Minimizing Response Times In Real Time Planning And Search

Shashi Shekhar & Soumitra Dutta

Computer Science Division
University of California,
Berkeley, CA 94720

Abstract

Real time artificial intelligence (AI) systems are required to respond within a given deadline, or have optimal response times. While some researchers have addressed the issue of planning under deadline constraints, there has been very little research towards optimizing the response time of problem-solving methods. The costs for a response consists of the cost to plan for a solution and the cost of executing the chosen solution. There is an intimate trade-off between these two costs. This paper presents an algorithm for providing near optimal response times by formalizing the trade-offs between planning and execution costs. We provide a proof of correctness and describe an implementation of the algorithm in a real time application of query planning. We also provide a model for considering response times in the context of the A* heuristic search algorithm.

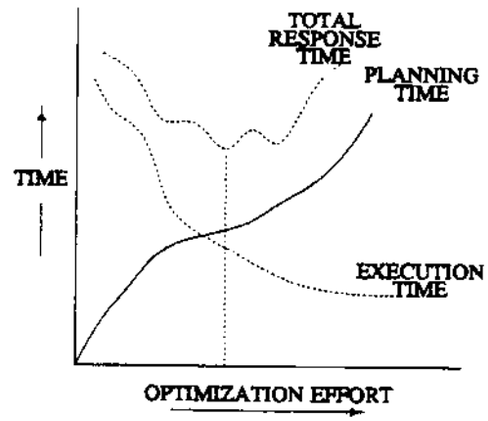

Figure 1: Planning, Execution and Response Times

1. Introduction

As the application of AI systems evolves from an art to an engineering science, we can expect more challenging applications to be addressed. Some of the most challenging and interesting applications can be found in real-time domains. An AI system operating in a real-time situation will typically need to respond within a certain deadline, or have optimal response times (for planning and execution). For example in managing defensive weapons against nuclear missiles, the system has to respond within a few seconds or minutes[l]. On the other hand, in credit approval systems[2], optimal response times are desired, but there are no hard deadlines. The deadline problem is hard and many such problems (e.g., scheduling with deadlines) are NP-complete [3] and guaranteeing an optimal solution within a fixed deadline is often not possible. The optimal response time is required in the case wherein the plan execution time is comparable to planning time. The response time is the sum of planning time and execution time, and the application has no strict short deadlines. As shown in figure 1, in general, the optimal response time is not achieved by solely minimizing execution time, as then planning time increases to offset the gains.

Currently, ad hoc techniques are used for making a system produce a real-time response , and these methods suffer from poor extensibility, brittleness and a lack of a formal proof of "reliable real time performance" [4). Traditional search algorithms like depth first search, breadth first search and depth first iterative deepening [5], A* [6] and IDA* [5] are useful with small search spaces and large deadlines only. However, the size and complexity of most search spaces faced by real time AI systems preclude the use of these algorithms, as they may take exponential time in producing a solution. These search strategies may be able to guarantee an optimal solution in the absence of limits on search time, but cannot function with the constraint of deadlines and response times. Consequently the planning methods based on A* and EDA* may potentially take exponential time for problem-solving. Within a given deadline to solve certain problem, the system may not produce any solution (complete or partial). Scaling them up with real world data and adequate knowledge-bases, would amplify their performance problems, including their inability to meet real time constraints[7].

Research on real-time AI systems have concentrated on software and implementation issues like interrupt handling[7]. There has been little research on problem-solving methods, which can meet a deadline or provide an optimal response time. Some researchers [8,9] have worked towards planning and search algorithms to meet reasonable deadlines by finding a partial solution within the given deadline. The algorithm is based on bounded look-ahead search, i.e. one searches forward from the current state to a fixed depth horizon determined by the deadline. A cost function (similar to A*) is used to evaluate the frontier nodes. The minimum value is then backed up and a single move is made in the direction of the minimum value. Korf has further proposed a real time modification of A* called RTA* for controlling the sequence of moves actually executed. Russell[10] has extended RTA* by adding meta-greedy decision-theoretic search control. Some of the results from *reasoning under resource constraints*[\ 1,12] are also applicable, if time is considered as a resource. This line of work is based on assigning utility numbers with choices and maximizing utility.

