LEFT-DEEP VS. BUSHY TREES: AN ANALYSIS OF STRATEGY SPACES AND ITS IMPLICATIONS FOR QUERY OPTIMIZATION †

Yannis E. Ioannidis
Younkyung Cha Kang
Computer Sciences Department
University of Wisconsin
Madison, WI 53706

Abstract

We present a combination of analytical and experimental results that shed some light into the shape of the cost function of the strategy spaces that query optimizers must deal with. These are the space that includes only left-deep trees and the space that includes both deep and bushy trees. We conclude that the cost functions of both spaces essentially form a "well" but of a distinctly different quality. Based on this result, we discuss how Iterative Improvement, Simulated Annealing, and Two Phase Optimization perform on these spaces. We conclude that the space of both deep and bushy trees is easier to optimize than the space of left-deep trees alone.

1. INTRODUCTION

Query optimization is an expensive process, primarily because the number of alternative access plans for a query grows at least exponentially with the number of relations participating in the query. The application of several useful heuristics eliminates some alternatives that are likely to be suboptimal [Seli79], but it does not change the combinatorial nature of the problem. Future database systems will need to optimize queries of much higher complexity than current ones. This increase in complexity may be caused by an increase in the number of relations in a query [Kris86], by an increase in the number of queries that are optimized collectively (*global optimization*) [Gran81, Sell86], or by the emergence of recursive queries. The heuristically pruning, almost exhaustive search algorithms used by current optimizers are inadequate for queries of the expected complexity. Thus, the need to develop new query optimization algorithms becomes apparent.

Randomized algorithms have been successfully applied to various combinatorial optimization problems. Three such algorithms have been proposed for optimizing complex queries in the past: *Simulated Annealing* [Kirk83, Ioan87, Ioan90], *Iterative Improvement* [Naha86, Swam88], and *Two Phase Optimization* [Ioan90]. Our focus in this paper is these three algorithms when applied to the optimization of select-project-join queries.

The performance and appropriateness of any of these algorithms depends heavily on the shape of the cost function of the space of alternatives that these algorithms search. The major contributions of the paper are the following. First, we identify several factors that affect the shape of the cost function of arbitrary spaces. These include the structure of the space represented as a graph (node degree and node distance), the cost distribution of all the alternatives in the space, and specialized properties of the cost functions. Second, we study the two primary spaces with which query optimizers deal and present results on their properties that affect the above mentioned factors. Hence, we are able to obtain strong indications for the shape of the cost function for these spaces. The main conclusion is that in both spaces of interest, the shape of the cost function resembles a "well", although of a distinctly different quality. Third, based on the above conclusion, we discuss the expected behavior of the above algorithms on each of these spaces. Our conjectures regarding the cost function shape in the space and the expected algorithm behavior have been verified by a series of experiments, whose results are nevertheless not presented in this paper due to lack of space. The interested reader, however, can find them elsewhere [Kang91].

In every instance of the query optimization problem, there are three types of graphs that are important. To avoid any confusion, we use three different terms for them. First, there is the *query graph* [Ullm82], which has the query relations as nodes and the joins between relations as undirected edges. In this paper, we study tree queries (i.e., queries whose query graph is a tree [Ullm82]) containing only equality joins. Occasionally, because of their interesting characteristics, special attention is given to *star* and *string* queries. In a star query, one relation participates in all joins and each of the rest participates in one join. In a string query, two relations participate in one join and each of the rest participates in two joins. Second, there is the *strategy* (or *access plan*), which is a directed tree and is defined in Section 2.1. Finally, there is the *strategy space*, which is a general undirected graph that is searched by the optimization algorithms for the node of least cost.

This paper is organized as follows. Section 2 describes the two spaces of interest and the three

† Partially supported by NSF under Grant IRI-8703592.

algorithms that are the focus of this paper. Section 3 classifies the different shapes of cost functions that affect the behavior of these algorithms. Sections 4 and 5 discuss the factors that determine the shape of the cost function of arbitrary spaces. Sections 6, 7, and 8 contain a combination of analytical and experimental results that shed some light into the properties of the spaces of interest to query optimization. Section 9 uses the results of the last five sections to draw conclusions on the shape of the cost functions of these spaces. Section 10 analyzes the behavior of the three algorithms according to the shape of each space. Section 11 compares the spaces themselves and suggests one of them as the most appropriate for query optimization. Section 12 presents the results of our previous work [Ioan90] and that of Swami and Gupta [Swam88] in light of the findings in this paper. Finally, Section 13 concludes and presents some directions for future work.

2. RANDOMIZED QUERY OPTIMIZATION ALGORITHMS

Each solution to a combinatorial optimization problem can be thought of as a *state* in a space, i.e., a node in a graph, that includes all such solutions. Each state has a cost associated with it, which is given by some problem-specific cost function. A state of low (high) cost is a *low (high) state*. The goal of an optimization algorithm is to find a state with the globally minimum cost. Randomized algorithms usually perform *random walks* in the state space via a series of *moves*. The states that can be reached in one move from a state S are called the *neighbors* of S. Their number is the *degree* of the state. A move is called *uphill (downhill)* if the cost of the source state is lower (higher) than the cost of the destination state. A state is a *local minimum* if, in all paths starting at that state, any downhill move comes after at least one uphill move. It is a *global minimum* if it has the lowest cost among all states. A state is on a *plateau* if it has no lower cost neighbor and yet it can reach lower cost states without uphill moves.

Using the above terminology we describe three randomized optimization algorithms that are of interest to query optimization. Due to lack of space, the descriptions are very brief. Details can be found elsewhere [Ioan90].

Iterative Improvement (II) performs a large number of *local optimizations*. A local optimization starts at a random state and improves the solution by repeatedly accepting random downhill moves until it reaches a local minimum. Its output at the end is the least cost local minimum that has been visited.

