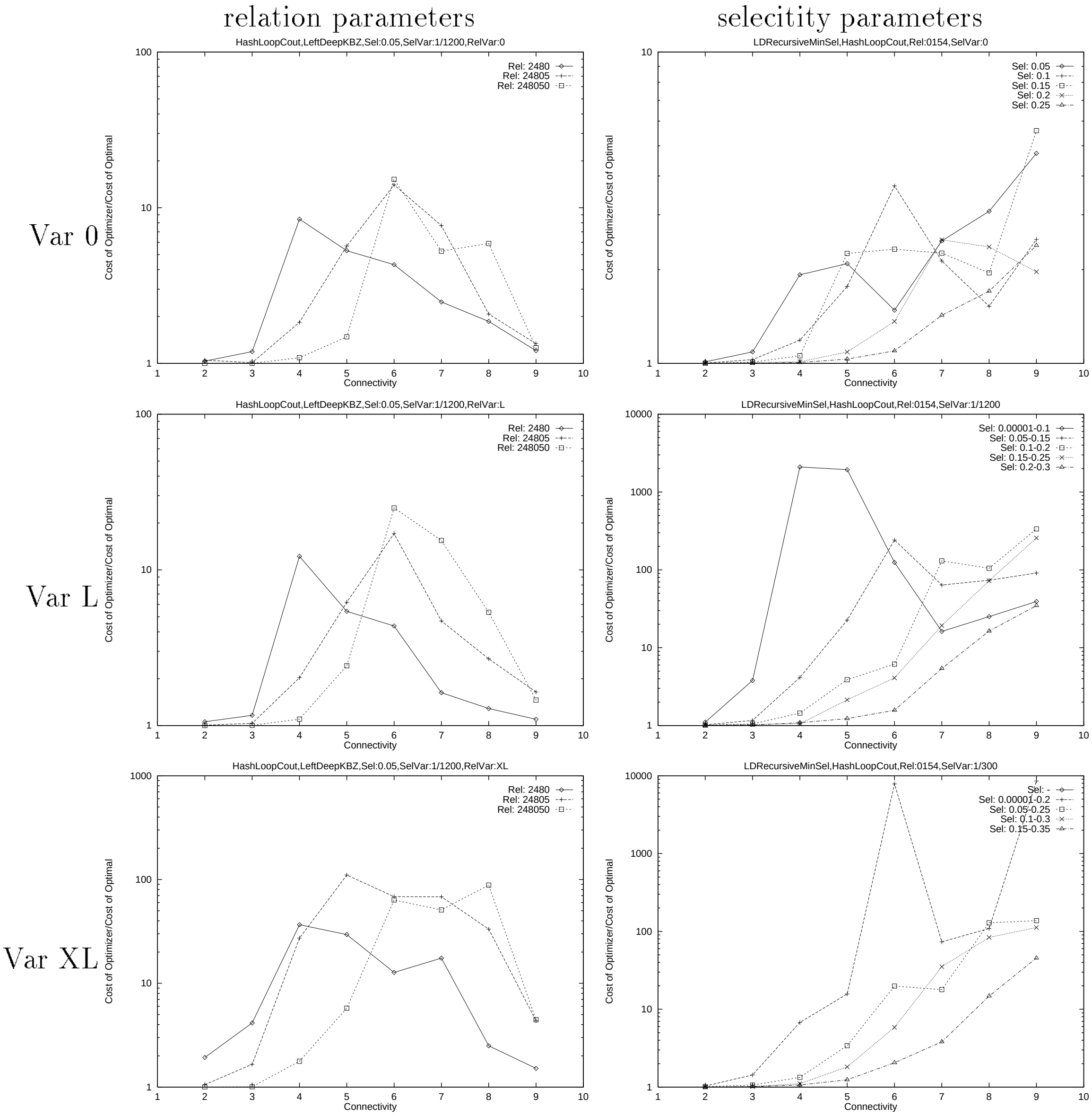

Figure 12: Impact of parameters on MinSel

of 50 random join sequences. These are not considered when counting the number of generated alternatives. At each step 25 alternatives are generated and after each step, the temperature is multiplied by 0.95. II is organized into runs. Each run starts with a random join sequence. A run stops if 500 generated random successor states of the current state fail to improve the cost.

