A Massively Parallel Heuristic Search Algorithm

Chang-Duk Jung*
You-Keun Park

Online access at www.publishingindia.com

Abstract

Heuristic search is an important technique in artificial intelligence. The best-first branch-and-bo ( ) algorithm is not only the most general search algorithm, but also one of a few general algorithms which can be used to solve a wide range of nondeterministic polynomial (NP)-hard processors. The algorithm can be parallelized by using a loosely-coupled network of processors. In this distributed algorithm, each processor maintains a local OPEN list and processes it implementation. the processors are required to exchange nodes of their local OPEN lists. One of the effective methods of exchanging information between processors is by using a BLACKBOARD. However, the BLACKBOARD requires shared memory. In this paper, we present a distributed algorithm using Binary Multi-Level Multi-Access (BMLMA) communication protocol. The communication network of BMLMA provides the global sorting of messages as a by-product of the protocol. Logically, this is analogous to a global associative memory: thus, it can be used to implement a logical associative BLACKBOARD. Computer simulation using a traveling salesman problem has confirmed the linear scalability of proposed algorithm.

Keywords : branch and bound, parallel search, multi-access protocol, linear scalability

1. Introduction

Search is the heart of many applications, such as document retrieval, operations research, and artificial intelligence. The search operation can naturally be implemented concurrently, since each operation is independent of other operations. The parallel search algorithm would be based on a simple divide-and-conquer strategy. The search space may be distributed uniformly to each processor of a parallel computer, and the search argument is broadcast to all processors. The processors search the argument and transmit the results back to the host processor.

For many problems, heuristic domain knowledge is available, which can be used t avoid searching unpromising parts of the search space. The branch-and-bound algorithm with under-estimate (A^*) tries to take advantage of the heuristic domain knowledge with dynamic programming technique. [3] The A^* algorithm is used to find a least-cost path between a start state and a set of goal states of a given state-space graph. The A^* algorithm maintains an OPEN list which is a set of nodes with estimated lower bound of the cost each node. The A^* algorithm is the most general search algorithm. For example, if the OPEN list is maintain in a first-in first-out list. the algorithm becomes a depth-first search, and if the list is maintained in a last-in first-out list. the algorithm becomes a depth-first search. Lastly, if

*PhD, Professor, Korea University, Korea

the node is selected based on the minimum lower bound (under estimate), the search becomes a best-first search.

However, the *A** algorithm cannot be effectively parallelized using above simple divide-and-conquer strategy because the processors of a parallel computer may end up doing a lot more searching than a serial processor would do. This would greatly reduce the speedup of the parallel computer. Thus, a parallel *A** algorithm requires control strategy which enables each processor to select the globally most promising node to process. A simple control strategy would be maintaining a global OPEN list in shared memory consisting of all nodes which have been generated but not yet examined in the search tree and to share it with all processors. One of the drawbacks of this approach is that the synchronization and memory contention delays of shared memory are do great that the speedup of parallel processing in degraded rapidly with the increase if the number of processors. [4] Distributed approached using a network of processors have been proposed to alleviate the memory contention problem. IN a distributed approach, each processor maintains its own local OPEN list. To ensure that each processor has its share of promising nodes, there must be a method with which the processors exchange the nodes of their local OPEN lists.

Wah [2] has proposed a distributed computer system using a unidirectional ring network. Rao [14] proposed a distributed approach using a shared BLACKBOARD. However, neither approach can be used effectively for a large number of processors because of the limitation of the ring network and the contention problem of the shared BLACKBOARD.

In this paper, we present a new approach using a logical associative BLACKBOARD.