Simulated Annealing (SA) starts at a random state and proceeds by random moves, which if uphill, they are only accepted with certain probability. As time progresses this probability gradually decreases until the time it becomes zero, which signifies the end of the algorithm. The output of the algorithm is again the least cost state that has been visited.

Two Phase Optimization (2PO) is divided into two phases. In the first phase, II is run for a small period of time, i.e., a few local optimizations. The output of that phase is the initial state of the next phase, where SA is run with very low initial probability for uphill moves.

When the above generic optimization algorithms are applied to query optimization, three parameters need to be specified: the state space, the neighbors function, and the cost function.

2.1. State Space

Each state in query optimization corresponds to a *strategy* (or *access plan*) of the query to be optimized. Hence, in the sequel, we use the term strategy and state indistinguishably. Using the heuristics of performing selections and projections as early as possible and excluding unnecessary cartesian products [Seli79], we can eliminate certain suboptimal strategies to increase the efficiency of the optimization. Thus, we reduce the goal of the query optimizer to finding the best join order, together with the best join method for each join. In this case, each strategy can be represented as a *join processing tree*, i.e., a tree whose leaves are base relations, internal nodes are join operators, and edges indicate the flow of data. If all internal nodes of such a tree have at least one leaf as a child, then the tree is called *deep*. Otherwise, it is called *bushy*. Most join methods distinguish the two join operands, one being the *outer* relation and the other being the *inner* relation. A *left-deep tree* is a deep tree whose inner relations of all joins are base relations. In this study, we deal with two strategy spaces: one that includes only left-deep trees, which is denoted by L, and one that includes both deep and bushy ones, which is denoted by A.

2.2. Neighbors Function

The neighbors of a state, which is a join processing tree (i.e., a strategy), are determined by a set of transformation rules. Each rule is applied to some internal nodes of the state, replaces them by some new nodes, and usually leaves the rest of the nodes of the state unchanged. There are several sets of transformation rules from which one could choose, and the options are different for the spaces A and L. The ones adopted in this study are described below.

Let a, b, and c be join processing formulas and assume for simplicity that joins are signified by concatenation. The set of transformation rules in A is given below [Ioan90]:

(1)	*Join method choice:*	a ($meth_i$) b $\rightarrow$ a ($meth_j$) b
(2)	*Join commutativity:*	a b $\rightarrow$ b a
(3)	*Join associativity:*	(a b) c $\leftrightarrow$ a (b c)
(4)	*Left join exchange:*	(a b) c $\rightarrow$ (a c) b
(5)	*Right join exchange:*	a (b c) $\rightarrow$ b (a c)

Strategies in L can be represented as a sequence of relations, which signifies the order in which the relations are joined in the strategy. Let a, b, c, and d be relation sequences, and R, S, and T be single relations. The set of

transformation rules in L is given below [Swam88]:

(1)	*Join method choice:*	a ($meth_i$) R → a ($meth_j$) R
(2)	*Swap:*	a R b S c → a S b R c
(3)	*3-cycle:*	a R b S c T d → a S b T c R d

In both spaces, rule (1) is only applicable when multiple join methods are available in the system and simply changes the join method of a join operator, e.g., from nested-loops to merge-scan.

2.3. Cost Function

In our study, we used several models for the cost functions of database operations to observe their effect on our findings. The details of each model are not presented here due to lack of space and because the results were insensitive to the specific cost model. We should mention, however, that all three join methods, nested-loops, merge-scan, and hash-join were used, and that several interesting combinations of them were tried in the various cost models.

3. SHAPES OF COST FUNCTIONS

Hereafter, we refer to the shape of the cost function of a strategy space as the shape of the space. Also, consider two low local minima S_1 and S_2 in some space, and let P be the set of paths that connect them. For each path $p \in P$, let N_p be the set of strategies in the path excluding S_1 and S_2. The *connection cost* of S_1 and S_2 is equal to

$$\min_{p \in P} \max_{s \in N_p} \{c(s) \mid s \in N_p\}. \quad (3.1)$$

There are two parameters of the shape of some strategy space that determine the behavior of the above optimization algorithms:

(A) the cost distribution of local minima, and

(B) the connection cost of low local minima.

The question of interest in (A) is what percentage of the local minima are close to the cost of the global minimum. The question of interest in (B) is whether or not, for the low local minima, their connection costs are much higher than their costs. Based on the cost distribution of local minima, shapes can be placed in three categories that can be defined qualitatively as follows.

A1 Most local minima are low.

A2 A nontrivial percentage of local minima are low, but most of them are high.

A3 Local minima are scattered at all costs.

Based on the connection cost of low local minima, shapes can be placed in three categories as well.

B1 The connection cost is similar to the cost of the local minima, i.e., the area of low cost strategies is smooth.

B2 The connection cost is somewhat distant from the cost of most low local minima, but still relatively low compared to the cost range in the whole space, i.e., the area of low cost strategies is bumpy.

B3 The connection cost is much higher than the cost of the local minima, i.e., there are multiple areas of low cost strategies, which are separated by high "hills".

This classification leads to the following *qualitative* definition of a *well*. Any cost function whose shape is of type Ai-Bj, with $i,j \in \{1,2\}$, forms a well. In later sections, it is shown that the behavior of the algorithms depends not only on whether the cost function forms a well or not, but also on the specific well class to which it belongs.

4. COST DISTRIBUTION OF LOCAL MINIMA

This section identifies three aspects of a strategy space that affect the cost distribution of its local minima: the degree of the strategies in the space, the range of the cost difference between neighbors, and the cost distribution of all strategies. The results presented below hold for spaces that are generated randomly. Clearly, such spaces are not accurate models of A or L. Nevertheless, although these results cannot serve as proofs of what is happening in the spaces of interest, they are still valuable in providing some intuition that can help explain the experimental observations.

