Principles of probabilistic query optimization
Waas, F.

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Assessing Difficulty

Since even the \texttt{xopt} problem is NP-hard, query optimization as a whole cannot be expected to be any easier. As we noted at the very beginning, the theoretical complexity is in contrast with practitioners' observation that most optimization decisions along the optimization of a query are straightforward. Roughly speaking, often only the technical details as to which implementation of a join should be used, where to deploy bit filters etc., are the more difficult ones to figure out. Of course, there is no clear division between those optimization decisions and counter examples dominated by this kind of technicalities can be easily made up.

In this Chapter we present fundamental considerations about the seeming discrepancy of theoretical intractability and the practical complexity of the query optimization problem.

To measure a problem's computational difficulty belongs to the very core of theoretical computer science. The well-known concept of NP-completeness which is based on verifiability in polynomial time and reducibility among all problems of this class emerged as a standard to assess the worst case complexity. Since its introduction, literally thousands of problems have been proven NP-complete acknowledging their computational intractability under the common assumption $P \neq NP$ [GJ79]. This concept owes its fame to both its simplicity as well as its generality. Though originally defined for decision problems that allow only a binary answer, the gap to optimization problems is easily bridged by giving a numerical bound $B$ and reformulating the optimization problem as a decision problem as to whether there is a solution having lower costs than $B$, in event of a minimization problem. Maximization problems are analogous.

Much as NP-completeness has revolutionized computing since then, it also displayed a significant drawback: it describes "only" the worst case complexity, i.e., the potential difficulty of a type of problems without taking a given instance of this problem into account. However, it seems rather natural that not all instances of an NP-complete problem are hard to solve—and it is not only the pathological cases that are an exception, like a Knapsack problem.
sack Problem with equal weights and values for all items. Consider for example a Traveling Salesman Problem where all cities are situated on a circle having equal distance from their immediate neighbors; a greedy heuristic apparently delivers the optimal tour. Still, one could argue these cases are exceptions rather than average cases. A more striking example in this field is $k$-colorability of graphs which also belongs to the class of NP-complete problems. Turner observed that a simple backtracking algorithm is able to color almost all graphs efficiently and that hard instances are seldom encountered [Tur88].

These examples suggest two important further directions to investigate: (a) hard problems may have a large number easy to solve instances, and (b) the ratio of hard to easy instances may differ significantly from problem to problem area but may yet be characteristic for a certain problem.

In the following, we briefly outline the concept of phase transition which, still an unproven phenomenon, caused a major shake-up among the artificial intelligence and machine learning communities [FA85, HH87, CKT91]; for a brief survey see [Hay97]. According to this theory, hard cases in decision problems are concentrated in a small interval of a critical order parameter. It is, however, unclear how this theory extends to optimization problems.

In a case study of the Asymmetric Traveling Salesman Problem, we contrast phase transition elements with cost distributions and analyze the influence of parameters on the difficulty of the problem. Based on these observations we then introduce the concept of the probabilistic difficulty of a problem, which assess the chances of finding high quality solutions by random sampling. We discuss the transfer of our results to query optimization and lay the foundations for an analysis of randomized optimization algorithms.

### 5.1 Phase Transitions

The original term phase transitions refers to the observation that matter commonly undergoes changes of states depending on its temperature. Exceeding a certain temperature, solid matters may liquefy, liquids may evaporate, and reverse. The important detail, however, is that those phase transitions appear within a very small range of temperature compared to the ranges of stable states.

In the field of artificial intelligence the concept of phase transitions has first been observed by Huberman and Hogg [HH87], though a deeper connection between intractability and mechanical statistics has been conjectured already earlier by Fu and Anderson [FA85]. It gained enormous popularity over the last decade and in a ground breaking paper by Cheeseman et al. [CKT91] a wider range of applications is demonstrated. They suggest that decision problems have a single, characteristic parameter that determines the difficulty of the problem. Instances where the value of
this parameter happens to be within a small critical range appear difficult, whereas instances, where this parameter is off the critical range are easy. Easy means, the problem is still of the same complexity as the hard cases but algorithms like backtracking which potentially enumerate the whole space, can solve this instances with only few corrections, running practically in polynomial time [Pur83, CKT91].

