Proceedings of the 40th IEEE
Conference on Decision and Control
Orlando, Florida USA, December 2001

Speed and Accuracy Comparison of Techniques to Solve a Binary Quadratic Programming Problem with Applications to Synchronous CDMA

F. Hasegawa, J. Luo, K. Pattipati, P. Willett [1]

Abstract

We compare methods for solving the NP-hard binary quadratic programming (BQP) problem. Various methods are discussed, including box-constrained quadratic programming, branch and bound, coordinate descent, group decision making and semi-definite relaxation. An algorithm from target-tracking, the Probabilistic Data Association Filter (PDAF), is modified to the BQP application. Simulation results show that this and several other methods can significantly outperform the decision feedback detector (DFD) or its group counterpart, GDFD.

1 Introduction

Binary quadratic programming (BQP) problems arise in a variety of applications, e.g., capital budgeting and financial analysis problems [14], CAD problems [4], traffic message management problems [6], machine scheduling [1], molecular conformation [17], and so on. Of particular current relevance, some digital communication problems, such as synchronous code-division multiple access (CDMA), can be formulated as BQPs.

In the CDMA context, prior research has focused on designing suboptimal receivers with low computational complexity and better performance than a conventional detector. Among them are the multistage detection [19], the group detection [20] and the decision feedback detection [21]. Usually, suboptimal methods need to perform a projection to satisfy the integrality constraints, which can cause detection errors.

Based on the idea of successive cancelation, a systematic Decision Feedback Detection (DFD) approach was given in [21]. While maintaining the computational complexity of $O(K^2)$, DFD methods provide a significant improvement in probability of error when compared with conventional and decorrelation detector [21]. However, computer simulations show that, in most cases, there is still a large gap between the probability of error of DFD outputs and that of the optimal solution.

In the paper, we compare several methods for solving the BQP problem arising in synchronous CDMA. We discuss box-constrained quadratic programming, "best-first" and "depth-first" versions of branch and bound, coordinate descent, group decision feedback detection (GDFD), semi-definite relaxation and Probabilistic Data Association Filter (PDAF). Simulation results show that the PDAF and several other of the proposed methods can significantly outperform the decision feedback detector (DFD) or its group counterpart, GDFD.

2 Problem Formulation

A discrete-time equivalent model for the matched-filter outputs at the receiver of a CDMA channel is given by the K-length vector [12]

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

where $b \in \{-1,+1\}^K$ denotes the K-length vector of bits transmitted by the K users. Here $\mathbf{H} = \mathbf{W}^{\frac{1}{2}}\mathbf{R}\mathbf{W}^{\frac{1}{2}}$ is the signature waveform correlation matrix, $\mathbf{R}$ is the symmetric normalized correlation matrix with unit diagonal elements, $\mathbf{W}$ is a diagonal matrix whose k^{th} diagonal element, w_k, is the received signal energy per bit of the k^{th} user, and n is a real-valued zero-mean Gaussian random vector with a covariance matrix $\sigma^2\mathbf{H}$.

When all the user signals are equally probable, the optimal solution of (1) is the output of a Maximum Likelihood (ML) detector [12]

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

The ML detector has the property that it minimizes, among all detectors, the probability that not all users' decisions are correct. Except in pathological cases, ϕ_{ML} is NP-hard and exponentially complex to implement [22]; the focus is then on developing easily implementable integer programming approaches

[1]The Authors are with the Electrical and Computer Engineering Department, University of Connecticut, Storrs, CT 06269-2157, E-mail:willett@engr.uconn.edu

3 CWMA Detection by Various Algorithms

3.1 Matched Filter

The simplest sub-optimal algorithm is a single-user matched filter [22]. It makes a decision based on the sign of the observation in (1).

$$\phi_{Match} : \hat{b}_M = sign\,(y) \tag{3}$$

It would be the optimal detector if the signature waveforms were orthogonal, i.e., if $\mathbf{H}$ were diagonal. As the simulation results show, this method performs poorly.

