Efficiency of Randomized Parallel Backtrack Search

Yanjun Zhang and A. Ortynski

Abstract. This paper presents an improved analysis of a randomized parallel backtrack search algorithm (RPBS). Our analysis uses the single-node-donation model that each donation contains a single tree node. It is shown that with high probability the total number of messages generated by RPBS is \( O(phd) \) where \( p \) is the number of processors, and \( h \) and \( d \) are the height and degree of the backtrack search tree. Under the assumption of unit-time message delivery, it is shown that with high probability the execution time of RPBS is \( n/p + O(hd) \) where \( n \) is the number of nodes of the backtrack search tree and the leading term \( n/p \) has no constant factor. As the result of limited communication requirement, RPBS can be efficiently implemented in message-passing or shared-memory multiprocessor systems. A general analysis of network implementation of RPBS is presented. The concept of total routing time, the sum of routing times of all messages, is introduced as a measure of communication cost. It is shown that the overall effect of message delay to the execution time of RPBS is small if the total routing time is small. Some experimental data on a shared-memory machine are reported.

Key Words. Backtrack, Parallel computation, Randomized algorithm, Distributed computing, Combinatorial search problem.

1. Introduction. Backtrack search is a fundamental method for solving combinatorial search problems. A backtrack search has the property that it is possible to determine that some initial choices cannot lead to a solution. This property allows the search to terminate a sequence of choices that cannot lead to a solution and “backtrack” to a point where a new choice for the search can be made. On each problem instance, a backtrack search generates a backtrack search tree in a depth-first traversal. An early study of backtrack search programs [K] contains an excellent account of backtrack search. The computational complexity of backtrack search is investigated in [CSW]. This paper assumes that a backtrack search seeks all solutions to a problem instance; an all-solution backtrack search must generate the entire backtrack search tree.

A simple randomized parallel backtrack search algorithm (RPBS) for message-passing multiprocessor systems was analyzed in [KZ]. The algorithm uses a multinode-donation model in which a donation encodes a set of nodes that are consecutive children of a parent node. The analysis assumes the unit-time delivery of a donation message. It was shown that with high probability the execution time of RPBS using \( p \) processors is \( O(\log d(n/p + h)) \) on any instance \( H \) where \( d \) is the maximum number of children of a node in \( H \). This upper bound is within a factor \( O(\log d) \) from the inherent lower bound \( \max\{n/p, h\} \). The same analysis would render an upper bound of \( O(d(n/p + h)) \) on the
The execution time of RPBS if a donation of $k$ nodes are encoded by $k$ separate messages, each containing one node.

This paper presents an improved analysis of RPBS in the *single-node-donation* model in which each donation contains only one node. Our analysis bounds the total number of messages generated by RPBS. It is shown in Section 4 that with high probability the total number of messages generated by RPBS is $O(phd)$ where the constant factor is fairly small. As a consequence, under the assumption of unit-time message delivery, the execution time of RPBS is $n/p + O(hd)$ with high probability where the leading term $n/p$ has no constant factor. In practice, $n$ usually dominates $phd$ and the upper bound of $n/p + O(hd)$ approaches the lower bound $\max\{n/p, h\}$. Thus RPBS is highly efficient for executing backtrack search in parallel.

As RPBS generates a limited number of messages, backtrack search can be efficiently implemented in message-passing networks. The communication cost of RPBS will be insignificant to the overall computation, even when the messages are delivered one at a time in a single-bus architecture. The empirical results of a shared-memory RPBS in Section 7 support the same conclusion.

This paper introduces the concept of “total routing time,” which is the sum of routing times of all messages, as a measure of communication cost for message-passing models. A network-independent analysis in Section 5 shows that if the total routing time of RPBS is small, though the routing times of some individual messages may be large, the overall effect of message delays will be small. For the specific networks of ring, hypercube, and mesh, we show in Section 6 that the expected total routing time of RPBS is $O(M\Delta)$ where $M$ is the number of messages and $\Delta$ is the network diameter. These bounds indicate that RPBS can be efficiently executed in practical message-passing networks.

2. Related Work. Wu and Kung [WK] studied the tradeoff between the computation time and the communication cost in the context of parallel divide-and-conquer. Their model applies to parallel backtrack search. In their model the communication cost is taken as the total number of “cross-nodes,” the nodes generated by one processor but expanded by another processor. A shared data structure, called global pool, is used to gather the nodes that are closest to the root of input tree $H$; an idle processor finds new work by taking nodes from the global pool. It was shown that using the global pool, the number of cross-nodes is $O(phd)$. However, this bound does not include the communication cost for keeping the global pool up to date and the access conflicts to the global pool. The number of cross-nodes is the same as the number of donations in the single-node-donation model. We will show that in the single-node-donation model, the number of requests of RPBS, which is at least the number of donations, is $O(phd)$ with high probability.

Another paper by Wu [W] described a distributed implementation of the global pool for a class of networks that includes mesh and hypercube. It was stated that the overhead for an explicit load-balancing is $O(phd\Delta)$ and the computation time is $n/p + O(hd\Delta)$ where $\Delta$ is the diameter of the network. However, these results are network-specific and the overhead for load-balancing can be high in practice. Our bounds on the number of messages of RPBS are network-independent. We have similar time bounds for specific networks of ring, mesh, and hypercube (see Theorem 5(ii) and Section 6).