In what follows, in order to compare cost distributions, we need the following definitions. For a random variable X, its distribution function is denoted by F_X and is defined as $F_X(x) = \text{Prob}(X \le x)$. Consider two random variables X and Y on the same range $[m,M]$ of possible values. Then, the distribution of Y is *shifted to the left relative to* the distribution of X if $F_Y(x)/F_X(x) > 1$ for all $x < M$. The ratio $F_Y(x)/F_X(x)$ is the *relative shift* of the distribution of Y over the distribution of X at point x. If the random variables do not have the same range, then the relative shift of their distributions can be defined after scaling them so that their ranges become equal.

Consider an arbitrarily large space and let the strategy costs be assigned based on some probability distribution. Also, let each strategy have d neighbors, which are chosen randomly among all other strategies. Then the following holds.

Theorem 4.1: Let X be the random variable for strategy costs and Y be the random variable for local minimum costs. Then, $F_Y(x) \approx 1-(1-F_X(x))^{d+1}$.

As with all other analytical results in this paper, in the interest of space, we omit their proofs. Note that the above theorem holds for arbitrary cost distributions in the strategy space. Due to the exponential form of the formula in the theorem, we see that with high enough degree, the local minimum cost distribution is very much shifted to the left relative to the cost distribution of all strategies. This is clearly exemplified in Figure 4.1a, where $F_X(x)$ is drawn as a function of $F_Y(x)$ for various values of d. Even with degree d=10, 50% of the local minima are found among

the lowest 7% of all strategies, and this drops to 2% of all strategies for d=50.

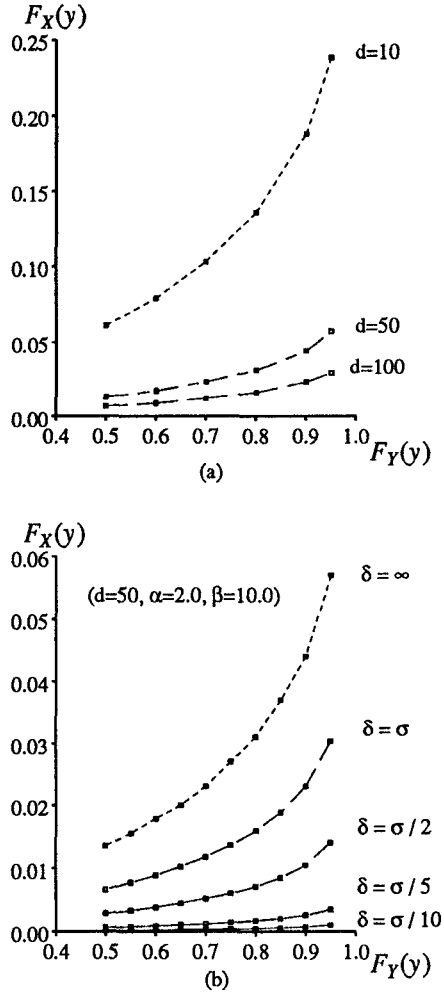

Figure 4.1: The relationship of the strategy cost and local minimum cost distributions: (a) $\delta=\infty$; (b) d=50.

Next, we enhance the above graph model to capture constraints on the cost difference between neighbors. Specifically, consider the additional constraint that the d neighbors of a strategy with cost c are chosen randomly among all strategies whose cost is between $c-\delta$ and $c+\delta$, for some threshold δ. Unfortunately, for this case, it is not possible to capture the relationship between F_X and F_Y in closed form independent of the probability distribution. Thus, to obtain an understanding of the effect of δ, we have experimented with several values of δ and with several forms of the Γ distribution. The choice of the Γ distribution was motivated by experimental results that showed that it resembles the distribution followed by the cost of strategy spaces in query optimization (Section 7). The conclusion from the experiments is that, in general, the relative shift of the local minimum distribution to the left increases as δ decreases, i.e., local minima tend to be of lower cost. Occasionally, for very low degree d and at high values of $F_Y(x)$, we observed a reversal of this trend, i.e., the relative shift started decreasing as δ decreased beyond a certain point. Overall, however, these cases were few. As a typical example, Figure 4.1b shows $F_X(x)$ as a function of $F_Y(x)$ for the Γ distribution with parameters α=2 and β=10 [Roth86], and for d=50. Diagrams for many values of δ expressed as a fraction of the standard deviation σ of the used Γ distribution are shown. The above mentioned trend is obvious in these diagrams. Hence, in general, less abrupt changes in cost between neighbors result in less local minima among higher cost strategies.

Another interesting outcome of both experiments above is that as the cost distribution of all strategies changes, the local minimum cost distribution follows along. More specifically, if the distribution of X is shifted to the left then the distribution of Y is shifted to the left as well. This establishes the rather intuitive fact that, as more strategies move closer to lower costs, more local minima do the same as well.

5. CONNECTION COST BETWEEN LOW LOCAL MINIMA

This section identifies three aspects of a strategy space that affect the connection cost of low local minima: the distance between the local minima and the number of paths connecting them, the cost distribution of all strategies, and some special properties of the functions used to compute the cost of each strategy.

