Branch-and-Bound-Based Fast Optimal Algorithm for Multiuser Detection in Synchronous CDMA

J. Luo, K. Pattipati, P. Willett, L. Brunel

Abstract— **A fast optimal algorithm based on the branch and bound (BBD) method is proposed for the joint detection of binary symbols of K users in a synchronous Code-Division Multiple Access (CDMA) channel with Gaussian noise. Relationships between the proposed algorithms (depth-first BBD and fast BBD) and both the decorrelating decision feedback (DF) detector and sphere decoding (SD) algorithm are clearly drawn. It turns out that decorrelating DF detector corresponds to a "one-pass" depth-first BBD; sphere decoding is in fact a type of depth-first BBD, but one that can be improved considerably via tight upper bounds and user ordering as in our fast BBD.**

I. Introduction

ALTHOUGH the Maximum Likelihood (ML) multiuser detection in synchronous CDMA is generally NP-hard [1], the optimal algorithm often serves as a benchmark against which to evaluate the sub-optimal algorithms. Since the multiuser detection problem can be viewed as a binary quadratic programming problem, smart search techniques, such as a branch and bound (BBD) method based on tight lower and upper bounds and user ordering, can speed up the solution process significantly. Prior research on using BBD algorithm to multiuser detection includes [5]. An optimal algorithm based on sphere decoding (SD) was also proposed recently in [6] and [7]. These results show that the average computational cost can be significantly less than that of the worst case one for an optimal multiuser detector.

Prior research on optimal multiuser detection used only the quadratic cost function and the binary constraints on user signals. Problem-domain information in the form of matched filter outputs being generated from a known statistical model is essentially ignored. In this paper, we propose a fast optimal BBD algorithm, and show that using the statistical information in the matched filter outputs significantly reduces the average computational cost of the optimal multiuser detector.

II. Problem Formulation and Existing Methods

A discrete-time equivalent model for the matched-filter outputs at the receiver of a CDMA channel using BPSK modulation is given by the K-length vector [1]

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{b} + \boldsymbol{n} \tag{1}$$

where $\boldsymbol{b} \in \{-1,+1\}^K$ denotes the K-length vector of bits transmitted by the K active users. Here $\boldsymbol{H} = \boldsymbol{W}\boldsymbol{R}\boldsymbol{W}$ is a nonnegative definite signature waveform correlation matrix, $\boldsymbol{R}$ is the normalized correlation matrix, $\boldsymbol{W}$ is a diagonal matrix whose k^{th} diagonal element, w_{kk}, is the square root of the received signal energy per bit of the k^{th} user, and $\boldsymbol{n}$ is a real-valued zero-mean Gaussian random vector with a covariance matrix $\sigma^2\boldsymbol{H}$. Letting $\boldsymbol{H} = \boldsymbol{L}^T\boldsymbol{L}$ be the Cholesky decomposition of $\boldsymbol{H}$, the system can also be represented by a white noise model

$$\tilde{\boldsymbol{y}} = \boldsymbol{L}^{-T}\boldsymbol{y} = \boldsymbol{L}\boldsymbol{b} + \boldsymbol{v} \tag{2}$$

where $\boldsymbol{v} = \boldsymbol{L}^{-T}\boldsymbol{n}$ is a white Gaussian noise with zero mean and covariance $\sigma^2\boldsymbol{I}$.

When all the user signals are equally probable, the optimal solution of (1) is the output of a ML detector [1]

$$\phi_{ML} : \hat{\boldsymbol{b}} = \arg \min_{\boldsymbol{b}\in\{-1,+1\}^K} \left(\boldsymbol{b}^T\boldsymbol{H}\boldsymbol{b} - 2\boldsymbol{y}^T\boldsymbol{b}\right) \tag{3}$$