The most prominent example, receiving multi-fold attention recently, is $k$-Satisfiability which is given by a number of disjunctive clauses each of which consisting of a conjunction of $k$ variables. Each variable may also be negated. A typical instance of 3-Satisfiability is

$$(x_1 \land x_3 \land \overline{x_7}) \lor (x_4 \land \overline{x_5} \land x_7) \lor (x_3 \land x_6 \land x_7) \lor (\overline{x_2} \land x_4 \land \overline{x_5})$$

where a bar above a variable denotes negation. For $k \geq 3$, the problem is NP-complete.

In Figure 5.1, the fraction of unsatisfiable formulae out of a sample of 10000 cases is shown as a function of the ratio of clauses to variables (after [MZK+99]). Up to a value of 4 there are hardly any unsatisfiable formulae—compared to the number of satisfiable ones—encountered. Within a small interval between 4 and 4.5, this ratio suddenly rises from almost zero to almost 1. For a higher ratio almost all formulae are unsatisfiable. Note, the curve sharpens with increasing size of the sample resulting in an abrupt change from 0 to 1 at the threshold. In several further publications following the paper of Cheeseman et al., models for this threshold have been devised. However, the theoretically derived models do not match the ob-
servation, i.e., actual and expected threshold do not coincide. Finding the exact value for the phase transition is still an open problem.

The reducibility among all NP-complete problems suggests that phase transitions could be also transferred between problems. Particularly, the high stability and accuracy of this phenomenon with \( k \)-Satisfiability seem appealing to carry this concept over to other problems, separating areas of simple and difficult cases in other problems too. Moreover, using the link between decision and optimization problems, we could hope to find a similar order parameter in optimization problems.

However, the transfer to optimization problems raises a number of fundamental questions:

1. In \( k \)-Satisfiability, the order parameter determined the probability of the existence of a solution. Optimization problems do not have instances without solutions at all. What would different ranges of a phase transition parameter actually describe?

2. Difficulty in the context of a decision problem always means the difficulty to solve—i.e. obtaining a binary answer. In optimization commonly approximation are used in order to achieve reasonable trade-off between time spent on the optimization and quality of the result. What would be a phase transition with respect to approximation?

3. Assuming we can derive a measure of difficulty, how can this measure be verified? Is there a difference depending on the algorithm used, or can a phase transition be verified with any algorithm?

To answer these questions we undertake a case study with the Asymmetric Traveling Salesman Problem. In doing so, we take care to keep close contact to our original problem of query optimization by mimicking the cost distributions found there.

The Traveling Salesman Problem is a highly suitable candidate to study effects regarding the difficulty of an optimization problem for several reasons. It is easy to describe and, as we have seen above, to model, it is known to be difficult not only in the sense of NP-hard but also in terms of approximability [Aro98] therefore often serving as a test bed for novel algorithmic principles, and finally, the large number of papers published about it provide a wealth of knowledge and observations that can be used for our analysis. Specifically, in context of the analysis of phase transitions, the Asymmetric Traveling Salesman Problem has been used as an experimentation platform (see e.g. [ZP94]).

This case study was also inspired by an example taken from TSPLIB—named BR17 in the collection. Though only of size 17 it poses a major difficulty as its number of optimal solutions is very large thus exact algorithms need a considerable running time after having found an optimal solution to rule out further improvements. Evidently, this behavior is tightly coupled with the underlying cost distribution. In this case study we gradually
transform an Asymmetric Traveling Salesman Problem from the average case into this very special extreme by manipulating its cost distributions.

5.1.1 Cost Distribution

In Section 4.2.2, we analyzed the cost distribution of the euclidean symmetric variant of the Traveling Salesman Problem, given only by the coordinates of the single cities, referred to as nodes in the following. The resulting cost distribution is, even for small instances, very stable and shows only little variation from what is basically a normal distribution. The problem as is, does not offer a possibility to change this distribution gradually to one that is similar to an exponential distribution, for instance. For an experiment where we want to investigate the role of the cost distributions we therefore use the Asymmetric Traveling Salesman Problem, which is a generalization as all distances are given explicitly by e.g. a distance matrix. The cost distributions of the Asymmetric Traveling Salesman Problem can be influenced to a certain degree as we will show.

Like in the symmetric case, the cost distribution of the Asymmetric Traveling Salesman Problem converges toward a normal distribution as \( n \) approaches infinity due to the central limit theorem. However, the velocity of convergence is significantly lower than that observed in the symmetric case, which enables us to generate skewed cost distributions in the range from normal to exponential distributions.