In our system, there is no physical BLACKBOARD, but the communication network of a parallel computer called Binary Multi-Level Multi-Access (BMLMA) communication protocol provides a logical associative BLACKBOARD through which processors exchange the nodes of their local OPEN lists. The logical associative BLACKBOARD is based on the communication protocol BMLAM is a non-blocking contention-free multi-access protocol. It is based on ideas from associative memory organization and provides some of the features available using this structure such as ordered data retrieval and determination of extremal values. Thus, the BMLAM communication protocol is logically analogous to a global associative memory which can be used to implement a logical associative BLACKBOARD. Since there is no physical shared memory, the now distributed algorithm suffers no memory contention problem, i.e. the synchronization and queuing delays, that limit the maximum number of processors which can be used whithout degrading the speedup of the distributed algorithm for a large number of processors when the problem size is large computer simulation using the traveling salesman problem has confirmed the linear scalability of the architecture. In section2, parallel branch-and-bound algorithms are reviewed. Section 3 describes the BMLAM protocol. In Section 4, the distributed branch-and-bound algorithm using the logical associative BLACKBOARD is presented. In Section5, computer simulation results are presented. Conclusions are drawn in Section 6.

2. Parallel Branch-and-Bound Algorithms

The class of NP-hard problem covers a wide range of application in fields such as operation research and artificial intelligence [1] an important characteristic of NP-hard problems is that there is no known optimal algorithm to solve the problems with a computational time that increases polynomially with the size of the problem: computational time goes up at least as fast as k^n with $k>1$. Thus it is obvious that improving the computer speed alone is not enough to expand the solvable problem space of NP-hard problems. Likewise, parallel processing itself does not provide the solution, because to achieve an exponential reduction in processing time, an exponential number of processors must be used. Thus heuristic or approximate algorithms which reduce the problem size solvable in polynomial time must be utilized. Although various polynomial-time approximation algorithms with guaranteed bounds have been developed, these algorithms are problem-dependent. [15]The branch-and-bound algorithms is one of a few known general techniques for solving these NP-hard problems. The branch-and-bound algorithm decomposes the solution space onto smaller disjoint subspaces repeatedly until infeasibility is proved or a solution is found. [3] Although the complexity of the branch-and-bound algorithm is exponential in nature, it has been shown that an exponential reduction in iterations can be achieved with a linear reduction in accuracy [2]. Then, parallel processing can be used to linearly reduce the number of iterations. Thus, approximation and parallelism can be applied together to reduce the complexity of branch-and-bound algorithms[2]. Approximate branch-and-bound algorithms differ from the optimal branch-and-bound algorithms only by the termination rules. Suppose it was decided that a deviation of 10percent from the optimum was tolerable. If a feasible solution of 100 is obtained, all subproblems with lower bounds of 90 or more will be terminated. Techniques are also available to fine the best solution in a given length of time. [3]

There has been wide interest in parallelizing the branch-and-bound algorithm. The behavior of parallel branch-and-bound algorithms has been studied by several researchers [11]. The theoretical speedup obtained when k processors are used can be greater than k (acceleration anomalies), between one and k (decelerating anomalies), or less than one (detrimental anomalies). However, acceleration and detrimental anomalies are unlikely to occur in parallel best-first branch-and-bound (*A**) algorithms unless there are a large number of sub-problems having the same lower bound. [12]

For a parallel *A** algorithm, it is required to select a set of the most promising (the minimum lower bound) nodes equal in size to the number of processors in each iteration. There are two basic approaches in implementing a parallel *A** algorithm : the centralized approach and the distributed approach.

The centralized approach maintains a global OPEN list which is shared by all processors. At most one processor can access the OPEN list at any given moment, and the other confliction processors must be blocked. Thus, This memory contention problem combined with the communication overhead greatly reduces the gain in system speedup for a large number of processors. Simulation results have shown that the speedup

saturates very quickly with approximately 40 processors. [4]