The above mentioned real time algorithms essentially deal with the problem of deadlines in real time systems (see classification in figure 1). They ignore the execution cost of any solution and essentially spend all available time (the deadline) searching. This may not be realistic in cases where executing the solution takes some appreciable amount of time and it is no longer possible to ignore it. Our research concerns the other class of real-time applications where response time is important.

To optimize the response time, a real time system has to consider the tradeoff between execution and planning costs. A greater planning cost may possibly lead to a lower execution cost, but the extra time spent planning may also lead to a potential increase in the response time as shown in Fig.t. Thus it is crucial that a real time system be able to decide upon the appropriate moment to stop planning and start executing the best solution plan obtained till then. Our focus in this paper is to formalize the trade-off between planning cost and execution cost in such real time systems and prove bounds on the optimality of the results obtained. We present a search algorithm for providing near optimal response times. We provide correctness proofs and bounds on the optimality of the response time performance obtained. We describe an implementation of our algorithm in a real-time application of query planning [13], which substantiates our results experimentally. Finally we explore the combination of our algorithm with A*.

The structure of this paper is as follows. Section 2 describes our search algorithm, correctness proofs and other results. Section 3 and 4 describes the application of our proposed algorithm in two example domains: query optimization (planning) and the A* heuristic search algorithm[14]. Finally, section 5 summarizes our contributions and lists directions for future research.

2. Optimizing Response Time

For a real time system with optimal response time requirement, the total time for responding to a situation can be conceptually modeled as consisting of the time for planning a path to solution, and the time for actually executing the chosen solution path. In some situations we need to generalize the planning and execution times to planning and execution costs or utilities [10,11] to account for various other aspects of the problem domain.

For simplicity, we will adopt to the model given by Korf, and consider costs as equivalent to the times required for planning and execution. In this context, the process of planning a response can be modeled as a search among different possible responses or *actions* for the best possible action. The space of different possible actions can be very large, and as the sequence of search for finding the best possible action is (usually) not known *apriori,* the process of planning (to find the best action) can take prohibitively large amounts of time, so as to thwart the desired optimality of the response time of the real time system. Thus it is important to devise algorithms which not only allow a real time system to suitably terminate the planning phase, but also prove some bounds on how well it has done as compared to the optimal response time.

2.1. NORA (Near Optimal Response-time Algorithm)

In this sub-section, we shall describe NORA, a simple and intuitively appealing algorithm for providing near optimal response times in real time systems that satisfies the above-mentioned concerns. Blind exhaustive searches like BFS or DFS are generally applicable to limited domains. Most heuristic search algorithms require the ability to recognize the goal node (i.e., the best action). This is not always possible, (e.g., recognizing the cheapest query execution plan in query optimization! 15]) and one needs other termination criteria to stop the search.

Given a search algorithm, a real time system has to make a crucial decision of determining the time to stop searching so as to provide a near optimal response time. This problem is simplified for small search spaces (where it may be possible to search the entire search space quickly) or when it is possible to recognize the best action (so that the search can be stopped as soon as the best action is found). In general it may not be possible to characterize apriori the best action and even if it is possible to characterize it for recognition, reaching it in the many search traversal algorithms may require prohibitively large time. Thus a real time system must have some algorithm for appropriately balancing the planning and execution costs.

The basic idea of NORA is simple and intuitive. Conceptually, the search space can be thought of as a set of nodes with interconnecting arcs. Each node represents one possible action (or set of actions) and has an associated execution cost. A search traversal algorithm specifies the order of traversal of these nodes and there are costs associated with this traversal. These costs include the cost of actually moving from node to node and any associated computation costs at each node (for example, to estimate the execution cost of the actions at the node). The chosen search traversal algorithm goes from node to node in some order to find the node with the best action (or here for simplicity, the least *action time).* At each node, NORA keeps track of two metrics:

[1] Planning cost so far this represents all associated costs of planning so far and includes all cost of traversing the search space.

[2] Least execution cost so far: this is the node with the least execution cost so far.