The key to the desired skew is the edge distribution, i.e., the distribution of distances between cities (cf. 4.2.2). In contrast to the symmetric variant where these lengths were defined by the coordinates, we can now access them directly. This allows us to generate edge distributions that would be impossible when defined by coordinates in \( \mathbb{R}^2 \).

Shortest and longest tour are bound by

\[
l_{\text{min}} = \sum_{n} \min \{d_{ij} \text{ distance between nodes } i \text{ and } j\}
\]

and

\[
l_{\text{max}} = \sum_{n} \max \{d_{ij} \text{ distance between nodes } i \text{ and } j\}
\]

where \( n \) denotes the size of the instance. We use the previously introduced approximation for the mean of the cost distribution

\[
\mu = n \cdot \mu_c,
\]

with \( \mu_c \) being the mean of the edge distribution. We can achieve a skewed cost distribution by reducing the distance between \( l_{\text{min}} \) and \( \mu_c \) with respect to the total cost range. To that end, we use Gamma distributions with shape parameter \( \alpha \leq 1 \), distributions that are left of the exponential distribution (see Figure 5.2).
The quality of the bounds depends on the number of edges that are of a length close to $l_{\text{min}}$ and $l_{\text{max}}$. Using Gamma distributions with their very strong skew to the left, the lower bound is very tight, the upper bound rather loose, since we focus on left skew only, the loss of accuracy on the upper bound does no harm to our further considerations. However the fact that the lower bound is very tight is favorable as it reliefs us of the problem to determine the distance between $l_{\text{min}}$ and $\mu_d$. This is the more important, as the greedy algorithms we used in Section 4.2.2 perform poor on average in the asymmetric case.

The shape parameter $\alpha$ used in the Gamma distribution for the edge distribution serves as a measure of skew. In Figure 5.3 three pairs of edge distributions and resulting cost distributions are shown for a problem size of 50 cities. The tendency of increasing skew with decreasing $\alpha$ is clearly visible. For large $\alpha$, the cost distribution becomes more and more symmetric, for small values, it shifts increasingly to the left covering the whole range of cost distributions we found for the query optimization problem.

To avoid rounding problems when adding two numbers that are close to zero, we cut the Gamma distribution on the left edge and use only values greater than $10^{-5}$ and multiply by $\frac{1}{\mu}$. Thus, the lower bound is

$$ l_{\text{min}} \geq \frac{n \cdot 10^{-5}}{\mu}, $$

whereas the mean can be simply approximated by $n$. 

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{gamma_distributions.png}
\caption{Gamma distributions with shape parameter $\alpha \leq 1$}
\end{figure}
Figure 5.3. Skewed cost distributions for Asymmetric Traveling Salesman Problem of size 20. Left, distributions of edge lengths; resulting cost distribution right.
5.1.2 Branch and Bound

Branch and bound algorithms emerged as the standard technique for solving both Symmetric and Asymmetric Traveling Salesman Problem to optimality. Specifically for the symmetric problem, a series of advancements which make use of the geometric properties have led to more sophisticated variants like branch and cuts algorithms [GJR84, PR87, PR91]. Using these algorithms, the problem sizes which could be solved to optimality were pushed from a few hundred in the mid 1980s up to 13509 in 1998 [ABCC98], the problem we analyzed in Section 4.2.2.

However, these algorithms are not applicable to the asymmetric problem so that the largest solved problems in this case are still only the size of a few hundred nodes.

In this Section we want to determine the effects of several parameters on the run time of a branch and bound algorithm. Let us first outline the principles of this technique in some detail to provide the background necessary to understand the later results.

The Asymmetric Traveling Salesman Problem can be written as a linear program of the form

\[
\text{minimize } \sum_{i,j} d_{ij} x_{ij} \quad (1)
\]

subject to \( \sum_{j} x_{ij} = 1, \quad \forall i \quad (2) \)

\( \sum_{i} x_{ij} = 1, \quad \forall j \quad (3) \)

\( x_{ij} \) describe connected component \( (4) \)

with \( x_{ij} \in \{0, 1\}, \quad d_{ij} \geq 0 \)

(1) is the optimization objective with \( d_{ij} \) being the distance of nodes \( i \) and \( j \), and \( x_{ij} \) a boolean variable that indicates whether \( i \) and \( j \) are connected in the tour. Condition (2) and (3) demand that there is only one incoming and one outgoing edge per node, and (4) is the requirement that all selected edges form a connected component. If (2) and (3) hold, (4) implies that we have found a tour.