In the distributed approach, each processor maintains a local OPEN list. However, the processors exchange the nodes of their local OPEN lists a communication medium between iteration. Wah [2] proposed a new multicomputer architecture called MANIP which maintains the OPEN list locally. The ring network employed in MANIP redistributes the local OPEN list among processors so that the n best sub-problems are assigned to each of n processors. simulations showed that for a small number of processors, the total number of iterations are relatively constant regardless of the number of processors. Rao [14] proposed an algorithm using a shared BLACKBOARD through which nodes are exchanged among processors. After selecting a node from its local OPEN lost, the processor proceeds with its expansion only if it is within a 'tolerable' limit of the best node in the BLACKBOARD. If the selected node is much better than the best node in the BLACKBOARD, then the processor transfers some of its good nodes to the BLACKBOARD before expanding the current node. If the selected node is much worse than the best node in the BLACKBOARD to itself, and then reselects a node for expansion. However. both of the above distributed approaches can be used only for a small number of processors. In MANIP, for a large number of processor, the number of shifts between iterations becomes too large of the number of total iterations increases rapidly with the number of processors: thus, the performance of the system degrades substantially for a large number of processors. The BLACKBOARD requires the critical section of shared memory. The synchronization and memory contention overheads increase rapidly with the number of processors [4].

The recent advances in VLSI technology enable the designer to build massively parallel computers which consists of hundreds or even thousands of processors. These massively parallel computers can effectively solve many NP-hard problems practically if we can develop a distributed A^* algorithm which provides linear performance improvement with the number of processors without saturating the speedup of the parallel computer. In this paper, we propose a distributed A^* algorithm using a logical associative BLACKBOARD which takes of advantage of the architecture of BMLMA protocol.

3. The BMLMA protocol

In this section, a general description of BMLAM protocol is presented. There are communication protocols which provide the sorting of messages as a by-product of the communication protocol [8]. Rothauser and wild[10] have proposed called MLMA (Multi-Level Multi-Access) to resolve contention in a common transmission medium. The

cycle	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bus	0	0	0	1	0	0	1	1	0	1	0	0	0	1	0	1
			*			*				*					*	
Contending Processors	1	1	1	2	1	1	1	1	3	3	3	3	3	3	3	3
	2	2	2		3	3			4	4	4	4				
			3	3			4	4								
			4	4												

* *The underlined processor indicates the processor which has successfully sent data.*

Figure 1 : An example of the BMLMA protocol

Original MLMA was designed to resolve the priority of processors in decentralize manner, but the algorithm can be applied to the data instead of the processor number. As suggested in [10], the algorithm has been modified for a parallel computer in which the propagation delay is extremely small. The algorithm is analogous to the parallel depth first search of the binary tree representation of messages.

BMLMA is one of multi-access communication protocols using a broadcasting common channel. The main issue in the broadcasting channel is the design of multi-access disciplines of the common channel. The multi-access protocols can be grouped into three different classes: fixed assignment techniques, random access (contention based) techniques, and demand assignment (contention free) techniques [5]. BMLMA is a priority-based attempt-and-defer demand assignment technique. In this scheme, every previous data transmission. However, if transmission from a higher priority processor is detected, processors with lower priority must suspend transmission and defer it to the next slot. The basic principle accordingly. In BMLMA, the processors are contending for the global channel with data at the bit level.