The decorrelating DF method is described in [3]. If we denote the i^{th} component of a vector $\boldsymbol{y}$ by y_i and denote the $(i,j)^{th}$ component of a matrix $\boldsymbol{A}$ by a_{ij}, the decorrelating DF detector can be characterized by

$$\phi_{DF} : \hat{\boldsymbol{b}} = \boldsymbol{P}\tilde{\boldsymbol{b}},\ \tilde{b}_i = \text{sign}\left(\sum_{j=1}^{K} f_{ij}[\boldsymbol{P}\boldsymbol{y}]_j - \sum_{j=1}^{i-1} a_{ij}\tilde{b}_j\right) \tag{4}$$

where $\boldsymbol{F} = \text{U}\left([\boldsymbol{P}\boldsymbol{H}\boldsymbol{P}^T]^{-1}\right)$, $\boldsymbol{A} = \text{L}(\boldsymbol{F}\boldsymbol{P}\boldsymbol{H}\boldsymbol{P}^T)$. Here, U$(\cdot)$ represents the upper triangular part of a matrix, L$(\cdot)$ represents the strictly lower triangular part of a matrix, and $\boldsymbol{P}$ is a permutation matrix. The choice of $\boldsymbol{P}$ has been discussed in Theorem 1 of [3].

III. Optimal Algorithm Based on Depth-first BBD (Simplified Version of Fast BBD)

The idea of using a BBD method in solving combinational optimization problems is already well known [4]. In multiuser detection, BBD method with breadth-first search has been used in [5] to find the minimum distance. In this section, we present an optimal algorithm based on BBD with depth-first search. We point out the relationship between the proposed depth-first BBD method, the decorrelating DF detector, as well as the SD detector [6] [7]. For the convenience of the readers, the algorithm presented in this section is a simplified version of the fast BBD algorithm. The fast optimal algorithm that fully utilizes the statistical information in (1) is proposed and studied in section IV.

J. Luo, K. Pattipati, P. Willett are with the ECE Dept., Univ. of Connecticut, Storrs, CT06269, USA. L. Brunel is with Mitsubishi Electric ITE, 80 av. des Buttes de Coesmes, 35700 Rennes, France. Contact authors' e-mail:krishna@engr.uconn.edu

[1]This work was supported by the Office of Naval Research under contract #N00014-98-1-0465, #N00014-00-1-0101, and by NUWC under contract N66604-1-99-5021

A. Depth-first BBD Algorithm

Since $\boldsymbol{H}^{-1} = \boldsymbol{L}^{-1}\boldsymbol{L}^{-T}$, define $\boldsymbol{D} = \boldsymbol{L}\boldsymbol{b}$, we have

$$\begin{aligned}\phi_{ML} : \hat{\boldsymbol{b}} &= \arg \min_{\boldsymbol{b}\in\{-1,+1\}^K} \|\boldsymbol{L}\boldsymbol{b} - \tilde{\boldsymbol{y}}\|_2^2 \\ &= \arg \min_{\boldsymbol{b}\in\{-1,+1\}^K} \sum_{k=1}^{K} (D_k - \tilde{y}_k)^2 \end{aligned} \quad (5)$$

Here, since $\boldsymbol{L}$ is a lower triangular matrix, D_k depends only on $(b_1, b_2, \ldots, b_k)$. When the decisions for the first k users are fixed, the term $\xi_k = \sum_{i=1}^{k}(D_i - \tilde{y}_i)^2$ can serve as a lower bound of (5). It can be easily seen that the lower bound is achievable when the binary constraints on $(b_{k+1}, \ldots, b_K)$ are disregarded. The BBD tree search to find the minimum value of $\|\boldsymbol{D} - \tilde{\boldsymbol{y}}\|_2^2$ is described below.

Similar to a general BBD method [4], the algorithm maintains a node stack called $OPEN$, and a scalar called $UPPER$, which is equal to the minimum feasible cost found so far, i.e., the "current-best" solution. Define k to be the level of a node (virtual root node has level 0). Label the branch which connects the two nodes $(b_1, \ldots, b_{k-1})$ and $(b_1, \ldots, b_k)$ with $D_k(b_1, b_2, \ldots, b_k)$. The node $(b_1, \ldots, b_k)$ is labeled with the lower bound ξ_k. Also, define $\boldsymbol{z}_k = \tilde{\boldsymbol{y}} - \sum_{i=1}^{k} b_i \boldsymbol{l}_i$, where $\boldsymbol{l}_i$ denotes the ith column of $\boldsymbol{L}$. Denote $[z_k]_j$ as the jth component of vector $\boldsymbol{z}_k$. The depth-first BBD algorithm proceeds as follows.