The effect of the first factor is rather intuitive although difficult to express formally. As the distance of two local minima increases and/or the number of paths connecting them decreases, their connection cost most likely increases as well. The intuition behind the above statement is based on equation (3.1). Longer paths contain more strategies (N_p is larger) and therefore tend to be able to reach at higher costs (there are more choices for the maximum of the costs of strategies in N_p). By the same token, fewer paths (smaller P) implies that there are fewer choices for the minimum among those maxima. Although there is no concrete relationship between distance/number of paths and connection cost, there is a rather intuitive trend. As two extreme examples, consider a fully connected graph with N nodes and a string of N nodes (a cycle with one of its edges removed). In the first graph, every pair of strategies is connected with a path of length 1, and the total number of paths is approximately $e\,(N-2)!$, where e is the Napierian number. The connection cost for all pairs of strategies is zero, since they are neighbors. In the second graph, for every pair of strategies, there is a unique path that connects them. Moreover, there exist strategies whose connecting path contains all nodes in the graph and therefore their connection cost is the highest possible. Most graphs fall between these two extremes and behave analogously. Note that the degree of a graph affects both the distance between two strategies and the number of paths between them. A higher degree, decreases the former and increases the latter, thus decreasing the connection cost of strategies.

The effect of the second factor is also straightforward. As the percentage of strategies having low cost increases, the connection cost of low local minima

decreases. The intuition behind the above statement is again based on equation (3.1). The further to the left the strategy cost distribution is shifted, the lower the costs will be in the set $\{c(s) \mid s \in N_p\}$.

The effect of the third factor is much more complex and is based on previous work [Monm79, Ibar84, Kris86]. Consider a strategy space whose nodes can be mapped to the distinct permutations of a set of symbols Σ. In addition, for every strategy S, let its neighbors include all strategies obtained from S by interchanging two adjacent subsequences of S. For example, if S corresponds to the permutation *abcdef*, the strategy that corresponds to *deabcf* is a neighbor of S. Let A, B, U, and V be sequences of symbols in Σ, with U and V not being null. The strategy cost function satisfies the *Adjacent Sequence Interchange* (ASI) property if the following holds: $c(AUVB) \leq c(AVUB)$ iff $rank(U) \leq rank(V)$, for some function *rank*. The importance of the ASI property for the more traditional approaches to query optimization has been discussed before [Ibar84, Kris86]. Its importance for randomized algorithms becomes evident with the following theorem.

Theorem 5.1: Consider a strategy space as described above, whose cost function satisfies the ASI property. Then the space has a unique area of local minima that are connected among themselves, which are therefore global minima.

Whenever the premises of the above theorem are satisfied by a strategy space, its conclusions suggest an extremely efficient optimization algorithm: execute one local optimization of II. Although neither A nor L satisfy the premises of Theorem 5.1, they contain many subspaces that do satisfy them, so that its conclusions are quite important in understanding the shape of their cost function.

6. STRUCTURE OF THE STRATEGY SPACE

The previous two sections presented several results on the properties of a strategy space that determine the shape of its cost function by affecting the (A) and (B) parameters. This and the following two sections contain results that characterize the A and L spaces with respect to these properties. Specifically in this section, we present results on the degree of strategies and the interstrategy distance in these spaces, which affect both (A) and (B). We should point out that L and A are examined with respect to the structure of the strategies alone, independent of other additional characteristics with which the spaces can be enriched, i.e., multiple join methods or indices. Incorporating these is straightforward.

6.1. Space of Left-Deep Strategies

For the L space, we have considered only the Swap rule (rule (2) in Section 2.2.2). Any neighbor of a strategy that is produced by applying the 3-cycle transformation rule is reachable with two consecutive applications of the Swap rule as well. Hence, extending the forthcoming results to the complete space is straightforward.

6.1.1. Degree of Strategies

The strategy space that corresponds to an arbitrary query is rather hard to analyze. We focus our attention on queries in whose query tree all non-leaf nodes have the same degree. That is, these are queries where each relation participates in either one join or g joins, for some constant g. Let L_g denote the space that corresponds to such a query. For such queries the following theorems hold.

Theorem 6.1: Consider a strategy in L_g for a query with J joins. If d is the degree of the strategy, then

$$d \leq \frac{J-1}{2}\left[J-\frac{J-g}{g-1}\right]+1.$$

Theorem 6.2: Consider the space L_g for a query with J joins whose query tree has a root node whose distance from all leaves is Δ. Then, there exist strategies in that space whose degree d satisfies

$$d \leq \frac{(g-1)^{\Delta-1}(g-2)(\Delta g-1)}{2}+g,$$

i.e., for a given g, $d=O(J\log_g J)$.

The general conclusion from the above theorems is that, in the L space, the degree of strategies is not always quadratic in the number of joins. Although high for many strategies in L, the degree of several other strategies in the same space is rather low.

Example 6.1: As an example, consider star queries, which correspond to the space L_J. By Theorem 6.1, all strategies in L_J are of degree $\leq J(J-1)/2+1$. In fact, for that space, all strategies have the same degree, which is equal to the given upper bound. As another example, consider string queries, which correspond to the space L_2. By Theorem 6.1, all strategies in L_2 are of degree that is less than J. By Theorem 6.2, there exist strategies of degree 2, which in fact is the lower bound on the degree as well. □

6.1.2. Interstrategy Distance

The interstrategy distance is a rather difficult parameter of a space to analyze, since it can be quite different for different pairs of strategies. The following results provide some bounds on interstrategy distances and hold for the L space in general.

Theorem 6.3: Consider the space L for a query with J joins. All strategies in that space are connected with a path of length less than $J(J+1)/2$.

Theorem 6.4: Consider the space L for a query with J joins. Let r be the length of the longest path in the query tree whose relations appear in reverse order in two strategies. Then, any path between the two strategies is longer than $r(r+1)/2-2$.

The implications of the above theorem are that the distance between strategies is rather short, quadratic in the number of joins in the worst case. Moreover, there do exist strategies whose distance has a quadratic lower bound, so

the results of these theorems are tight.