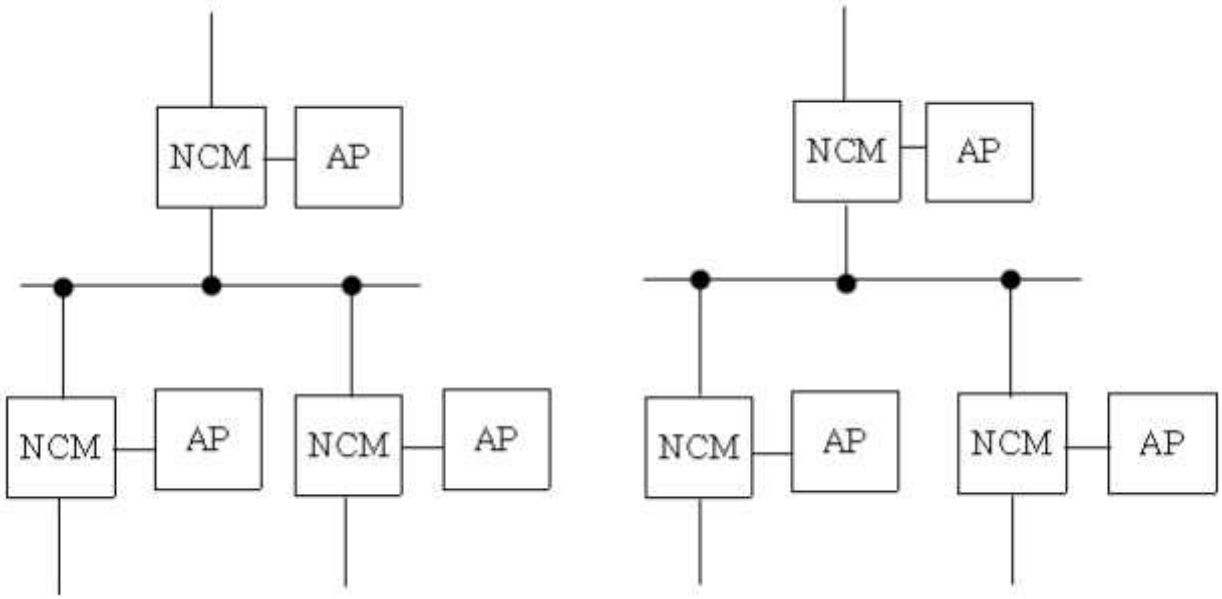
The basic operation of priority-based attempt-and-defer demand assignment multi-access protocol is as follows. If a processor wants to send a message, it transmits one bit at a time as soon as it senses that the channel is idle. To avoid conflicts, a processor always listens to the channel right after transmitting a bit of data. If it detects that its 1 bit has been overwritten with a 0 bit, it aborts its transmission until the end of the winner's transmission. At the end of the winner's transmission, the aborted processor immediately transmits the 1st bit of the data again and the algorithm is repeated. It is easy to show that the messages are transmitted in an ascending order and the algorithm is non-blocking.

Figure 1 shows an example with 4 processors sending the values 1, 3, 4, 5. As shown in Figure 1, the messages are transmitted in an ascending order.

It is easy to show that the algorithm is non-blocking. Since a processor is allowed to attempt to transmit data only when it detects that the channel is idle, there can be two groups of processors ready to send data: those in the current bucket and those in the next bucket. Processors in the current bucket are the processors which originally attempted to transmit data and aborted the transmission. since there can be only a finite number of processors in the current bucket, they are guaranteed to transmit data tin bounded time. Processors which become ready to transmit data during the transmission of the current bucket become the next bucket. At the end of the transmission of the current bucket and *an idle period*, the next bucket becomes a current bucket. A similar protocol called CAN(Control Area Network) has been proposed for a real-time system[9].

A parallel computer using BMLMA protocol may be organized to accommodate the hierarchical nature of complex computational processes(figure 3). A cluster is formed by a number of processors sharing a global bus. These cluster are hierarchically connected to from a tree. As Shown in Figure 2, the topology of the machine is a multiple global bus organized

as a tree. Each node of the machine is partitioned into two separate units, the Application Processor(AP) and the Network Communication Module (NCM) (Figure 2). The NCM is a communication handler designed through the interconnection network. Thus, the interconnection network is transparent to the AP. while the NCM would be a special-purpose custom-made chip common to all nodes[13]. the AP of the machine may be any processor appropriate to the given



NCM : Network Communication Module
AP : Application Processor

Figure 2 : Processor node of a parallel computer using BMLAM inter-connection network

application. The detailed organization of the machine would be beyond the scope of this paper.