A relaxation of the problem is a linear program where one or more of the conditions (2)–(4) are dropped in exchange of being able to solve the problem in polynomial time. A solution of the relaxation is consequently not necessarily a valid solution to the original problem, conversely however, every solution of the original is also a solution to the relaxed problem. Furthermore, a solution to a relaxation is also a lower bound to the original problem.

A branch and bound algorithm performs a conditional enumeration using a relaxation. In case of \( L_0 \) we start with the set of shortest outgoing
edges per node. Clearly $L_0$ is a lower bound of the shortest tour. If $L_0$ is a tour, it must be the optimal tour and we are done. Otherwise, $L_0$ contains at least one edge which does not belong to the optimal solution. We select one of the edges in $L_0$ and generate two subproblems: one where this edge is contained and one where it is excluded for the rest of the search, respectively. For the restricted subproblem, we compute a new relaxation $L_0$ and proceed recursively. Whenever a subproblem forms a tour, we found a global upper bound and all subproblems whose total length exceeds the global upper bound can be discarded because further restrictions can only increase their lengths.

In Figure 5.4, an outline of the branch and bound algorithms is given. It is based on a list $Q$ that holds all the subproblems resulting from restrictions and relaxations. The function $\text{LOCALLOWERBOUND}$ computes the length of the subproblem $p$ which is the lower bound of all problems, which can be derived form $p$. The procedure $\text{FIXEDGE}$ fixates a given edge all derived subproblems will include, whereas $\text{RESTRICTEDGE}$ causes the opposite, the exclusion of the edge. After restricting, we have to compute a new relaxation, denoted by $\text{RELAX}$. The algorithm terminates as soon as there are no further subproblems in the list. For simplicity we omit additional, technical termination criteria needed if for instance no further restrictions are possible.

The solution to relaxation $L_0$ usually violates conditions (2)—(4) in that a considerable number of restrictions has to be carried out before a valid solution is accomplished. Better relaxations can be achieved by dropping only conditions (2) and (3), or condition (4). Dropping (4) immediately changes the problem into a two-dimensional assignment problem which can be solved in $O(n^3)$. We refer to this relaxation as $L_1$ in the following. We outline only the basic ideas of this technique and refer the interested reader to [Kuh55, JV86, EM92] for the details on how to solve the associated assignment problem. Condition (2) and (3) demand that any node has one incoming and one outgoing edge. We can remodel the original problem as bipartite graph $G(V, W, E)$ with one node in $V$ and one in $W$ for every original node. For every directed edge between two cities in the original problem, we add an edge between the corresponding nodes in $V$ and $W$ and assign the distance in the original graph as a weight to the edge. In Figure 5.5, an example with four nodes is shown. The left picture is the original Traveling Salesman Problem, the right is the assignment problem associated with $L_1$.

The perfect matching with minimal weight is the solution to the relaxation. In Figure 5.5, the edges belonging to the solution are indicated by thick lines. Transferring the result back to the Traveling Salesman Problem gives two unconnected subtours between nodes A,B and C,D with total length 4.
Algorithm BRANCHANDBOUND

\[ g_u \rightarrow \infty \]
\[ Q \rightarrow \emptyset \]
\[ p \rightarrow \text{solution to the initial relaxation} \]
append \( p \) to \( Q \)

while (\( Q \) not empty) do
    choose \( p \in Q \)
    \[ l_i \rightarrow \text{LOCALLOWERBOUND}(p) \]
    if \( (l_i < g_u) \) then
        if \( (p \) is tour) then
            \[ g_u \rightarrow l_i \]
        else
            choose edge \( e \) in \( p \)
            \[ p' \rightarrow \text{FIXEDGE}(e, p) \]
            append \( p' \) to \( Q \)
            \[ p'' \rightarrow \text{RELAX}(\text{RESTRICTEDGE}(e, p)) \]
            append \( p'' \) to \( Q \)
    endif
endif
done

Figure 5.4. Outline of BRANCHANDBOUND algorithm

5.1.3 Experiments

Before we describe the actual experiments and their results, we want to outline the particulars of the setup.