Example 6.2: Consider again star queries. In the corresponding strategy space, any pair of strategies is connected with a path of length less than J. This space is the one with the closest connections between strategies. For string queries, by Theorem 6.4, there exists a pair of strategies whose shortest path connecting them is of length $J(J+1)/2-2$. □

6.2. Space of Deep and Bushy Strategies

The placement of edges in space A is a much more tightly controlled process than in L, and this makes the analysis of its characteristics easier as well.

6.2.1. Degree of Strategies

Theorem 6.5: Consider the space A for a query with J joins. Then, the degree of any strategy in that space is equal to $2J-1$.

Theorem 6.5 comes in interesting contrast to Theorems 6.1 and 6.2. It shows that in A, all strategies have the same degree, whereas in L, strategies have varying degrees, which can be higher or lower than the degree of the corresponding A space.

6.2.2. Interstrategy Distance

Theorem 6.6: Consider the space A for a query with J joins. All strategies in that space are connected with a path of length less than $J(J+1)/2$.

Comparing Theorems 6.3 and 6.6, we see that the maximum interstrategy distance in the two spaces for all queries is the same. We expect, however, that in general, interstrategy distance is shorter in L than in A.

7. COST DISTRIBUTION OF ALL STRATEGIES

Let the catalogs of the database contain information on the following relation characteristics: relation size, number of unique values per attribute, and primary and secondary indices on the relation. If we make the usual uniformity and independence assumptions for the values in the relation attributes, these characteristics are enough to compute the cost of any equality join query on these relations.

To obtain an understanding on the strategy cost distribution for tree queries, we have experimented with several types of catalogs and cost models for both strategy spaces. For several queries, we generated 500,000 random strategies in the space of each query to obtain an approximation of its distribution. Due to the presence of strategies with cross-products, generating truly random strategies in spaces of random tree queries is extremely time consuming. Hence, we have confined ourselves in experimenting with star queries alone, which do not present similar difficulties. The results for all tested queries and for all cost models in both strategy spaces have been very consistent. As the variance in the values of the relation characteristics in the catalogs increases, two important parameters of the distribution change. First, the cost range increases dramatically. Second, the distribution shifts further to the left. Example distributions for two cases corresponding to catalogs with low and high variance in their contents are shown in Figure 7.1. The x-axis represents the ratio of the strategy cost over the least such cost among the sampled strategies. The specific case is for a 20-join query and for the A space. Similar trends were observed in all other experiments. Note the strong resemblance with the Γ distribution with parameters $\alpha = 1$ and 2 [Roth86]. This justifies our experimentation with these distributions in Section 4.

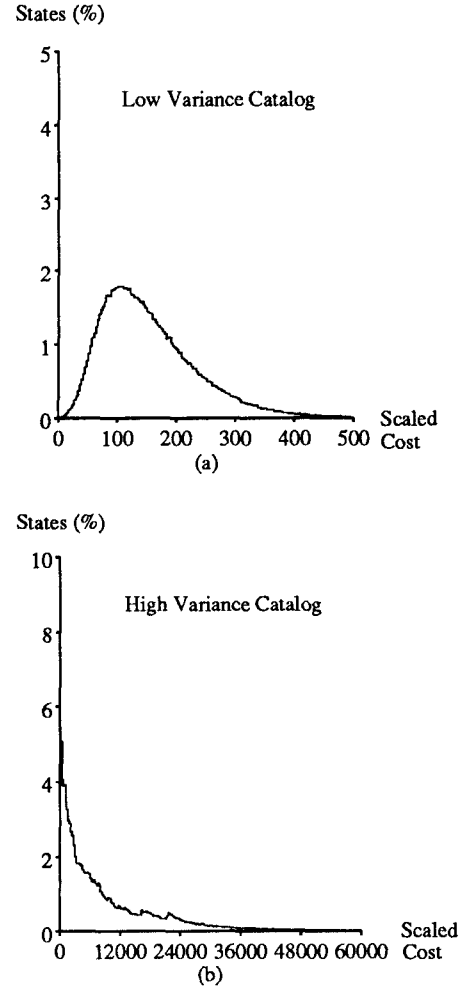

Figure 7.1: The strategy cost distribution of a star query.

The observed trends can be explained as follows. Clearly, a variance increase in the characteristics of the database relations results in opening the gap between the low and the high cost strategies. For example, if all relations have the same characteristics, all strategies will have the same cost. On the other hand, as the difference in characteristics grows, choosing the wrong order of joining the relations will be of much higher cost than choosing the correct order.

The above fact partly explains the change in the amount of shifting to the left as well. Consider a set of

strategies whose bottom part[†] is identical. Their cost difference is determined by the cost difference in the remaining joins, but is affected by the size of the results of the common part. For low cost strategies, the common bottom part usually produces small relations, whereas for high cost strategies, it produces large ones. Small relations allow a relatively small range in the cost of the subsequent operations, whereas large ones allow larger differences. This has the effect that the strategy cost distribution is in general shifted to the left. As the variance in the characteristics of the relations increase, the overall cost range increases as well, and the distribution is shifted further to the left.

8. INTERACTION BETWEEN ASI AND NON-ASI COST FUNCTIONS

A *ranked* join method is defined to be one whose cost formula is of the form $n_1 g(n_2)$, where n_1 and n_2 are the sizes of the outer and inner relations respectively, and $g(.)$ is some appropriate function [Kris86]. Suppose that R is a relation in some tree query. In the L space that corresponds to the query, consider its subspace whose nodes are the strategies having R as their first (leftmost) relation. It is well known that, if all available join methods are ranked, the strategy cost function for that subspace satisfies the ASI property [Kris86]. Hence, if exchanging adjacent sequences of relations in a strategy was a legal transformation to generate neighbors, by Theorem 5.1, that subspace would have a unique area of local minima. Although this is not the case in L, for star queries, exchanging any pair of adjacent relations is legal (assuming that the relation in the center of the star remains the first relation in the strategy). This fact can be used to prove several interesting properties regarding the connection cost of strategies in both L and A for star queries.