The search space traversal is terminated whenever the planning cost exceeds some fraction of the best execution cost found so far, i.e., when the following condition is satisfied:

$$\textit{planning cost so far} \geq \frac{\textit{least execution cost so far}}{\lambda}$$

where *X* defines the fraction. As shown in the next sub-section, this simple stopping criterion allows us to achieve a near-optimal response time (when the search is terminated with the above condition satisfied) satisfying the following bound:

$$\frac{RT(\textit{obtained when stopped})}{RT(\textit{optimal})} \leq max(1+\lambda,\ 1+\frac{1}{\lambda})$$

This is for the general case, in which we are unable to recognize *apriori* the best action. In cases, where it is possible to recognize the best action, a direct comparison can be made for a crisper bound (see section 3.2). As evident, the choice of *X* is crucial and at the end of the next sub-section, we describe how we can obtain heuristics for the choice of an appropriate *X.* We further note that the bound holds for any given search space traversal ordering (e.g. A*).

22. Correctness of NORA

In this section, we provide a correctness proof for NORA. Some notation used in the proof below is introduced in *Table 2.1.* We structure the search space as a interconnected set of nodes Q_i,, each node representing a possible action (or set of actions).

Symbol	Meaning
$SP(i)$	space of nodes already explored up to and including Q_i
$C_E(j)$	cost of best execution plan for node Q_j
$\tau(i)$	total planning cost up to node Q_i, i.e. $\sum_{Q_j \in SP(i)} C_E(j)$
$t(i)$	cost of best plan so far, i.e $\min_{Q_j \in SP(i)} C_E(j)$
$RT(i)$	response time if search terminates at Q_i i.e. $\tau(i) + t(i)$
$RT(opt)$	theoretically minimum response time possible for the given search algorithm $RT(opt) = \min_{0 \le i \le n} RT(i)$
$\tau(opt), t(opt)$	components of $RT(opt)$
Q_{opt}	node corresponding to $RT(opt)$

Table 2.1 Notation.

Correctness Theorem for NORA: If the following search te minating criterion is used,

$$\tau(i) \ge \frac{t(i)}{\lambda},$$

the following upper bound on $RT(i)$ is obtained,

$$\frac{RT(i)}{RT(opt)} \le max(1+\lambda, 1+\frac{1}{\lambda})$$

Proof: Supposing the search terminates after examining Q There are following two cases,

[a] $Q_{opt} \in SP(i)$ i.e. Q_{opt} has been visited.

[b] $Q_{opt} \notin SP(i)$ i.e. Q_{opt} hasn't been visited.

Case [a]:

$$\frac{RT(i)}{RT(opt)} = \frac{\tau(i)+t(i)}{\tau(opt)+t(opt)} = \frac{\frac{t(i)}{\lambda}+t(i)+\delta}{\tau(opt)+t(opt)} \quad \text{since } \tau(i) = \frac{t(i)}{\lambda}+\delta$$

A positive quantity δ is added to the numerator to make the equali hold. It satisfies the condition $0 \le \delta \le$ *Opt cost in current step*.

$$\Rightarrow \frac{RT(i)}{RT(opt)} \le \frac{\frac{t(i)}{\lambda}+t(i)+\delta}{t(opt)} \quad \text{since } \tau(opt) \ge 0$$

$$\le \frac{(\frac{1}{\lambda}+1)t(opt)+\delta}{t(opt)}$$

because $Q_{opt} \in SP(i) \Rightarrow t(i) \le t(opt)$

$$\simeq 1+\frac{1}{\lambda} \quad \text{Opt cost in current step} \ll t(opt) \Rightarrow \frac{\delta}{t(opt)} \simeq 0 \ddagger$$

Case [b]:

$$\frac{RT(i)}{RT(opt)} = \frac{\tau(i)+t(i)}{\tau(opt)+t(opt)} \le \frac{\tau(i)+\lambda\tau(i)}{\tau(opt)+t(opt)} \quad \tau(i) \ge \frac{t(i)}{\lambda}$$

$$\le \frac{\tau(i)+\lambda\tau(i)}{\tau(opt)} \quad t(opt) \ge 0$$

$$\le \frac{(1+\lambda)\tau(i)}{\tau(i)} \quad Q_{opt} \notin SP(i) \Rightarrow \tau(opt) \ge \tau(i)$$

$$= 1+\lambda$$

Combining the results of the cases above we have,

$$\frac{RT(i)}{RT(opt)} \le max(1+\lambda, 1+\frac{1}{\lambda})$$

The bound provided by the above theorem is not very tight, and the termination criterion performs much better in practice. This was observed when we ran an experiment, details of which are discussed elsewhere[16].