Depth-first BBD Algorithm (Simplified Version of the Fast BBD):

1) Order users according to theorem 1 of [3], which is also presented in Proposition 2 of section IV below. Compute $\boldsymbol{y}$, $\boldsymbol{H}$ and $\boldsymbol{L}$ matrices for the ordered system.

2) Precompute $\tilde{\boldsymbol{y}} = \boldsymbol{L}^{-T}\boldsymbol{y}$.

3) Initialize $k = 0$. $\boldsymbol{z}_k = \tilde{\boldsymbol{y}}$, $\xi_k = 0$, $UPPER = +\infty$ and $OPEN = NULL$.

4) Set $k = k + 1$. For both nodes, let $\boldsymbol{z}_k = \boldsymbol{z}_{k-1}$, $\xi_k = \xi_{k-1}$. Choose the node in level k such that $b_k = sign([z_k]_k)$. Set flag $f = 1$.

5) Compute $[z_k]_k = [z_k]_k - b_k l_{kk}$.

6) Compute $\xi_k = \xi_k + (D_k - \tilde{y}_k)^2 = \xi_k + [z_k]_k^2$.

7) If $\xi_k \geq UPPER$ and the $OPEN$ list is not empty, drop this node. Pick the node from the end of the $OPEN$ list, set k, ξ and $\boldsymbol{z}_k$ equal to the stored values associated with this node. Set flag $f = 0$ and go to step 5).

8) If $\xi_k < UPPER$ and $k < K$, $\forall j > k$ precompute $[z_k]_j = [z_k]_j - b_k l_{jk}$. If $f = 1$, append the other node with $b_k = -sign([z_k]_k)$ to the end of the $OPEN$ list, and store the associated k, ξ, $\boldsymbol{z}_k$ together with this node. Go to step 4).

9) If $\xi_k < UPPER$, $k = K$ and the $OPEN$ list is not empty, update the "current-best" solution and $UPPER = \xi_k$. Pick the node from the end of the $OPEN$ list, set k, ξ and $\boldsymbol{z}_k$ equal to the stored values associated with this node. Set flag $f = 0$ and go to step 5).

10) If $\xi_k < UPPER$, $k = K$ and the $OPEN$ list is empty, update the "current-best" solution and $UPPER = \xi_k$.

11) For all other cases, stop and report the "current-best" solution.

The computational cost for step 1) is $\frac{K(K+1)}{2}$ multiplications and $\frac{K(K-1)}{2}$ additions. Steps 5) and 6) need 2 addition and 1 multiplication. Notice that step 1) is outside the BBD search. In step 8), since b_k can only take known discrete values, $b_k\boldsymbol{l}_k$ can be precomputed and stored; hence, only $K - k$ additions are needed to obtain $\boldsymbol{z}_k$. To update the lower bound for a node on level $K-k+1$ ($k = 1, ..., K$), at most $k + 1$ additions and 1 multiplication are needed. In addition, the computational requirements for finding the first feasible solution (also the optimal solution in the noise-free case) are $\frac{K(K+3)}{2}$ multiplications and $K(K + 1)$ additions.

B. Relationship Between the Depth-first BBD and the Decorrelating DF Detector

Proposition 1: The first feasible solution obtained from the above depth-first BBD search is the solution of decorrelating DF method.

Proof: Check Proposition 1 of [8] for the proof.

C. Relationship Between the Depth-first BBD and the Sphere Decoder

Sphere decoder is a well known efficient lattice decoding algorithm and was introduced to the multiuser detection community recently in [6] [7]. In this subsection, we first rewrite the SD algorithm as follows:

Sphere Decoding Algorithm:

1) Compute the Cholesky decomposition matrix $\boldsymbol{H} = \boldsymbol{L}^T\boldsymbol{L}$.