8.1. Space of Left-Deep Strategies

Theorem 8.1: Consider a star query with J joins and a pair of strategies S_1 and S_2 in the corresponding L space. Then the following hold:

(a) If all available join methods are ranked, there exists a path connecting S_1 and S_2 whose most expensive strategy is either S_1 or S_2 or one of their neighbors.

(b) If non-ranked join methods are available, there exists a path connecting S_1 and S_2 whose most expensive strategy is at most $J+1$ steps apart from S_1 or S_2.

Considering the huge size of the strategy space, one realizes that the above linear bound is extremely small, especially since all steps mentioned in (b) correspond to applying the join method choice rules, which do not affect the strategy structure. Hence, by Theorem 8.1, the connection cost of low local minima should in general be low as well.

[†] Recall that each strategy corresponds to a processing tree. For star queries, at least one of the children of any internal node in the strategy is a base relation. Hence, a common bottom part implies that some number of the first relations to be joined are the same.

8.2. Space of Deep and Bushy Strategies

Theorem 8.2: Consider a star query with J joins and a pair of strategies S_1 and S_2 in the corresponding A space. Then the following hold:

(a) If all available join methods are ranked, there exists a path connecting S_1 and S_2 whose most expensive strategy is at most J steps apart from S_1 or S_2.

(b) If non-ranked join methods are available, there exists a path connecting S_1 and S_2 whose most expensive strategy is at most $2J$ steps apart from S_1 or S_2.

(c) If merge-scan is the only available non-ranked join method, there exists a path connecting S_1 and S_2 whose most expensive strategy is at most J steps apart from some strategy that is connected to S_1 or S_2 with an equal cost path.

The implications of Theorem 8.2 for A are very similar to those of Theorem 8.1 for L. Essentially, the general conclusion is that for both spaces the connection cost among low local minima is relatively low as well.

9. THE SHAPE OF THE COST FUNCTION OF STRATEGY SPACES

In this section, we use the whole set of the results presented in Sections 4 to 8 to identify the shape of the cost function of the spaces A and L. We want to emphasize again that, except for very few special cases, no definite results can be proved for this problem. There is enough evidence, however, based on which useful conclusions can be drawn. These conclusions are presented in this section and have been verified by a series of experiments, whose results are not presented here due to lack of space, but can be found elsewhere [Kang91].

9.1. Space of Left-Deep Strategies

The characteristics of the L space are as follows. Most strategies in the space have a high degree (Theorem 6.1), so the local minimum distribution is shifted to the left relative to the cost distribution of all states. On the other hand, there are several strategies with relatively few neighbors (Theorem 6.2) and the cost difference between neighbors can be high. Hence, given also that the strategy cost distribution is in general shifted to the left (Section 7), from our analysis in Section 4, most likely the shape is of type A2.

Because of the relatively high degree of strategies, the interstrategy distance is in general relatively small and the number of paths between strategies is relatively high (Theorems 6.3 and 6.4). Therefore, the connection cost of low local minima should not be very high, i.e., the shape should be of type B2 or B1. This is further strengthened when using some ranked join methods. Moreover, as the variance among relation characteristics in the catalog increases, the strategy cost distribution is shifted to the left,

and the shape should move closer to B1.

The overall conclusion from the above observations is that the shape is of type A2-B2 or A2-B1.

9.2. Space of Deep and Bushy Strategies

The characteristics of the *A* space are as follows. All strategies in the space have exactly the same degree, which is relatively high (Theorem 6.5), so the local minimum distribution is shifted to the left. Also, the cost difference between neighbors is relatively low. Hence, from our analysis in Section 4, and for most strategy cost distributions, the shape should be of type A1. The major distinction between *A* and *L* is that there is no degree difference among the strategies, so *A* is unlikely to include local minima of high cost. This, together with the low neighbor cost difference, results in the different type of shape.

With respect to the connection cost, *A* has similar characteristics with *L* (Theorems 6.3 and 6.6), and therefore the same arguments apply to both. The primary difference is that the interstrategy distance in *A* is in general larger than in *L*. However, that difference is much smaller than the difference in the size of the strategy space, so one can conclude that the connection cost of low local minima cannot be very high in *A* either. Hence, the shape should be of type B2 or B1. The effect of using some ranked join methods and the specifics of the strategy cost distribution is the same in the two spaces as well.

The above observations lead to the overall conclusion that the shape is of type A1-B2 or A1-B1.

10. ALGORITHM BEHAVIOR

In this section, we discuss how the II, SA, and 2PO algorithms should be expected to perform based on the shape of the cost function of the strategy spaces *A* and *L*. These conclusions are supported by several experimental results, which can be found elsewhere [Kang91].

10.1. Space of Left-Deep Strategies

We have concluded that the shape of the *L* space is of type A2-B2 or A2-B1. This implies that II needs many local optimizations before it reaches one that is of low cost. In general, much effort is wasted because of local optimizations finishing at high local minima. Nevertheless, since the shape is of type A2 and not A3, after a reasonable number of local optimizations, some strategies of low cost will be visited, so the final output will be of good quality.

SA will waste much time among strategies in high cost in the beginning. As the probability for uphill moves decreases, it should eventually be able to escape the high cost local minima that it will encounter and converge into the well bottom. Due to the risk of getting trapped in one of the many high cost local minima, however, its performance should not be very stable.