Corollary 1: If the planning is the *compile–and–store* type, and is expected to be executed α times, the above bound changes to,

$$\frac{RT(i)}{RT(opt)} \le max(1+\alpha\lambda, 1+\frac{1}{\alpha\lambda}) \dagger$$

Proof: Identical to the above theorem.

Heuristics For Choice Of λ

The choice of λ depends on the size of the search space, and the time available for planning. For a small search space, we would usually expect *Case [a]* to occur when the search terminates, i.e. $Q_{opt} \in SP(i)$. Thus,

Small Space => Case [a]
=> choose large λ for tight bound
=> less planning is good for response time

For a large search space, we would usually expect *Case [b]* to occur when the search terminates, i.e. $Q_{opt} \notin SP(i)$. Thus,

Large Space => Case [b]
=> choose small λ for tight bound
=> more planning is good for response time

Rule of Thumb for choosing λ: Above analysis shows that the thumb-rule is to choose a large λ if the search space is small and *vice versa*. It is not a certain rule because of the approximate implications (=>) shown above.

3. Query Optimization: A Real-time Application

We implemented our algorithm for optimizing the response time of query optimizer for data-bases. Most databases are on-line and users access desired information via a *query,* represented in a suitable query language. Queries involve operations like join, select, and project, and can be answered by executing several different execution-plans each with different execution-times. The query optimizer generates and examines many execution plans to choose the one with the least execution-time. The planning time increases exponentially, as the optimizer expands its search of possible execution plans.

The optimization cost can be comparable to the cost of query execution. The tradeoff between optimization time and the cost of query execution becomes a major issue in optimizing the total cost of query processing. We further note that it is not possible to *apriori* specify the query with the least execution cost, while traversing the search space.

We tested NORA for semantic query optimization for a shipping database of six relations. The database schema, the relation sizes, semantic integrity constraints and the various indexes available are reported elsewhere[16,17]. The first step in our experiment was to generate the search space. The NORA algorithm was simulated on the search space. The stopping rule of NORA terminates the search when the following condition becomes true : T(/) > *t(i)* The stopping rule was examined for the values λ = 2, 1, and 1/2. The results are presented in *Table 3.1.*

X This assumption is justified by many planners, e.g. query optimizers,
t *RT(i)* = T(i) + a/ (i), and represents an integrated cost.

Iteration	τ(i)	t(i)	τ(i)+t(i)	Stopping(λ)	RT(i)/RT(opt)
1	10	605.7	615.7		
2	97	127.8	224.8	λ=2	1
3	188	127.0	315.0	λ=1	1.5
4	359	127.0	486.0	λ=1/2	2.5
5	647	127.0	774.0		

Table 3.1 *Performance of NORA*

The result can be verified for all the three values of *X.* Here we shall illustrate the validation for *λ-* 1. The search stops after iteration 3, since the stopping criterion of NORA evaluates to (163X1) + (5)(5) > 127.0 =» 188 > 127.0, which is true. The optimization cost and best execution cost estimate are,

t(i)=188w; t(i)= 127.0 *ms*

The bound of the *Correctnes Theorem* is satisfied since,

$$)\frac{\tau(i)+t(i)}{\tau(opt)+t(opt)} = \frac{315.0}{224.8} \le 2 = \max(1+1, 1+\frac{1}{1})$$

We see that stopping rule is quite effective. Especially notable is the fact that even though we attempt to minimize a weighted sum of T(I) and r(i), and not *t(i)* alone, the value of /(i) obtained is actually quite close to the minimum. We believe a more careful analysis of the algorithm would help us get a better bound.

4. NORA and Heuristic Search Algorithm A*

A* +[18] is probably the best known heuristic search algorithm for finding a solution path from an initial *start* node to a *goal* node, when the execution costs are negligible. It finds an optimal solution path in optimal planning time if the heuristic function, h(n), is monotone and admissible, i.e., it satisfies triangle inequality and it never overestimates the actual cost of reaching a goal node. A* ignores execution time and essentially concentrates on minimizing the searching time instead on response time. Furthermore A* terminates by recognizing the goal node. Thus the problem definition must provide adequate characterization of the goal node, so that it can be recognized easily.