The figure is organized as follows. The left-hand side corresponds to SA, the right-hand side to II. The curves in the top row correspond to different mean selectivities, the main rows investigate different relation variances. We consciously skipped varying the selectivity variances, since they are of no influence. Within each such picture, a single curve corresponds to a mean relation size.

We observe the following. II performs better than SA. The peaks of both, SA and II, i.e., where they exhibit their worst performance, correspond to the connectivity where the U-curves reach the phase transition. That is, the performance of both algorithms is directly tied to the shape of the search space. This nicely reflects the probabilistic nature of both algorithms.

7 Conclusion

We investigated the influence of connectivity, cost functions, selectivity, relation size and the variances of the latter two onto

- the shape of the search space, particulary on the existence and position of phase transitions
- the performance of
  - search procedures,
  - heuristics, and
  - probabilistic optimization algorithms.

The shape of the search space is clearly determined by the parameters. Since the analysis was exhaustive, this allows to predict the shape. Further, the performance of search procedures as well as probabilistic optimization algorithms is directly tied to the shape of the search space. Hence, their performance can also be predicted from the parameters. Only the heuristics show a very unstable performance which is only roughly tied to the search space. Their usage can be recommended solely for very small connectivities (2 or 3 at most).

All these findings are experimental. The main remaining challenge is to develop analytical tools to derive these findings and justify the observations. But since this is tied to one of the most challenging problems in computer science, namely the NP-P-question, fast progress in this direction seems unlikely.

Acknowledgements. We thank all our colleagues for hundreds of CPU-hours on their machines.