2) Precompute $\tilde{\boldsymbol{y}} = \boldsymbol{L}^{-T}\boldsymbol{y}$, $C = \alpha K\sigma^2$ where α is chosen so that [7]

$$\int_0^{\alpha K} \frac{\lambda^{K/2-1}}{\Gamma(K/2)} e^{-\lambda} d\lambda = 0.99 \quad (6)$$

3) Initialize $k = 0$, $\boldsymbol{z}_k = \tilde{\boldsymbol{y}}$, $\xi_k = 0$. Initialize $UPPER = C$. Initialize $OPEN = NULL$.

4) Set $k = k + 1$. For both nodes, let $\boldsymbol{z}_k = \boldsymbol{z}_{k-1}$, $\xi_k = \xi_{k-1}$. Choose the node in level k such that $b_k = -1$. Append the node with $b_k = +1$ to the end of the $OPEN$ list, and store the associated k, ξ and $\boldsymbol{z}_k$ together with this node.

5) Compute $[z_k]_k = [z_k]_k - b_k l_{kk}$.

6) Compute $\xi_k = \xi_k + (D_k - \tilde{y}_k)^2 = \xi_k + [z_k]_k^2$.

7) If $\xi_k \geq UPPER$ and the $OPEN$ list is not empty, drop this node. Pick the node from the end of the $OPEN$ list, set k, ξ and $\boldsymbol{z}_k$ equal to the stored values associated with this node and go to step 5).

8) If $\xi_k < UPPER$ and $k < K$, for $j = k + 1$, precompute $[z_k]_j = [z_k]_j - \sum_{i=1}^{k} b_i l_{ji}$. Go to step 4).

9) If $\xi_k < UPPER$, $k = K$ and the $OPEN$ list is not empty, update the "current-best" solution and $UPPER = \xi_k$. Pick the node from the end of the $OPEN$ list, set k, ξ and $\boldsymbol{z}_k$ equal to the stored values associated with this node and go to step 5).

10) If $\xi_k < UPPER$, $k = K$ and the $OPEN$ list is empty, update the "current-best" solution and $UPPER = \xi_k$.

11) If no solution is available yet, let $C = 2C$ and go to step 3). Otherwise, stop and report the "current-best" solution.

Although written in a different form with a different notation, it is easy to show that the above algorithm is indeed identical to the SD methods proposed in [6] and [7] [1]. Apparently, the SD method can be categorized as a depth-first BBD algorithm. The major differences between the SD and the proposed depth-first BBD algorithm, however, are in steps 1), 3), 4) and 8). The lower bound update is also identical to the breadth-first BBD algorithm proposed in [5].

As we have shown before, the choice of b_k in step 4) of the proposed depth-first BBD corresponds to the solution of the DF detector. This guarantees a minimum computational cost when the system is noise-free. It is also a key step that allows the fast optimal BBD algorithm (proposed in the next section) to further use of the statistical information. However, these enhancements are not exploited in the SD algorithm because statistical information in the model is ignored in step 4).

The user ordering corresponding to step 1), the lower bound computations corresponding to step 8), the upper bound initialization corresponding to step 3) are studied in the next section.

IV. Fast Optimal BBD Algorithm Using the Statistical Information (Full Version)

A key feature of multiuser detection is that the matched-filter output $\boldsymbol{y}$ is generated from a statistical model given by (1). Typically, the variance of the noise is not very large, which means that a significant fraction of optimal multiuser detection problems can be solved easily. In this section, we present the full version of the fast optimal algorithm. The key ideas of utilizing the statistical information are: the user ordering, the search strategy and the lower bound computation.

The BBD search can be separated into two stages. We term the first stage the "search" stage where the "current-best" solution is not the optimal solution. The second stage is termed the "confirm" stage where the "current-best" solution is optimal, but the algorithm needs to confirm that it is indeed better than any other solution.

Assume that the true solution $\bar{\boldsymbol{b}}$ is also the maximum likelihood solution. In the "confirm" stage, we have

$$UPPER = \left\| \boldsymbol{L}\bar{\boldsymbol{b}} - \boldsymbol{L}^{-T}\boldsymbol{y} \right\|_2^2 = \|\boldsymbol{v}\|_2^2 \tag{7}$$

Asymptotically, $\|\boldsymbol{v}\|_2^2 \to 0$.