The aforementioned problem instance BR17 illustrates that the histogram of lengths of edges alone does not sufficiently describe the hardness of the problem. Rather the location of the single edges within the problem is an important component which is unfortunately hard if at all possible to describe. Specifically, we lack a concise description which was available with \( k \)-Satisfiability in the form of the clause-to-variable ratio.

Another point that needs to be taken care of in the design of experiments is the randomness of a test case, that is even if a problem's probability to be easy to solve is close to zero there is still the chance that the initial relaxation—specifically when using more sophisticated relaxation techniques like \( L_1 \) or 1-trees—solves the problem. The generation of problems that anticipate solvability by initial relaxation is possible, but would mean an enormous adaption to the algorithm used—thus, probably not being
5.1. PHASE TRANSITIONS

To overcome both these problems we use very large numbers of problems per configuration and eliminate outliers by using the median rather than the arithmetic mean. The downside is an vastly increased running time in general, but also the limitation to smaller instances. Problems which require several hours to solve are out of the question in order to maintain practicability of the experiments.

We chose 1000 problems per configuration together with a limitation of the total running time, a setting that proved sufficient stability in numerous preliminary trials. The limitation of the running time has to be tuned carefully for each configuration to prevent untimely termination on average, but filter out outliers only. Since we take the median instead of the arithmetic mean, the truncation does not cause negative side effects. Similar techniques are commonly used in related work; see e.g. [CKT91]. Furthermore, experiments using $L_0$ relaxations are also limited by the memory requirements which quickly exceed several GigaByte for problems of size greater than 30 nodes. These memory limitations could be overcome by using depth-first-search instead of breadth-first-search, however, the latter proved significantly more effective in that a substantially smaller number of subproblems has to be solved.

As a measure of running time we use the number of subproblems, i.e., relaxations, solved instead of the actual elapsed time. This is a further concession to practicability to run large numbers of experiments in parallel on various platforms. For both relaxations $L_0$ and $L_1$, this measure turned out to be very practicable as the times needed for single relaxations differ only marginally.

In Figure 5.6, the number of relaxations solved is plotted as a function of the shape parameter. The smaller this parameter, the stronger the skew,
i.e., the sharper the left flank of the resulting distribution.

The graphs uniformly show gradually increasing running time with decreasing shape parameter. This effect has a simple explanation by the upper and lower bounds used during the optimization. With the gradual shift of the distribution's bulk, the quality of the bounds remains largely unchanged, i.e., the number of subproblems between bounds increases. Since branch and bound is an enumeration technique—capable of exploring the whole search space in the worst case—the number of subproblems that need to be checked depends on the portion of the search space between the bounds. Accordingly, this portion and thus the running time increase with increasing skew. The graphs show scaled figures where 1 corresponds to the minimal number of relaxations solved per experiment.

5.1.4 Discussion

Let us now analyze the results from two different perspectives: in comparison with $k$-Satisfiability on the one hand and against the background of related work on the other.
5.1. PHASE TRANSITIONS

In contrast to \( k \)-Satisfiability our experiments did not show as sharp an increase in running time as was reported there [CKT91]. Their results, suggested an area where difficult cases are concentrated and on both sides of this parameter range, the expected running times for the problem are very short compared to the ones needed in the critical range.

The number of solved problems is qualitatively shown in Figure 5.7. We can identify a region of harder problems framed by two areas of apparently easier problems in the Traveling Salesman Problem. The extreme of the cost distribution skew results in a cost distribution that consists only of one single data point. Then the first relaxation is always the optimal solution and thus, the problem is easy to solve as a whole. The other area of easy problems is the area with the least skew. However, the running time in this area, indicated by \( \beta \) in Figure 5.7, increases exponentially with the problem size. The running time for the case \( \alpha = 0 \) on the other hand is always constant no matter the problem size. The two areas, though separated by a region of higher difficulty, do not correspond in the sense seen in \( k \)-Satisfiability. Rather, the case \( \alpha = 0 \) appears to be a pathological extreme.

Moreover, compared to \( k \)-Satisfiability, the Traveling Salesman Problem does not display the sharp changes from one area to the other which is characteristic for phase transitions.
Previous Work.