To apply NORA formalism to A* we generalize A* in two respects: (a) introduce the notion of execution time, (b) relax the assumption that the search algorithm can recognize the *goal* node during the search process, even though one can define heuristic functions. We use a more general stopping criteria for search termination.

To introduce execution costs in A*, we proceed as follows. Associated with any node n, in the search space, let $g_e(n)$ represent the execution time for executing the partial solution (corresponding to the path from the start node to node n) found so far and let *h, (n)* represent the heuristic estimate of the cost of execution from the partial solution to a goal node (corresponding to a path from node n to a goal node). Note that $g_e(n)$ and $h_e(n)$ are distinct from the conventional g and h functions of A*. While g(n) estimates the cost of expanding the search tree from the start node to the node n and thus contributes to the planning costs for finding a solution path from the initial state to the partial solution state specified by node n, $g_{e\ell}(n)$ estimates the cost of *actually executing* that partial solution path in the real world. Similarly, while h(n) estimates the planning cost of further expanding the search tree from node n to a goal node, $h_e(n)$ estimates the actual cost of execution while trying to reach a goal from the partial solution state specified by node n. These functions can be computed as it is possible to *apriori* precisely characterize the goal state.

+ a best-first search of the search space, where the merit of a node, f(n) is the si of reaching that node, n, from the start node, S, and the estimated cost, h(n), of reac that node.

While trying to optimize response times in real time planning and search using A*, it is possible that we may decide to terminate the search (based on a chosen search stopping criterion, e.g., as specified by NORA) at some node i, before reaching a goal node, G. Assume that stopping at some node i, (i * goal node), forces us to incur some penalty for execution in real time (caused by the distance from the partial solution state, node i, to a goal node), and let it be represented by h_t(i). We assume that h_e(i) is proportional to the distance of the node i from a goal node and that it monotonically decreases as i approaches a goal node.

4.1. Applying Stopping Criterion of NORA

Let us assume that the A* heuristic evaluation function, h, satisfies the monotone t restriction and is admissible. Under these conditions, A* never expands any nodes other than those that lie on the optimal solution path and thus at any node, i, (i = goal node), on the optimal solution path, the total planning cost so far is given by g(i) and the best estimate of the execution cost is given by the sum of $g_{\epsilon}(i)$ and $h_e(i)$. Let G represent the goal node, and thus the planning cost for reaching the goal node is *g(G)* and the execution cost is given by the sum of g_e(G). The stopping criterion as specified by NORA would be: stop when

$$g(i) = \frac{g_e(i)+h_e(i)}{\lambda}$$

Writing the ratio of the response time, RT(i), when stopping at node i over the response time, RT(G), when stopping at goal node, G, we have:

$$\frac{RT(i)}{RT(G)} = \frac{g(i)+g_e(i)+h_e(i)}{g(G)+g_e(G)} \le \frac{g(i)+g_e(i)+h_e(i)}{g(i)+h(i)+g_e(G)}$$

Note that *g*(G) £*g(i)* + *h(i)* as h is assumed to be admissible. Since h(i) is positive, we can ignore it from the denominator and on substituting the stopping condition we get:

$$\le \frac{(1+\lambda)\,g(i)}{g(i)+g_e(G)}$$

Dividing both numerator and denominator by g(i), we get

$$\le \frac{1+\lambda}{1+\frac{g_e(G)}{g(i)}} \le (1+\lambda)$$

as both *g,(G)* and g(i) are positive. Thus the bound of NORA is satisfied. Note that as λ is positive, 1 + λ is greater than 1.

Thus result provides an interesting extension of A* for minimizing response time for the real-time problems, where plan-execution time is not negligible. Use of stopping criteria of NORA leads to a near optimal response time, which is shown to be near-optimal even when h_e(goal) = 0. We have verified this assertion for the best case of A*, ie the case of monotone admissible heuristic functions. It can easily be verified that the assertion holds in all cases.

5. Conclusions

We have looked at real-time AI applications, where planning and execution costs are equally important. We have presented a classification of real-time problem solving systems, and an algorithm for producing near optimal response times. We proved that balancing planning cost with execution cost, can lead to near-optimal response time, and provided bounds on the worst-case deviation from the optimal point. We have then provided an empirical validation of the algorithm, via a real-time application of query planning for databases. Experimental results show that the performance of our algorithm is often much better than the worst case bounds. We have also developed a framework for obtaining bounds on response times in the context of real time heuristic search using A*.