[1] Two different upper bound initializations are proposed for the SD algorithm in [6] and [7]. According to computer simulations, the average computational complexity of the SD in [6] is higher than the one in [7], both in the low and the high SNR regimes. Bounding and sphere-enlargement parameters respectively in steps 2) and 11) may be better coordinated, but no suggestions of such tuning have appeared to date.

Now, consider any other branch associated with vector $(b_1, b_2, \ldots, b_k)$. Without loss of generality, suppose $\forall j < k$, $b_j = \bar{b}_j$, and $b_k \neq \bar{b}_k$. The lower bound is

$$\xi_k = \sum_{i=1}^{k} (D_i - \tilde{y}_i)^2 = \sum_{i=1}^{k-1} (v_i)^2 + (v_k \pm 2l_{kk})^2 \tag{8}$$

Apparently, when $\sigma \to 0$, $\xi_k \to 4l_{kk}^2$. This shows that, asymptotically, whenever the algorithm enters the "confirm" stage, all the branches will be discarded with a high probability.

A. User Ordering

According to the above intuitive analysis, the major task in the "search" stage is to maximize the probability that the "current-best" solution is optimal, so that the algorithm can enter the "confirm" stage as soon as possible. For the DF detector, which gives the first feasible solution in the fast BBD, define $P_e(k)$ to be the probability of error on user k given that all the decisions on users $1, \ldots, k-1$ are correct. We have from [3],

$$P_e(k) = Q(\frac{l_{kk}}{\sigma}) \tag{9}$$

In the high SNR regime, the probability of error of the DF solution is dominated by the user corresponding to the minimum diagonal element of $\boldsymbol{L}$.

Proposition 2: The user ordering presented in Theorem 1 of [3] maximizes the asymptotic probability that all decisions of the decorrelating DF detector are correct, i.e., it maximizes the probability that the first feasible solution in the fast BBD algorithm is optimal.

Proof: See Proposition 1 of [2].

B. Search Strategy

In the high SNR regime, defining $P_e(1^{st})$ to be the probability of error of the first feasible solution, we have $P_e(1^{st}) = \sum_{i=1}^{K} Q(\frac{l_{ii}}{\sigma})$. Define

$$\begin{aligned} m_1 &= \arg\min_j l_{jj} \\ m_i &= \arg \min_{j \neq m_1, \ldots, m_{i-1}} l_{jj} \end{aligned} \tag{10}$$

Given that the first feasible solution is not optimal, user m_1 has a high probability to be the erroneous user since $Q(\frac{l_{m_1 m_1}}{\sigma})$ dominates $P_e(1^{st})$. Consequently, swapping the decision on user m_1 and applying DF detection to find the second feasible solution is the best choice. The probability that neither the first nor the second feasible solutions is optimal is given by $P_e(2^{nd}) = \sum_{i \neq m_1} Q(\frac{l_{ii}}{\sigma})$. Similarly, m_2 is the next user we should search, and m_3 should be searched after m_2, etc.

Apparently, unlike the search strategy in the depth-first BBD, which searches nodes in descending order of their levels in the tree, the optimal search strategy visits nodes in ascending order of the values of the diagonal elements of $\boldsymbol{L}$ matrix.

Due to the dynamic choice of which node to explore next, the worst case storage requirement (i.e., of previously-visited-node data) is exponential. The fixed search strategy of the depth-first BBD, including the SD, obviates this; but likewise do other versions of BBD that perform a smart search only on certain users. Extreme demands on memory are rare, but if they are of concern it is worthwhile to recall that there is a continuum of trade-offs between memory and speed.

C. Computational Enhancement

Step 8) of the SD algorithm precomputes part of the lower bounds for the sibling nodes. However, it does not take advantage of the fact that other nodes may also share part of the computations. The depth-first BBD method does pre-computing for all the nodes under the same branch. However, if the branch is discarded, the pre-computing itself is a waste of computational resources.

In the high SNR regime, since the error performance of the DF detector is characterized by the diagonal elements of $\boldsymbol{L}$, it is reasonable to make the following assumption in the BBD search:

Assumption: If a branch on level k is accepted (not discarded), then the sub-branches on levels $k+1,\ldots,k+m$ may also be accepted with a high probability as long as $\forall k < j \leq k+m,\ l_{jj} < l_{kk}$.