Another point problematic with the concept of phase transitions is the observation that different algorithms encounter different kinds of difficulty. Deploying random sampling, as used in our assessment of cost distributions in Chapter 4, now as an optimization algorithm displays this discrepancy: *The stronger the skew the higher the probability to find near optimal tours in a sample of given size.* In contrast to the experiments above, random sampling additionally provides far higher stability as it relies only on the cost distribution whereas branch and bound techniques rely to a considerable degree also on the structure of the graph. Also, the changes in running time are gradually developing.

The difficulty appears fully reversed when using random sampling as optimization technique instead of branch and bound. Consequently, we can identify different kinds of what could be interpreted as a phase transition depending on the algorithm used and whether we solve to optimality or whether we are content with an approximation.

Contrasting these two basic directions is also important when comparing to related work. Cheeseman et al. conducted a similar experiment in [CKT91] however using Little’s algorithm [Lit63] as optimization strategy, i.e. an approximation technique. Their observation is in line with what we found for bare random sampling: the more solutions there are close to the optimum the easier it becomes to find one thus the shorter the running time. They report on a precipitous decrease when they approach a level of maximal skew. However, different from the results for random sampling or branch and bound algorithms, the authors claim to have found two areas of similar difficulty separated by one of higher difficulty. They interpret their findings as phase transition in the process of reducing the deviation in the edge distribution.

Further related work concentrated on exact solutions of the Asymmetric Traveling Salesman Problem usually with variants of branch and bound techniques like Truncated and Epsilon Branch and Bound [ZP94], however without determining a threshold in the original problem, but rather devising techniques how to overcome difficult cases in enumeration trees in general.

Finding a distinct phase transition in query optimization can be expected to be as difficult as in the Traveling Salesman Problem. On the one hand, like with the Asymmetric Traveling Salesman Problem, there is no concise way of capturing the problem like with $k$-Satisfiability. On the other hand the multiplicative character of the cost function further compounds the distinction of influential parameters. König-Ries et al. undertook an attempt to identify a phase transition of the $\text{JOPT}$ only based on the connectivity of the join graph [KRHM95]. However, this approach is questionable as authors used Simulated Annealing to test for the difficulty of an instance. Simulated Annealing is known as an algorithm which heavily depends on a set of parameters thus the experiments must be strictly inter-
interpreted against the background of a particular parameter setting. And furthermore, the case of connectivity greater 1 or even up to $n$, the complete clique graph (9 in their experiments) are rarely encountered in practice. Hence, these experiments fall short of explaining easier and more difficult cases in query optimization as known from practical examples.

Our results do not rule out the possibility of the existence of a phase transition in optimization and specifically query optimization. However, the concept as found in $k$-Satisfiability is not as evident in optimization problems for several reasons:

1. problems like Traveling Salesman Problem or query optimization lack a concise description similar to the one found in $k$-Satisfiability. Rather these problems constitute several largely independent dimensions of parameters. The resulting compound problems deny the boiling down to one single parameter of difficulty.

2. There is no obvious correspondence between decision problems and approximative optimization. Assume we found a phase transition for the case where solving to optimality is concerned, it is not clear what the consequences are for an approximation. As seen above, approximation can be the easier the more difficult the exact solution is to find.

3. In optimization difficulty appears to be difficulty with respect to a certain optimization algorithm particularly in case of approximation.

Having shown the problems of transferring the concept of phase transition to optimization we abandon this area and devise an own new measure of difficulty. Instead of focusing on a possible separation of areas of difficult and easy instances we discuss difficulty from a probabilistic optimization point of view using cost distributions.

**5.2 Quality Measures**

In Section 2.2 we pointed out the uncertainties of cost computation due to the underlying statistical data about the database and further inaccuracies arising from error propagation as show in [IC91].

Solving to optimality as seen in Traveling Salesman Problem and other NP-hard problems is thus not desirable and the result optimal by cost value is not necessarily the optimal one in the actual execution. Rather there is a range of solutions that are *good enough*. Further differentiation appears not useful.

In the following, the first discuss scaling-based quality measures and their drawbacks. We develop a new notion of quality base on ranges after that.
5.2.1 Scaling-based Classification

We need a quality measure based on the actual costs value that is abstracting yet meaningful. In [Swa91], Swami proposed a possible classification. Solutions are divided into three groups: good, acceptable, and bad query plans. Plans are considered good if they have costs below twice the minimal costs $c_{min}$, acceptable if they are no more expensive than 10 times $c_{min}$, and bad otherwise.