The behavior of 2PO depends on the length of its first phase, i.e., on how many local optimizations will be done before the simulated annealing phase starts. If the first phase involves few local optimizations, it is likely that the well bottom will not be reached during that time. This implies that in the second phase, there is high chance that the algorithm will be trapped in a high cost local minimum, so that its final output is unsatisfactory. On the other hand, if the first phase involves many local optimizations, the well bottom will be reached during that time and will be adequately explored during the second phase. So, the final output should be better than that of II.

10.2. Space of Deep and Bushy Strategies

By the previous analysis, the shape of the cost function of the strategy space *A* is of type A1-B2 or A1-B1. This implies that II can reach strategies of low cost after few local optimizations, and the final output will be of good quality again. SA will have a similar behavior to that in the *L* space, waste much time among high cost strategies, but eventually reach the well bottom. Its performance should be much more stable, however, since there are not many high cost local minima in which the algorithm can be trapped. Finally, 2PO should be able to reach the well bottom after few local optimizations and spend the rest of the time exploring that area. Taking advantage of the good qualities of the other two algorithms, it should be very stable and superior to them.

11. CHOICE OF STRATEGY SPACE

Given a specific strategy space (*A* or *L*), Section 10 identifies the algorithm that should be used for query optimization over that space. The main conclusion is that for *L*, the situation is not so clear-cut, with 2PO possibly being slightly better than II, especially when many local optimizations are attempted in the first phase. For *A*, on the other hand, 2PO is superior to the other algorithms across the board. In this section, we want to address the next higher level up issue, i.e., given the choice, which of the two spaces should be used for optimization. The following two subsections identify some inherent properties of the two spaces that are helpful in drawing some conclusions in the third subsection.

11.1. Structure of the Generated Strategy Space

In both *A* and *L*, the neighbors of any strategy *S* are determined by a set of transformation rules that can be applied on *S*. However, among the strategies thus constructed, only those that are cross-product-free are real neighbors of *S*. To move from some strategy to one of its neighbors, all algorithms described in this paper blindly apply one of the appropriate transformation rules and then check whether it includes a cross-product or not to determine its validity as a neighbor. The set of strategies that can be generated by applying all transformation rules on a given strategy *S*, independent of whether they contain a cross product or not, is the set of *generated neighbors* of *S*, and their number is the *generated degree* of *S*. Clearly, the degree of a strategy is less than or equal to its generated degree. The following two theorems shed some light on the generated degrees of strategies in the two spaces *L* and

A. For consistency with Section 6.1, we again concentrate on the Swap transformation rule in L; including the 3-cycle rule is straightforward.

Proposition 11.1: In L, every strategy is of generated degree $J(J+1)/2$.

Proposition 11.2: In A, every strategy is of generated degree $3J-2$.

If we compare the above results with Theorems 6.1, 6.2, and 6.5, a clear advantage of A over L emerges. For most queries, a large fraction of the generated strategies in L is not part of the actual space. This becomes worse as we move from a star query to a string query. On the other hand, for all queries, 2/3s of the generated strategies in A belong to the space. The corresponding ratio in L is higher for star queries, but lower for most other queries. Moreover, it is independent of the number of joins in A, whereas in general it grows with the number of joins in L.

Clearly, the lower the ratio of the degree over the generated degree of a strategy is, the more time will be spent in trying to find a legal neighbor, and the slower any algorithm will be. Hence, one can conclude that, in general, the same amount of useful work requires more time when searching L than when searching A.

11.2. Incremental Cost Computation

When a move is attempted from a strategy S_1 to one of its neighbors S_2, the cost of the latter need not be recomputed from scratch. It is enough to subtract the cost of all operations in S_1 that were modified and add the cost of the operations that replaced them in S_2. The following two results shed some light on the cost computation of neighbors in the two spaces L and A.

Proposition 11.3: Consider a strategy in the space L for a query with J joins. The number of operations whose cost needs to be recomputed when transforming the strategy into one of its neighbors is bounded by J.

Proposition 11.4: Consider a strategy in the space A for a query with J joins. The number of operations whose cost needs to be recomputed when transforming the strategy into one of its neighbors is bounded by 2.

If merge-scan is one of the available join methods, due to the side-effects of modifying the interesting order of a temporary result [Seli79], there can be transformations of a strategy in A that require a recomputation of the costs of up to J operations as well. Nevertheless, cost recomputation is in general much more time-consuming in L than A (linear in the number of joins vs. constant). Hence, the same conclusion with the previous section can be drawn, that the same amount of useful work requires more time when searching L than when searching A.

11.3. Comparison Between A and L

Space A has many more strategies than space L. Nevertheless, contrary to conventional wisdom, the former is easier to optimize than the latter. The shape of the cost function in A has a much more definite shape of a well than in L: type A1 vs. A2. Hence, algorithms like 2PO can take advantage of that and have a robust performance in optimization. Also, the results in Sections 11.1 and 11.2 show that searching itself is more efficient in A than in L. Finally, in most cases, the optimum strategy in L is not as efficient as the optimum strategy in A. The combination of easier and more efficient optimization with the potential of a better quality output makes A the strategy space of choice. We believe that it should be the preferred one when optimizing large join queries.

12. COMPARISON TO PREVIOUS WORK

Query optimization has been a very active area of research for relational database systems [Jark84, Kim86]. Regarding large join queries, their optimization was specifically addressed by Krishnamurthy et al. [Kris86], who proposed a quadratic algorithm that took advantage of the ASI property in the cost function when ranked join methods were used alone. In this section, we want to primarily discuss some earlier results of ours [Ioan90] and some results of Swami and Gupta [Swam88] in light of the findings in this paper with respect to the shape of the cost function of the strategy space.

Our previous work was with the A space whose shape, based on our previous analysis, was A1-B2. Our observations were consistent with what would be expected in that case: 2PO was superior to SA, which was superior to II. Also 2PO was the most efficient.