Based on this assumption, suppose for user k, u_k and d_k are defined as

$$u_k = arg \min_{0<i<k} (\forall i \leq j < k,\ l_{jj} < l_{kk}) \tag{11}$$

or $u_k = k$ if no solution can be found in (11)

$$d_k = arg \max_{k<i\leq K} (\forall k < j \leq i,\ l_{jj} \leq l_{kk}) \tag{12}$$

or $d_k = k$ if no solution can be found in (12)

Similar to the simplified fast optimal algorithm, we also keep a vector $\boldsymbol{z}$. When computing the lower bound ξ_k, we precompute for users $k+1,\ldots,d_k$

$$\forall k < j \leq d_k, [z_k]_j = [z_{k-1}]_j - \sum_{i=u_k}^{k} b_j l_{ji} \tag{13}$$

i.e., precomputing involves only the block in $\boldsymbol{L}$ with rows $[k+1, d_k]$ and columns $[u_k, k]$.

D. Fast Optimal BBD Algorithm (Full Version)

Similar to the depth-first BBD, the user ordering is precomputed offline, and we assume that all matrices are properly precomputed for the ordered system. In order to implement the new search strategy, instead of using the $OPEN$ stack, in the full version, we have K queues, termed $\boldsymbol{q}_1,\ldots,\boldsymbol{q}_K$. Queue $\boldsymbol{q}_k$ is associated with user k. The nodes in a queue follow the "first-in-first-out" rule, i.e., nodes enter from the tail and are taken from the head of the queue. In addition, we order queues according to the values of the diagonal elements of $\boldsymbol{L}$, i.e., in the BBD search, we take nodes from the queues in the order $[\boldsymbol{q}_{m_1},\ldots,\boldsymbol{q}_{m_K}]$, where $m_1,\ldots,m_K$ are defined in (10).

To implement the proposed method, for each user k, the block margins u_k and d_k are precomputed and stored in vectors $\boldsymbol{u}$ and $\boldsymbol{d}$.

The full version of the fast optimal BBD algorithm proceeds as follows:

Fast Optimal BBD Algorithm:

1) Order users according to theorem 1 of [3], which is also presented in Proposition 2; Compute $\boldsymbol{y}$, $\boldsymbol{H}$ and $\boldsymbol{L}$ matrices for the ordered system; precompute the vectors $\boldsymbol{u}$ and $\boldsymbol{d}$, the components of which are defined by (11) and (12).

2) Precompute $\tilde{\boldsymbol{y}} = \boldsymbol{L}^{-T}\boldsymbol{y}$.

3) Initialize $k = 0$. $\boldsymbol{z}_k = \tilde{\boldsymbol{y}}$, $\xi_k = 0$, $UPPER = +\infty$ and initialize K queues by $\forall k$, $\boldsymbol{q}_k = NULL$.

4) Set $k = k+1$. For both nodes, let $\boldsymbol{z}_k = \boldsymbol{z}_{k-1}$, $\xi_k = \xi_{k-1}$. Choose the node in level k such that $b_k = sign([z_k]_k)$. Set flag $f = 1$.

5) Compute $[z_k]_k = [z_k]_k - b_k l_{kk}$.

6) Compute $\xi_k = \xi_k + (D_k - \tilde{y}_k)^2 = \xi_k + (z_k)_k^2$.

7) If $\xi_k \geq UPPER$ and not all the queues are empty, drop this node. Go to step 11).

8) If $\xi_k < UPPER$ and $k < K$, do

8.1) If $f = 1$, for both nodes in level k,

8.1.1) If $d_k > k$, precompute $\forall k < j \leq d_k$, $[z_k]_j = [z_k]_j - \sum_{i=u_k}^{k-1} b_i l_{ji}$

8.1.2) If $d_k = k$, precompute $j = k+1$, $[z_k]_j = [z_k]_j - \sum_{i=u_{k+1}}^{k-1} b_i l_{ji}$

8.1.3) Append the node $b_k = -sign([z_k]_k)$ to the tail of queue $\boldsymbol{q}_{m_k}$, and store the associated k, ξ and $\boldsymbol{z}_k$ together with this node.