However, this schema suffers from the severe drawback to be not invariant under additive translation. Consider the two cases shown in Figure 5.8 where two queries have very similar shaped cost distributions. In this example we assume the distribution to be an exponential distribution for both queries, i.e. $\phi(t) = e^{-t}$. The only difference between the two is a shift along the x-axis.

Let us assume that for the original distribution the cost of the cheapest plan is $c_{min} = 1$ and that the average cost of a plan is $c_\mu = 2c_{min}$. The ratio of good plans in the search space, i.e. plans with costs below $c_{good} = 2c_{min} = c_\mu$ is

$$\int_{c_{min}}^{c_\mu} \phi(t) dt$$

which normalizes to

$$\int_0^1 e^{-t} dt \approx 0.63$$
(see Figure 5.8, shaded area). We expect the ratio to keep being this way as long as the distributions are shaped similarly no matter what the actual cost values are. The previous classification, however, falls short of this invariance.

Translating the original distribution by adding a factor $2c_{\min}$ to the costs, gives $c'_{\min} = 3c_{\min}$ and $c'_{\text{good}} = 2c'_{\min} = 6c_{\min}$ for the shifted cost distribution

$$
\int_{c'_{\min}}^{c'_{\text{good}}} \phi(t) dt
$$

which is, after normalization,

$$
\int_{3c_{\min}}^{6c_{\min}} \phi(t) dt = \int_0^3 e^{-t} dt \approx 0.95
$$

(see Figure 5.8 dashed area). The ratio of good plans increased by 50% to 0.95 although the distribution stayed the same—the whole range of costs did not change either.

### 5.2.2 Range-based Classification

The cause for the insufficient valuing seen above is that only one single reference point, namely $c_{\min}$, is taken into account. To overcome this drawback, we classify plans with respect to two parameters, namely $c_{\min}$ and $c_{\mu}$.
We denote the quality of a plan \( q \) by its normalized costs

\[
T(q) = \frac{C(q) - c_{\text{min}}}{c_{\mu} - c_{\text{min}}}.
\]

The new measure is translation invariant. For the optimum, the normalized costs always equal 0 while \( T(q) \) is 1 for plans of average costs. Plans above \( c_{\mu} \) have normalized costs greater than 1, accordingly. In principle, the maximal cost value could also serve as a reference point, however, incorporating \( c_{\mu} \) into the quality measure links it automatically to the particular distribution—\( c_{\mu} \) is characteristic for a given distribution. In Figure 5.9, the areas of plans with \( T(q) \leq 0.1 \) and \( T(q) \leq 1.0 \) are shown for the same distribution as before.

In our experience high quality plans show a \( T \) of less than 0.1, although greater values are justified with respect to large join queries. Hence, the optimization goal we are aiming at is to find a plan with \( T \) below 0.1. In Figure 5.9 this target cost range is shaded.

### 5.3 Probabilistic Difficulty

Also with respect to the analysis of randomized optimization techniques in the following chapter, we develop a model of difficulty based on the nucleus of probabilistic algorithms, namely the randomly selecting of a single solution.

Randomly choosing an element from the unrestricted search space or a restricted subset of it is common to all these algorithms, be it in from of selecting an initial starting point or a larger set as it is the case with multi-start or genetic algorithms. In the further process, often subsets of the search space are used as candidates for additional random choices usually limited by means of a certain locality as in navigation-based algorithms like Hill Climbing or Simulated Annealing. The simplest of randomized algorithms using only the selection primitive without further restrictions is random sampling. For the moment we focus on uniform sampling only. In Chapter 8 we will also consider non-uniform selection schemes and discuss the differences.