4. Distributed Best-First Branch-and-Bound Algorithm

The new distributed branch-and-bound algorithm maintains the OPEN list locally and achieves the global optimum gradually by utilizing the logical associative memory-property of the BMLMA communication system. The main problem of parallel branch-and-bound algorithms is that processors are idle when they are blocked by a processor accessing the global OPEN. The distributed branch-and-bound algorithm takes the approach that the processors process with the best choice given to them at a given moment instead of idling while waiting for the optimal best choice. The worst case is that the processor processes an unnecessary sub-problem which might have been terminated without further evaluation if it were processed in the proper sequence; thus the processor wasted its entire processing. This is almost the same as if the processor would have been idle. So there is no negative effect.

best choice from its local OPEN list, processes it, and expands it into the local OPEN list. Therefore the AP is never idle except when there are no more sub-problems to process.

The global balancing and redistribution of the local OPEN lists are achieved by the NCM. Each NCM selects the lowest bound element (first element in the local OPEN list) and attempts to transmit it transmit it to the global bus. Once a processor successfully transmits an element, it selects the next one and immediately tries to transmit it at the next time slot. In each group of processors sharing a virtual bus, processors are assigned a unique logical processor number. The first element successfully transmitted in the global bus is received by the logical number 1 processor in the group The next one is received by the next logical number processer and so on. NCM inserts the element received into the local OPEN list. It is worthwhile to note that there are multiple busses for a node a node in a machine. Thus, the processors achieve global balance of the OPEN lists at different levels simultaneously.

Since this distributed branch-and-bound algorithm is not an optimal best-first branch-and-bound algorithm, the number of iterations required to process increases with the number of processors in the system. But it is expected that if the problem size is large, the algorithm can provide speedup approaching the number of processors in the system. Moreover, this algorithm is expected to always improve the average performance (overall speedup) of the parallel computer with the increase of the number of processors in the system. The other parallel branch-and-bound algorithm actually reduce the system performance after the saturation number of processors due to the communication and global OPEN list contention overheads [4].

5. Simulation Results

To empirically demonstrate the distributed branch-and-bound algorithm, computer simulations have been carried out to solve the Traveling Salesman Problem (TSP). There are a number of algorithms to solve TSP based on the strategy of