3.2 Decorrelator

The conventional decorrelation detector improves on the matched filter output. It is found in two steps [12]. First, the unconstrained solution $\tilde{b} = \mathbf{H}^{-1}y$ is computed, and then this is projected onto the constraint set via: $\hat{b}_i = sign(\tilde{b}_i)\ \forall i$.

3.3 DFD Method

This method improves the probability of detection error by applying a successive cancelation technique on users. The DFD method based on the decorrelation detector is described in [21]. The users are sequentially demodulated by

$$\phi_{DFD}: \quad \tilde{b}_i = sign\left(\sum_{j=1}^{K} F_{ij}\mathbf{P}y_j - \sum_{j=1}^{i-1} B_{ij}\tilde{b_j}\right)$$
$$\hat{b} = \mathbf{P}\tilde{b}, i \in (1, \cdots, K) \tag{4}$$

where $\mathbf{F} = U([\mathbf{PHP}]^{-1}), \mathbf{B} = L(\mathbf{FPHP})$. Here, $U(\cdot)$ represents the upper triangular part of a matrix, $L(\cdot)$ represents the strictly lower triangular part of a matrix, and $\mathbf{P}$ is a permutation matrix (symmetric and orthogonal). The choice of $\mathbf{P}$ has been discussed in Theorem 1 of [21]. Conceptually, the "easiest", or the most powerful user, is detected first.

3.4 Quadratic Programming

The constraint in (2) can be relaxed to simple box-constraints of the form as $-e \leq b \leq e$, where $b, e \in \Re^K$ and $e = [11 \cdots 1]^T$, with the understanding that the constraints be enforced at the final step via a hard-limiting projection operation. Therefore, the minimization problem in (2) can be modeled as a box-constrained quadratic programming problem as follows:

$$\phi_{QP} : \hat{b} = \arg \min_{-1 \leq b_i \leq 1\, \forall i} (b^T\mathbf{H}b - 2y^T b) \tag{5}$$

In our simulations, the Reflective Newton Method [5] is used to solve (5). This method is utilized in **quadprog()** of MATLAB 5.3 Optimization Toolbox to solve large-scale quadratic programming problems with box constraints.

3.5 GDFD with Optimal Grouping

The group detection idea was first introduced by Varanasi in [20], and significantly enhanced in [10]. In DFD successive cancelation is applied using "groups" of cardinality one. In GDFD the DFD notion is generalized: users are grouped into an ordered set of $P(\leq K)$ groups, and decision-making is sequential. That is, dimensions (users) within the first group are decided based on an assumption that all other users contribute Gaussian noise. Then these decisions are assumed correct, and the users' effects are subtracted from the observations: the same procedure can now be implemented with the remaining $P-1$ groups, and we repeat until done. It is not unreasonable to use an optimal algorithm (branch and bound) to solve the subproblems associated with each group, since each is by design of quite low dimension. But the key is to use a clever means to design the groups in the first place; this and implementation concerns are detailed in [10].

3.6 Branch and Bound

The optimal solution to (2) can be obtained by interrogating each of the 2^K possible b's. There are intelligent ways to compute such combinations.

3.6.1 "Depth-First" Branch and Bound:

In [9], an optimal algorithm based on branch and bound with an iterative lower bound update was proposed. Suppose $\mathbf{H} = \mathbf{L^T L}$ is the Cholesky decomposition of $\mathbf{H}$; as shown in [9], (2) is equivalent to

$$\phi_{ML} : \hat{b} = \arg \min_{b \in \{-1,+1\}^K} \|\mathbf{L}b - (\mathbf{L}^{-1})^T y\|_2^2 \tag{6}$$

Denote $\tilde{y} = (\mathbf{L}^{-1})^T y$, $d = \mathbf{L}b$, and denote the k^{th} component of d and $\tilde{y}$ by d_k and $\tilde{y_k}$, respectively. (6) becomes [9]

$$\phi_{ML} : \hat{b} = \arg \min_{b \in \{-1,+1\}^K} \sum_{k=1}^{K} (d_k - \tilde{y}_k)^2 \tag{7}$$