8.2) If $d_k > k$, precompute $\forall k < j \leq d_k$, $[z_k]_j = [z_k]_j - b_k l_{jk}$

8.3) If $d_k = k$, precompute $j = k+1$, $[z_k]_j = [z_k]_j - b_k l_{jk}$

8.4) Go to step 4).

9) If $\xi_k < UPPER$, $k = K$ and not all the queues are empty, update the "current-best" solution and $UPPER = \xi_k$. Go to step 11).

10) If $\xi_k < UPPER$, $k = K$ and all the queues are empty, update the "current-best" solution and $UPPER = \xi_k$; Go to step 12).

11) Pick one node from the queues (note that we should check queues in the order of $\boldsymbol{q}_{m_1},\ldots,\boldsymbol{q}_{m_K}$). Set k, ξ and $\boldsymbol{z}_k$ equal to the stored values associated with this node. Set $f = 0$, go to step 5).

12) Stop and report the "current-best" solution.

E. Non-Binary BBD

In a system where user signals are taken from a finite alphabet, the fast optimal BBD algorithm can be applied directly. In step 5), we should choose the node associated with b_k that minimizes $|[z_k]_k - b_k l_{kk}|$. Step 8.1) should be applied to the rest of the nodes, which should be sorted in an ascending order according to the values of $|[z_k]_k - b_k l_{kk}|$.

Alternatively, one could choose to treat an M-ary user as several binary users (e.g. $M = 8$ equivalent to 3 binary users). Both philosophies are applicable to any BBD method, including sphere decoding.

F. Upper Bound Initialization

The upper bound initialization in step 3) of the SD method is the only place where the statistical information is used. Beside the drawback of requiring σ, it is the key step in SD that ensures the asymptotic average computational cost be polynomial [7].

Assume that the true solution $\bar{\boldsymbol{b}}$ is also maximum likelihood, the optimal cost is given by (7). Approximate $UPPER = \|\boldsymbol{v}\|_2^2$ by a Gaussian random variable, we have $\|\boldsymbol{v}\|_2^2 \sim N(K\sigma^2, 2K\sigma^4)$. Therefore, when σ is available, to further reduce the average computational cost, we recommend modifying steps 2), 3) and 12) in the fast BBD algorithm as follows:

2) Precompute $\tilde{\boldsymbol{y}} = \boldsymbol{L}^{-T}\boldsymbol{y}$. Precompute $C = K\sigma^2 + \sqrt{2K}l_{m_1m_1}\sigma$.

3) Initialize $k = 0$. $\boldsymbol{z}_k = \tilde{\boldsymbol{y}}$, $\xi_k = 0$, $UPPER = C$ and initialize K queues by $\forall k$, $\boldsymbol{q}_k = NULL$.

12) If no solution is available so far, let $C = C + \sqrt{2K}l_{m_1m_1}\sigma$ and go to step 3). Otherwise, stop and report the "current-best" solution.

Although the effect in the high SNR regime is limited, when SNR is moderate, the recommended upper bound initialization saves up to $\frac{1}{3}$ of the average computational cost of the fast BBD algorithm (speed-up of 33%).

V. Simulation Results

Example 1: In this example, we vary the number of users from 5 to 60. The ratio between the number of users and the signature length is fixed at $\frac{5}{6}$. The binary signature sequences are randomly generated, and the user signal powers are set to be equal. For the suboptimal detectors, the SNR is fixed at $14.77db$. Upper bound initialization is used by the fast BBD algorithm. A comparison of the average computational costs of the fast optimal BBD algorithm, the SD method, the decorrelator as well as the decorrelating DF method at different SNRs are shown in Figure 1. When $SNR = 14.77db$, the average computational cost of the fast BBD algorithm is comparable to that of the decorrelator for up to 60 users, and is significantly better than that of the SD method with the same SNR.