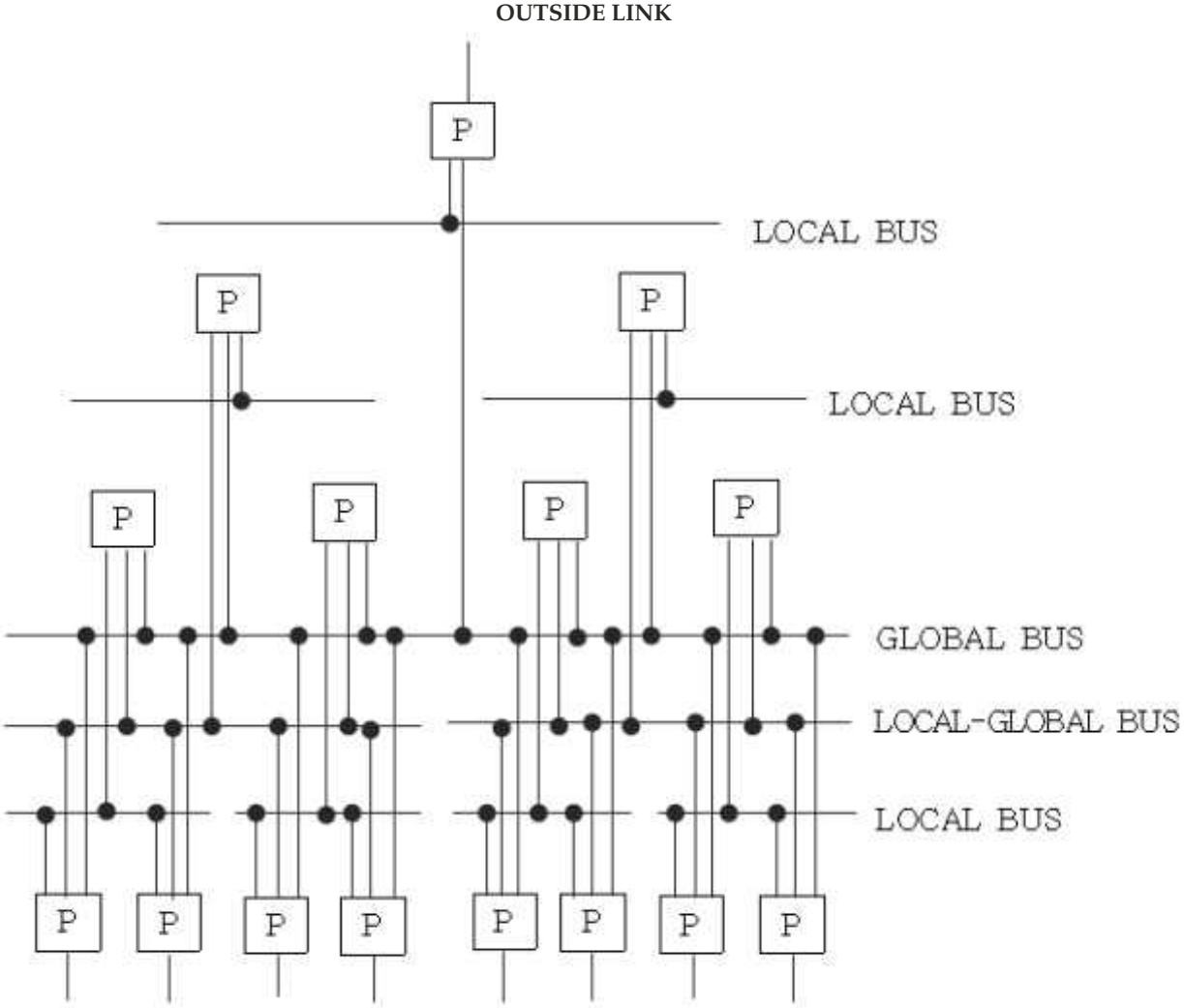


Figure 3 : A organization of parallel computer using BMLMA inter-connection network

calculating the lower bound (under-estimate) distance to the destination. An algorithm based on the *assianment problem* was used in these simulation. [6] The algorithm estimates the lower bound of remaining distances by computing the minimum distances from each of the unvisited cities to other unvisited cities. The algorithm uses a dynamic state space tree. Associated with each node in the state space tree is a graph. Each node represents a subproblem requiring to find a minimum length tour in the graph associated with that node. The original graph G = (V, E) is associated with the root node. A lower bound con the length of a shortest tour in the graph H = (V, A) associated with any node X is obtained by solving the following problem :

$$minimize \sum_{m}^{j=1}\sum_{i=1}^{m} c_{ij}x_{ij}$$

subject to $$\sum_{i=1}^{m} x_{ij} = 1,\ 1 \le j \le m \quad (1)$$

$$\sum_{i=1}^{m} x_{ij} = 1,\ 1 \le i \le m \quad (2)$$

for $$x_{ii} = 0,\ if\ < i, j > \notin A$$

$$x_{ij} = 0 \text{ or } 1 \le i \le m,\ 1 \le j \le m$$

where c_{ij} is the length of edge $< i, j >$ and $c_{ij} = \infty$ if $< i, j > \notin E$. However, usually, the solution will be made up to several disjoint cycles. One of these cycles is used to partition the solution space of H. The details of the solution can be found in [6].