The results can be extended in many directions. The bounds on the performance of the algorithm are far from tight as illustrated by experiments. We would like to prove tighter bounds on the optimality of our algorithm. Secondly, our termination criterion can be used with any search algorithm like A*, RTA* etc. More research needs to be done in this direction. In particular we would like to combine it with RTA*[9] and explore the properties of that algorithm. Finally, we are exploring the applicability of our results in domains.

6. Acknowledgements

We would like to thank Prof. Srivastava, Univ. of Minnesota, for help with the implementation and performance evaluation of the semantic query optimization example, and Prof. Korf and Dr. Horvitz for fruitful discussions on real time search.

7. References

References

1. RP. Bonaso, "What AI Can Do for Battle Management: A Report of the AAAI Workshop on AI Applications to Battle Management," *AI Magazine,* vol. 9, no. 3, p. AAAI, Fall 1988.
2. H.P.Newquist III, "American Express and AI: Don't Leave Home Without Them," *AI Expert,* vol. 2, no. 4, p. Miller Freeman Publications, 500 Howard St., San Francisco, CA 94105., April 1988.
3. M. R. Garey and D. S. Johnson, *Computers and Interactability,* W.H.Freeman and Company, New York, 1979.
4. CA. O"Reilly and A.S. Cromarty, "Fast is not Real-Time in Designing Effective Real-Time AI Systems," *Applications of Artificial Intelligence II,* p. International Society of Optical Engineers, Bellingham, Washington, 1985.
5. Richard E. Korf, "Depth-First Iterative Deepening : An Optimal Admissible Tree Search," *Artificial Intelligence,* vol. 27, pp. 97-109, North-Holland, 1985.
6. D. Galperin, "On the optimality *of A*," Artificial Intelligence,* vol. 8, no. 1, pp. 69-76,1977.
7. T.J. Laffey and P.A.Cox, "Real-Time Knowledge Based Systems," *AI Magazine* , vol. 9 , no. 1, p. AAAI, Spring 1988.
8. R.E.Korf, "Real-Time Heuristic Search: New Results," *Proc. AAAI Conference,* 1988.
9. R.E.Korf, "Real-Time Heuristic Search: First Results/' *Proc. AAA1 Conference,* 1987.
10. S.Russell and E. Wefald, "Decision Theoretic Control of Reasoning: General Theory and an Algorithm to Game Playing," *Report No. UCB/CSD 88/435,* p. Computer Science Division, U.C.Berkeley, 1988.
11. E. J. Horvitz, "Problem Solving Design: Reasoning about Computational Value, Tradeoffs and Resources," *Report No. KSL-87-64,* Knowledge Systems Laboratory, Stanford University, CA 94305, 1987.
12. E. J. Horvitz, "Reasoning Under Varying and Uncertain Resource Constraint," *Report No. KSL-88-35,* Medical Computer Science, Stanford University, Stanford, CA 94305, 1988.
13. Yao,S.B., "Optimization of query evaluation algorithms," *ACM TODS,* vol. 4, no. 2, 1979.
14. P.E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Systems Sci. Cybernet.,* vol. 4(2), pp. 100-107,1968.
15. King.J.J, "QUIST : A system for semantic query optimization in relational databases," *Proc. 7th VLDB Conf,* 1981.
16. Shekhar, S., Srivastava, J., and Dutta, S., "A Model of Trade-off between Optimization and Execution *costs* in Query Processing", *Int'l Conf. on Very Large Data- bases*. Los Angeles, CA. ,
17. J. Srivastava, "New Optimization Techniques in Database Access and Maintenance," *PhD. Dissertation,* p. Computer Science Division, U.C.Berkeley, 1988.
18. N. J. Nilsson, *Principles of Artificial Intelligence,* Tioga, Palo Alto, CA, 1980.

† if the heuristic function h, satisfies the *monotone restriction*[18] (i.e., the estimate of the optimal cost to a goal from node n_i not be more than the cost of the arc from node n_i to n_j plus the estimate of the optimal cost from n_j to a goal), then it can be proved that A* already has found an optimal path to any node that it selects for expansion, i.e., it never expands any nodes not on the optimal path from the start node to a goal node.