The work of Swami and Gupta was with the L space and with a relation catalog with very high variance in its contents. From the previous analysis, one can conclude that the study of Swami and Gupta dealt with a space whose shape was A1-B1 (Sections 9.1 and 10.1), as speculated in their paper as well [Swam88].

Swami and Gupta experimented with II and SA alone. By the discussion in Section 9.1 and the results in Section 10.1, one would expect that the two algorithms should provide output of similar quality. Moreover, II should reach low cost strategies relatively early and then fail to make any substantial improvement in the found cost. The observations of Swami and Gupta were slightly different [Swam88]: (a) II had a rather gradual improvement over time; (b) the output quality of SA was worse than that of II. We believe that these differences are due to two specific aspects of their modeling of the algorithms that do not conform to the model presented in Section 2. First, local minima were recognized by looking at J randomly generated (with replacement) neighbors of a strategy. Quite often, J represents a small fraction of the actual number of neighbors, especially for large queries. Hence, many strategies could be erroneously identified as local minima. This explains (a), since in the implementation of Swami and Gupta, local optimizations were prematurely ended much more often than in the presented model. Second, a time limit was used as one of the stopping criteria for SA. Because of the ability of the algorithm to make uphill moves and the high expense of each move (Proposition 11.1 and Proposition 11.3), SA needs ample

time before the probability for accepting uphill moves decreases enough for the algorithm to be positioned in the well bottom. This explains (b), since in the implementation of Swami and Gupta, it was very likely that the time limit was reached while that probability was still high, and therefore the best strategy visited at that time had high cost as well.

13. CONCLUSIONS

We have presented a combination of analytical and experimental results that shed some light into the shape of the cost function of the strategy spaces that query optimizers must deal with. We have argued that both such spaces essentially form a well, but of a distinctly different quality. We have shown how II, SA, and 2PO would perform on these spaces. The results have led to the conclusion that query optimization in the space of both deep and bushy trees is easier that in the space of left-deep trees alone. Since the former contains many more strategies as alternatives, it is expected that it would produce output of better quality as well, and therefore it should be the one used for query optimization.

We are currently working on extending the above results in two directions. First, we are looking into the optimization of other types of queries of the relational and other data models. Second, we are looking into other domains where optimization problems arise and test the validity of our generic results on the shapes of cost functions. The goal of both efforts is to examine whether a well is formed or not in the corresponding state spaces and thus test the applicability of algorithms like 2PO in those cases.

14. REFERENCES

[Gran81]
Grant, J. and J. Minker, "Optimization in Deductive and Conventional Relational Database Systems", in *Advances in Data Base Theory* **Vol. 1**, edited by H. Gallaire, J. Minker and J. M. Nicolas, Plenum Press, New York, N.Y., 1981, pp. 195-234.

[Ibar84]
Ibaraki, T. and T. Kameda, "On the Optimal Nesting Order for Computing N-Relational Joins", *ACM TODS* **9**, 3 (September 1984), pp. 482-502.

[Ioan87]
Ioannidis, Y. E. and E. Wong, "Query Optimization by Simulated Annealing", in *Proc. of the 1987 ACM-SIGMOD Conference on the Management of Data*, San Francisco, CA, May 1987, pp. 9-22.

[Ioan90]
Ioannidis, Y. E. and Y. C. Kang, "Randomized Algorithms for Optimizing Large Join Queries", in *Proc. of the 1990 ACM-SIGMOD Conference on the Management of Data*, Atlantic City, NJ, May 1990, pp. 312-321.

[Jark84]
Jarke, M. and J. Koch, "Query Optimization in Database Systems", *ACM Computing Surveys* **16**, 2 (June 1984), pp. 111-152.

[Kang91]
Kang, Y., *"Randomized Algorithms for Query Optimization"*, Ph.D. Thesis, University of Wisconsin, Madison, WI, April 1991.

[Kim86]
Kim, W., D. Reiner, and D. Batory, *Query Processing in Database Systems*, Springer Verlag, New York, NY, 1986.

[Kirk83]
Kirkpatrick, S., C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by Simulated Annealing", *Science* **220**, 4598 (May 1983), pp. 671-680.

[Kris86]
Krishnamurthy, R., H. Boral, and C. Zaniolo, "Optimization of Nonrecursive Queries", in *Proc. 12th International VLDB Conference*, Kyoto, Japan, August 1986, pp. 128-137.

[Monm79]
Monma, C. L. and J. B. Sidney, "Sequencing with Series-Parallel Precedence Constraints", *Mathematics of Operations Research* **4**, 3 (August 1979), pp. 212-224.

[Naha86]
Nahar, S., S. Sahni, and E. Shragowitz, "Simulated Annealing and Combinatorial Optimization", in *Proc. of the 23rd Design Automation Conference*, 1986, pp. 293-299.

[Roth86]
Rothschild, V. and N. Logothetis, *Probability Distributions*, John Wiley & Sons, New York, N.Y., 1986.

[Seli79]
Selinger, P. et al., "Access Path Selection in a Relational Data Base System", in *Proc. of the 1979 ACM-SIGMOD Conference on the Management of Data*, Boston, MA, June 1979, pp. 23-34.

[Sell86]
Sellis, T. K., "Global Query Optimization", in *Proc. of the 1986 ACM-SIGMOD Conference on the Management of Data*, Washington, DC, May 1986, pp. 191-205.

[Swam88]
Swami, A. and A. Gupta, "Optimization of Large Join Queries", in *Proc. of the 1988 ACM-SIGMOD Conference on the Management of Data*, Chicago, IL, June 1988, pp. 8-17.

[Ullm82]
Ullman, J. D., *Principles of Database Systems*, Computer Science Press, Rockville, MD, 1982.