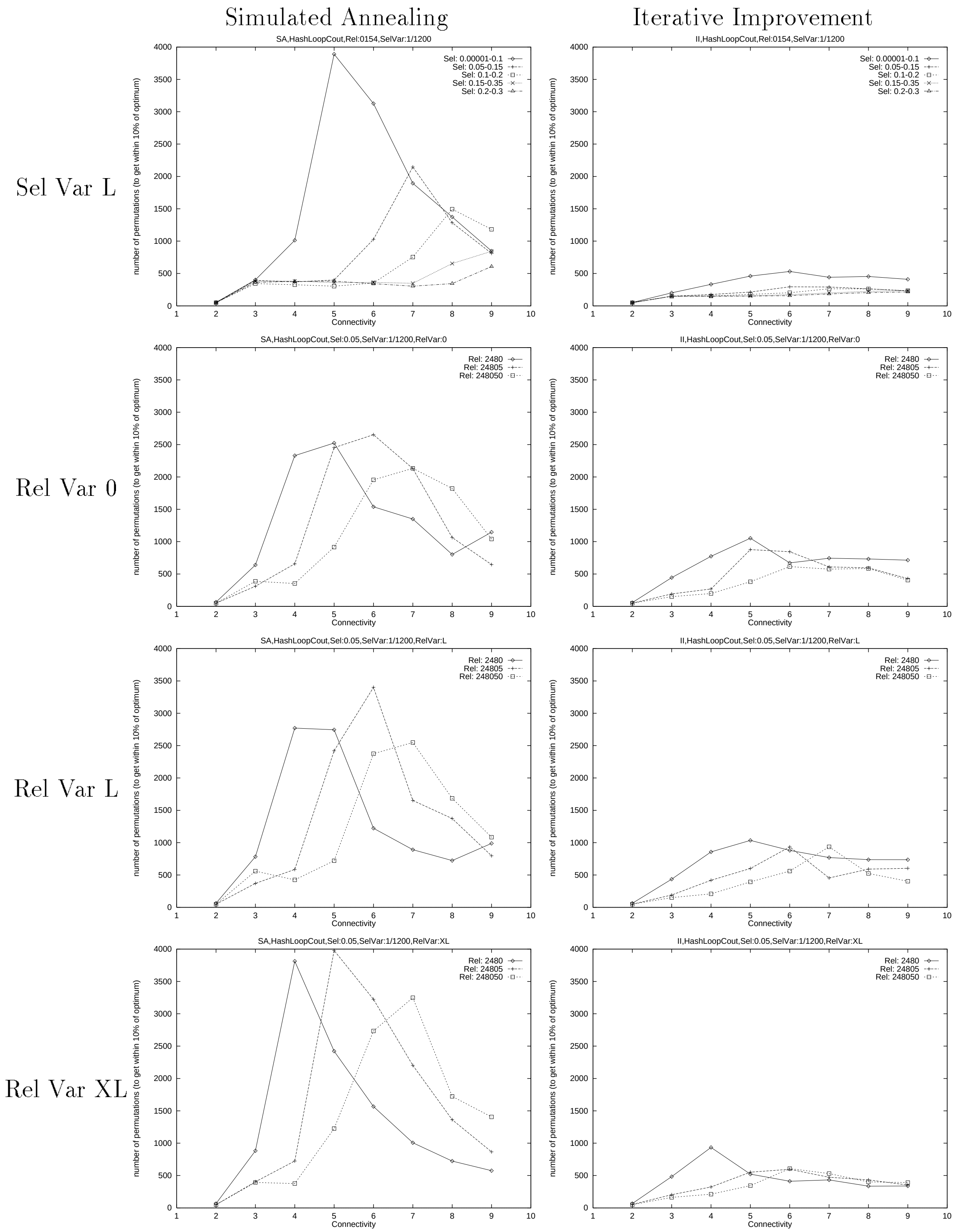

Figure 13: Impact of parameters on probabilistic procedures

References

[1] P. Cheeseman, B. Kanefsky, and W. Taylor. Where the *really* hard problems are. In *Int. Joint Conf. on Artificial Intelligence*, pages 331–337, 1991.

[2] S. Cluet and G. Moerkotte. Complexity results for producing optimal left-deep join trees. Working Paper.

[3] J. Crawford and L. Auton. Experimental results on the crossover point in satisfiability problems. In *Proc. National Conference on Artificial Intelligence*, pages 21–27, 1993.

[4] M. R. Garey and D. S. Johnson. *Computers and Intractability: a Guide to the Theory of NP-Completeness.* Freeman, San Francisco, 1979.

[5] I. Gent and T. Walsh. Towards an understanding of hill-climbing procedures for SAT. In *Proc. National Conference on Artificial Intelligence*, pages 28–33, 1993.

[6] S. Helmer, B. König-Ries, and G. Moerkotte. The relational difference calculus and applications. Technical report, Universität Karlsruhe, 1993. (unpublished manuscript).

[7] T. Hogg and C. Williams. Solving the really hard problems with cooperative search. In *Proc. National Conference on Artificial Intelligence*, pages 231–236, 1993.

[8] T. Ibaraki and T. Kameda. Optimal nesting for computing n-relational joins. *ACM Trans. on Database Systems*, 9(3):482–502, 1984.

[9] Y. E. Ioannidis and Y. C. Kang. Randomized algorithms for optimizing large join queries. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, pages 312–321, 1990.

[10] Y. E. Ioannidis and Y. C. Kang. Left-deep vs. bushy trees: An analysis of strategy spaces and its implications for query optimization. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, pages 168–177, 1991.

[11] Y. E. Ioannidis and Y. C. Kang. Cost wells in random graphs. *submitted*, 1992.

[12] Y. E. Ioannidis and E. Wong. Query optimization by simulated annealing. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, pages 9–22, 1987.

[13] R. Krishnamurthy, H. Boral, and C. Zaniolo. Optimization of nonrecursive queries. In *Proc. Int. Conf. on Very Large Data Bases (VLDB)*, pages 128–137, 1986.

[14] D. Maier. *The Theory of Relational Databases.* Pitman, London, 1983.

[15] D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions of SAT problems. In *Proc. National Conference on Artificial Intelligence*, pages 459–465, 1992.

[16] K. Ono and G. M. Lohman. Measuring the complexity of join enumeration in query optimization. In *Proc. Int. Conf. on Very Large Data Bases (VLDB)*, pages 314–325, 1990.

[17] P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. Access path selection in a relational database management system. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, pages 23–34, 1979.

[18] A. Swami. Optimization of large join queries: Combining heuristics and combinatorial techniques. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, pages 367–376, 1989.

[19] A. Swami and A. Gupta. Optimization of large join queries. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, pages 8–17, 1988.

[20] A. Swami and B. Iyer. A polynomial time algorithm for optimizing join queries. In *Proc. IEEE Conference on Data Engineering*, pages 345–354, 1993.

[21] C. Williams and T. Hogg. Using deep structure to locate hard problems. In *Proc. National Conference on Artificial Intelligence*, pages 472–477, 1992.