In the following we develop a measure of difficulty, called \textit{probabilistic difficulty} as it reflects the difficulty to optimize by random sampling. With the probability

\[
Q(x) = \sum_{c=c_{\text{min}}}^{x} \phi(c),
\]

we could simply use the probability to hit the range of costs below a given numerical threshold \( x \)

\[
F(x) = 1 - Q(x).
\]
5.3. PROBABILISTIC DIFFICULTY

The resulting measure has values between zero and one, where zero is reached in case of \( x = c_{\text{max}} \). Though useful when computing the probability of success for repeated sampling, the measure displays a distinct disadvantage: outliers in the distribution are hard to distinguish. Consider the two distributions given in Figure 5.10 of type B1 and B2. \( F(x) \) has a value of almost zero for both distributions, thus, they would appear to be of similar difficulty. We overcome this problem by first defining the target quantile and then determining the size of the interval we can expect to hit with this probability. The size of the interval directly translates to the distance from the optimum. Given for instance a distribution like in Figure 5.11, and a target quantile of 0.5, the resulting interval is \([0, \mu]\) as shown by the shaded area in the plot since the probability to select a solution in this interval is 0.5.

Similar to the quality measures presented in the previous section we rely on reference points. In the example, we took the whole cost range but for the reasons outlined above, we use optimum and mean again. Thus, we can define a problem’s probabilistic difficulty as

\[
D(x) = \frac{Q^{-1} - c_{\text{min}}}{\mu - c_{\text{min}}}
\]

where \( Q^{-1} \) is the inverse of the previously defined \( Q \). \( D \) does not map a problem to a single value of difficulty but provides a mapping of target quantiles to the ratio of intervals.

As the size of the target quantile depends strongly on the particular problem, the size of the given instance, and factors like the uncertainty in
the cost computation. For some problems a 15% quantile can be a rather slack target, e.g. Traveling Salesman Problem, for others like query optimization it can be a narrow objective.

To give an impression how the three major problem categories compare, we show the difficulty $D$ in Figure 5.12 as a function of the size of the target quantile.

For Type-A, the plot reads as follows: e.g. the quantile of 0.1 next to the optimum spans the interval $[c_{min}; 0.1 \cdot (\mu - c_{min})]$, the quantile 0.5 spans $[c_{min}; 0.7 \cdot (\mu - c_{min})]$. The curve is clipped for values larger than 1 as larger quantiles are not of interest for an optimization anyway. The other curves read analogously. In contrast to Type-A, however, the others soar up for even small intervals meaning that a large portion of the interval $[c_{min}; \mu]$ is needed to cover the target quantile, i.e., there is no cost concentration around the optimum.

The curves reflect our earlier experiences. The partitioning problem is “easy” even for large quantiles. The Traveling Salesman Problem instance in contrast appears difficult even for small quantiles.

5.4 Summary

Query optimization is known to be NP-hard and so are restricted subproblems like JOPT and XOPT. However, the intractability attested is “only” the worst case complexity.
In decision problems like $k$-Satisfiability phase transitions have been discovered where areas of easy instances are separated by a small area of hard instances. The link between optimization and decision problems might now suggest that similar phenomena occur in optimization problems and thus in query optimization too.

Using the Asymmetric Traveling Salesman Problem as an experimentation platform, we investigated possibilities of phase transitions. Our findings show that an optimization with exact techniques encounters difficulty that has its explanation in the associated cost distribution rather than in a phase transition. The difficulty increases gradually until it collapses into the pathological trivial case where all edges, and so all tours too, are of the same length. A separation as seen in $k$-Satisfiability cannot be identified.

For approximation algorithms the situation appears even reversed: what was increasingly difficult for branch and bound techniques is growing easier when using for example random sampling. Other heuristics may display further, different regions of difficulty. Thus difficulty in optimization has be viewed with respect to the algorithm used rather than by the problem itself. While a phase transition is not apparent, cost distribution can very well interpret the effect observed. Providing also a tool to assess the chances of successful deployment of branch and bound techniques in other fields. Concerning query optimization our findings thwart the hopes for a successful deployment: On the one hand, the cost distribution is in general
not favorable to these algorithms, on the other hand, the lack of a non-trivial relaxation technique allows only weak bounds. In Section 5.1.3 we have seen the importance of sophisticated relaxations. In preliminary tests in SQL Server with cost bound pruning techniques we observed qualitatively only little gains—often outweighed by the overhead added.

In order to use cost distributions for further analysis of the query optimization problem we re-formulated the optimization goal by defining a measure of quality that overcomes the drawback of invariance under translation. Finally, we developed an alternative measure of difficulty solely based on a problem's cost distribution.

In the following, we will use the concepts we devised here to analyze randomized algorithms as well as evolutionary techniques.