The distributed branch-and-bound algorithm has no communication overhead because the algorithm allows application processors in the system to process the branch-and-bound algorithm cased on the local OPEN lost concurrently with the communication of the network. Thus, the speedup of the algorithm is primarily dependent on the total number of evaluations, N_s, performed by processors in the system. If the total number of evaluations is equal to that of the single processor N_s, the speedup would be n, the number or processors in the system. But due to the anomalies (mainly deceleration anomalies), the total number of evaluations of the distributed branch-and-bound algorithm is expected to be more than that of the single processor. Then the speedup (S_n) is

$$n \cdot \frac{N_s}{N_p}$$

The simulation program calculates the total number of evaluations performed by processor for various number of processors in the system. Due to the system resource limitations of the computer, up to 15 city TSPs have been simulated. As shown in Figure 4, the simulation results clearly demonstrate the superior performance of the distributed branch-and-bound algorithm beyond the usual 20 to 40 processor limitation of parallel branch-and-bound algorithms.

6. Conclusions

In this paper, we have demonstrated that best-first heuristic search can be implemented on a large of loosely-coupled processors. The algorithm is totally distributed and asynchronous. It is based on an associative BLACKBOARD is logically created by using the associative processing property of the BMLMA communication.

Computer simulations have validated the linear scalability of the architecture. The linear scalability of the architecture for distributed heuristic search is important because there are many intractable NP-hard problems which can be solved only by combining approximation and massively parallel processing.

7. Reference

1. *Glaber, C., etc "Properties of NP-complete sets", Proceedings. 19th IEEE. Annual conference on Computational Complexity, 21-24 June 2008, Pages:184 - 197*
2. *Benjamin W. wah and Y. W. Eva Ma,, "MANPI - A Multicomputer Architecture for Solving Combinatorial Extremum-Search Problems". IEEE Trans. Comput., Vol. C-33, No 5, MAay 2007, pp. 377-390.*
3. *George Luger, "Artificial Intelligence: Structures and Strategies for Complex Problem Solving", 5/E, Addison-Wesley Copyright: 2005*
4. *S. R. Huang L. S. Davis, "A Tight Upper Bound for the Speedup of Parallel Best-First Brand-and-Bound Algorithms", Tech. Rept. DACA 76-84-C-0004, Univ. of Maryland, May, 2004*
5. *Michael Fine and F. A. Tobaji, "Demand Assignment Multiple Access Shemes in Broadcast Bus Local Area Networks", IEEE Trans. On Comput., Vol. C-33, No. 12, December 2004, pp. 1130-1158.*
6. *Anany V. Levitin, "Introduction to the Design and Analysis of Algorithms", Addison-Wesley: 2003*
7. *D. Roten, N. Santoro, and J. B. Sidney, "Distributed Sorting", IEEE Trans. comput., Vol. C-34, No. 4, Apr. 1985, pp. 372-376*
8. *Simon Y. Berkovich and Colleen Roe Wilson, " A Computer Communication Technique Using Content-Induce Transaction Overlap", ACM Trans. On Comput., Vol. 2, No. 1, February 1984, pp. 60-77*
9. *Thomas Nolte, etc, "Using bit-stuffing distributed in CAN analysis", IEEE Real-Time Emnbedded Systems, Dec 3, 2001*
10. *E. H Rothauser and D. Wild, "MLMA - A Collision-Free Multi-Access Method", Proc. to 1977 IFIP Congress, 1977, pp. 431-436*
11. *T. H. Lai and S. Sahni, "Anomalies in Parallel Branch-and-Bound Algorithms", Comm. ACM, Vol. 27, No 6, June 1984, pp. 594-602*
12. *G. J. Li, and B. Wah, "Coping with Anomalies in Parallel Branch-and-Bound Algorithms" IEEE Trans. Comput, Vio. C-35, No. 6, June 1986, pp. 568-573.*
13. *You-Keun Park, "the CITO Clock Transmitter", Allied-Signal ATC, Invention Number 450-87-011, Nov., 1987.*
14. *V.N Rao, V. Kumar, and K. Ramesh, "Parallel Heuristic Search on Shared Memory Multiprocessors", MCC-University Research Symp., July 14, 1987, austin, Texas.*
15. *S. Sahni, "General Techniques for combinational approximation", Oper. Res., Vol. 25, No. 6, 1977, pp. 930-936*