Since $\mathbf{L}$ is a lower triangular matrix, d_k depends only on $(b_1, \ldots, b_k)$. When the decision for the first k users are fixed, the term

$$\xi_k = \sum_{i=1}^{k} (d_i - \tilde{y}_i)^2 \tag{8}$$

becomes a lower bound of (7). The branch and bound algorithm is shown below. Similar to a general branch and bound method [3], this algorithm maintains a node list, $OPEN$, and a scalar called $UPPER$, which is denoted as the "Current-Best" solution, i.e., the minimum feasible cost found so far. Define k to be the level of a node (virtual root node has level 0). Label the branches with $d_k(b_1, b_2, \ldots, b_{k+1})$, which connect the two nodes $(b_1, \ldots, b_k)$ and $(b_1, \ldots, b_{k+1})$. The node is labeled with the lower bound ξ_k. Also, define

$v_k = \sum_{i=1}^{k}[b_i \times (\text{the } i^{th} \text{ column of } \mathbf{L})] - \tilde{y}$, denote $[v_k]_j$ as the j^{th} component of a vector v_k, and L_{ij} as the $(i,j)^{th}$ element of **L**.

```
"Depth-First" Branch and Bound Approach
  Step 1:  Precompute $\tilde{\mathbf{y}} = (\mathbf{L}^{-1})^T y$
  Step 2:  Initialize $k = 0$. $v_k = -\tilde{y}$, $\xi_k = 0$,
           $UPPER = +\infty$ and $OPEN = NULL$
  Step 3:  Set $k = k+1$. Choose the node in level
           $k$ such that $b_k = -sign([v_{k-1}]_k)$.
           If $k < K$, append the node with $b_k =$
           $sign([v_{k-1}]_k)$
  Step 4:  $v_k = v_{k-1} + b_k \times (k^{th}$ column of $\mathbf{L})$
  Step 5:  $\xi_k = \xi_{k-1} + (d_k - \tilde{y}_k)^2 = \xi_{k-1} + (v_k)_k^2$
  Step 6:  Update values as follows:
             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 $k$ equal to the level of this node
                 and go to Step 4
             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 $k$ equal to the level
                 of this node and go back to Step 4
             IF $\xi_k < UPPER$ and $k \neq K$
                 Go back to Step 3
             IF $\xi_k < UPPER$ and $k = K$ and the OPEN
             list is empty
                 Update the "Current-Best" solution
                 and $UPPER = \xi_k$
             For all other cases
                 Stop and output the "Current-Best"
                 solution
```

3.6.2 Pardalos' Branch and Bound: Pardalos branch and bound algorithm, based on the steepest descent algorithm was described in [16]. Here the minimal range of the gradient of the objective function is used to select a starting point and a new value of lower and upper bounds of the branch and bound algorithm. In the worst case, each step in the steepest descent algorithm requires $\mathcal{O}\left(K^2\right)$ operations. This algorithm has been found to be extremely slow for the ML detector problem and will not be discussed further. The method is slower than the "Depth-First" branch and bound by approximately one order of magnitude.

3.6.3 "Best-First" Branch and Bound: The "best-first" search is a slightly different version of "depth-first" approach. As far as we know, this method is first applied to BQP here. It also maintains a node list called OPEN. However, it differs from the "depth-first" approach in that the nodes in the OPEN list are sorted in ascending order of their lower bounds. The algorithm converges to an optimal solution. Several suboptimal variants can be derived by controlling the backtracking in the search process. For example, one could prematurely terminate the search as soon as a fesible solution is found.