Example 2: In this example, we have 50 equal-powered users. The 53-length binary signature sequences are randomly generated. Figure 2 shows the average computational costs versus SNR, together with the 10^{th} and the 90^{th} percentile curves, for the fast BBD and the SD algorithms. Upper bound initialization is used in the fast BBD algorithm. The computations of the SD method with $SNR < 14.34db$ is not shown on the figure due to its excessively high complexity.

VI. Conclusion

A fast BBD-based optimal algorithm for the multiuser detection of symbol synchronous CDMA is proposed. Due to the use of statistical information embedded in the system model, the average computational cost has been significantly reduced. The fast BBD is found to outperform the SD for all SNRs and all numbers of users in the binary signaling case. Comparisons for the M-ary modulation cases are in progress.

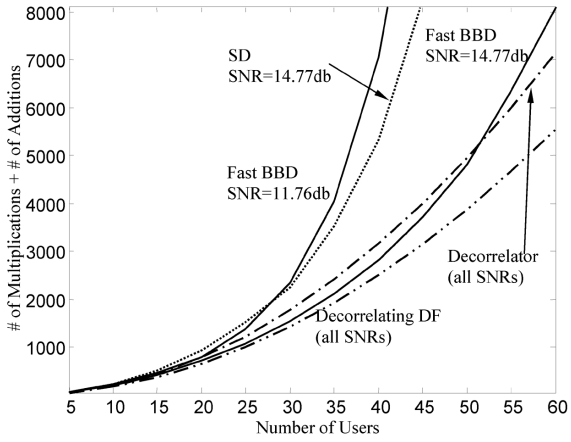

Fig. 1. Average computational costs vs. number of users. (Random signatures, $\frac{\text{Number of Users}}{\text{Signature length}} = \frac{5}{6}$, 10^4 Monte-Carlo runs.)

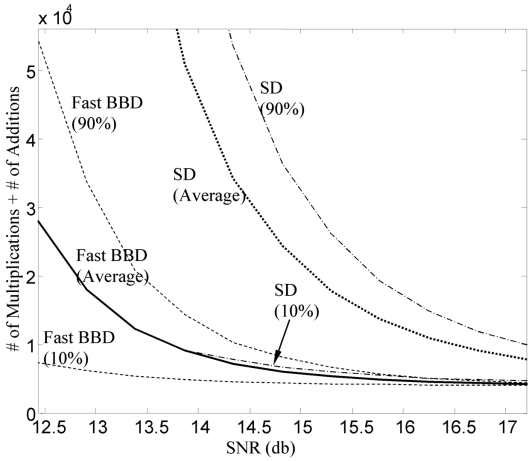

Fig. 2. Average computational cost vs. SNR. (50 users, 53-length random signatures, 10^4 Monte-Carlo runs.)

References

[1] S. Verdu, *Multiuser Detection*, Cambridge University Press, New York, 1998.

[2] J. Luo, K. Pattipati and P. Willett, *Optimal Grouping Algorithm for a Group Decision Feedback Detector in Synchronous CDMA Communications*, to appear in IEEE Trans. Comm.

[3] M. K. Varanasi, *Decision feedback multiuser detection: a systematic approach*, IEEE Trans. Inform. Theory, Vol. 45, pp. 219-240, Jan. 1999.

[4] D. Bertsekas, *Network Optimization, Continuous and Discrete Models*, Athena Scientific, Belmont, Massachusetts, Chap. 10, pp. 483-492, 1998.

[5] C. Schlegel and L. Wei, *A simple way to compute the minimum distance in multiuser CDMA systems*, IEEE Trans. Comm., Vol. 45, pp. 532–535, May 1997.

[6] L. Brunel and J. Boutros, *Lattice decoding for joint detection in direct sequence CDMA systems*, Accepted for publication in IEEE Trans. Inform. Theory.

[7] B. Hassibi and H. Vikalo, *On the Expected Complexity of Integer Least-squares Problems*, IEEE ICASSP, Orlando, FL, May 2002.

[8] J. Luo, K. Pattipati, P. Willett and G. Levchuk *Fast Optimal and Sub-optimal Any-Time Algorithms for CDMA Multiuser Detection Based on Branch and Bound*, Submitted to IEEE Trans. Comm., Aug. 2000.