3.7 Coordinate Descent Family

The problem in (2) is equivalent to the following [8]:

$$\phi_{cd} : \hat{b} = \arg \min_{b \in \{0,1\}^K} \left(\frac{1}{2} x^T \mathbf{Q} x + c^T x \right) \qquad (9)$$

where $x = \frac{b+e}{2}, c = -(y + \mathbf{H}e), \mathbf{Q} = 2\mathbf{H}$, and $e = [11 \cdots 1]^T$. In the *Descent I* algorithm, the largest decrease in $w(i)$ is sought by changing a variable of the form $x_i = 1 - x_i$ at each step where

$$w(i) = \left(\sum_{j=1, j \neq i}^{K} q_{ij} x_j + c_i + \frac{1}{2} q_{ii} \right) (2x_i - 1). \qquad (10)$$

This process can be viewed as searching for a discrete "greedy" local minimizer in the neighborhood of the point $x = [x_1, \ldots, x_K]$ of the form

$$\left\{ x^k = (x_1, \ldots, x_{k-1}, 1 - x_k, x_{k+1}, \ldots, x_K), k \in [1, K] \right\}$$

Descent II changes two variables at a time; there is a fortunate simplification similar to (10) [8].

3.8 Semi-Definite Programming

It is shown in [13] that semi-definite relaxation is an accurate and efficient approximation method for certain NP-hard problems and it can approximately solve the Maximum-Likelihood Detection problem in $\mathcal{O}(K^{3.5})$. In addition, since the model is convex, it does not suffer from local maxima. Moreover, efficient algorithms, namely interior-point methods, are available for solving the SD problem [18]. We can formulate (2) as a SD problem by writing

$$\hat{\mathbf{X}} = \arg \max_{\substack{\mathbf{X} \succeq 0 \\ X_{ii}=1 \, \forall i}} \left(Trace\left(\mathbf{X} \mathbf{Q}_{SD}\right) \right) \qquad (11)$$

where

$$\mathbf{Q}_{SD} = \begin{bmatrix} -\mathbf{H} & \mathbf{W}^{\frac{1}{2}} y \\ (\mathbf{W}^{\frac{1}{2}} y)^T & 0 \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} b \\ c \end{bmatrix} \begin{bmatrix} b \\ c \end{bmatrix}^{\mathrm{T}} \qquad (12)$$

in which $c \in \{-1, 1\}$ is a randomly-generated scalar. It is also shown that the SD relaxation solution can be converted to an approximate solution for a BQP problem by performing a computationally efficient randomization method [15]. Given a modest number of randomizations, the computational complexity of the randomization is approximately $\mathcal{O}(K^2 M_{rand})$, where M_{rand} denotes the number of randomizations. Implementation of the SD relaxation algorithm to (2) is described in [13]. The semidefinite model in (11) can be realized as follows:

$$\phi_{SD} : \begin{bmatrix} b^* \\ c^* \end{bmatrix} = \arg \max_{\substack{b \in \{-1,+1\}^K \\ c \in \{-1,+1\}}} \left(\begin{bmatrix} b \\ c \end{bmatrix}^T \mathbf{Q}_{SD} \begin{bmatrix} b \\ c \end{bmatrix} \right) \qquad (13)$$

The solution obtained from (11) is then factorized by using Cholesky decomposition $\hat{\mathbf{X}} = \hat{\mathbf{V}}^T\hat{\mathbf{V}}$. In the subsequent step, randomization is performed to obtain an approximate MLD solution.

Semi-definite Relaxation Approach

```
Step 1: Formulate $\mathbf{Q}_{SD}$ as in (12). Randomly generate $c$ in (12) which is either -1 or 1.
Step 2: Solve the semi-definite program in (11). Obtain the solution $\hat{\mathbf{X}}$ . (We used the interior-point method in [7]).
Step 3: Cholesky decompose $\hat{\mathbf{X}} = \hat{\mathbf{V}}^T\hat{\mathbf{V}}$.
Step 4: Perform randomization, where the number of randomizations is $M_{rand}$. Set $k = 1$.
    A 1: Generate a uniformly distributed vector $u^k$:
         $u^k = \frac{w}{\|w\|}, w \sim \mathcal{N}(0, \mathbf{I})$.
    A 2: Form $\bar{x}^k = sign(\hat{\mathbf{V}}^T u^k)$
    A 3: if $k > M_{rand}$, Go to Step 5. If not, $k = k + 1$ and return to A1.
Step 5: Choose $\hat{x} = \bar{x}^j$ as the approximate solution of $x^*$ in (13), where
         $j = \arg \max_{k=1,\ldots,M_{rand}} f(\bar{x}^k)$    (14)
Step 6: Obtain an approximate solution for (2) by
         $\hat{b}_{SDR} = \hat{x}_{K+1} \times [\hat{x}_1, \ldots, \hat{x}_K]^T$
```

3.9 The PDAF Approach

The PDAF is one of the simplest and most effective approaches to target tracking [2]. The PDAF idea can be applied to the CDMA model (1) as follows. The decisions on each user can be considered as binomial random variables, with the currently-estimated probabilities for the bit from user i to be +1 or -1, P_{b_i} and $1 - P_{b_i}$, respectively. By using a Gauss-Seidel iteration, the "soft" decisions are updated sequentially on all users. From (1) we have

$$y = r_i w_i b_i + \left(\sum_{j \neq i} r_j w_j b_j + n \right) \qquad (15)$$

where r_i is the i^{th} column of $\mathbf{R}$ and w_i is the i^{th} diagonal element of $\mathbf{W}$. When updating the probability of user i to be +1 or -1, the combination of interferences from other users are considered approximately as a Gaussian random vector, with the mean and variance for user j calculated according to the current decision probabilities as $(2P_{b_j} - 1)$ and $4P_{b_j}(1 - P_{b_j})$, respectively. We consequently obtain

$$P\left(y|b_i\right) \approx N\left(r_i w_i b_i + \theta_i, \Omega_i\right) \qquad (16)$$

in which $\mathcal{N}$ refers to the standard normal probability density function (pdf), and

$$\theta_i = \sum_{j \neq i} r_j w_j (2P_{b_j} - 1)$$

$$\Omega_i = \sum_{j \neq i} 4P_{b_j}(1 - P_{b_j}) w_j^2 r_j r_j^T + \sigma^2 \mathbf{R} \qquad (17)$$

Its computational burden is approximately $\mathcal{O}(K^3)$, and as the simulation results show, its performance is nearly indistinguishable from that of the branch and bound detector. A basic form of the PDA method is shown below, detailed description can be found in [11].

PDAF Approach

```
Step 1: Sort users according to the user ordering criterion proposed for the decision feedback detector in
Step 2: Initialize $P_{b_i} = 0.5\ \forall i$. Set $b = [1, 1, \cdots, 1, 1]^T$;
Step 3: Based on the current values of $P_{b_j}$, update $P_{b_i}$, $(i \neq j)$ as:
         $P_{b_i} = P\left\{b_i = 1 | y, \{P_{b_j}\}_{j \neq i}\right\}$   (18)
Step 4: If all $P_{b_i}$ converge, proceed to the next step. Otherwise, return to Step 3
Step 5: Make decisions $\forall i$ via,
         $b_i = \begin{cases} 1 & P_{b_i} \geq 0.5 \\ -1 & P_{b_i} < 0.5 \end{cases}$   (19)
```

4 Simulation Results

In this section, we compare the performance of the algorithms described in the previous section. The probability of group detection error is computed by varying the number of users with a fixed SNR. The number of users tested are 8, 13, 16, 20, and 30. In each case, the signal energy for each user is generated such that $w_i \sim \mathcal{N}(4.5, 2)$ and $w_i \in [3, 5]\ \forall i$. The number of group detection errors are used to measure the accuracy of each algorithm based on 20000 Monte-Carlo runs. All of the simulations were run on UNIX machines. The box-constrained quadratic programming is initialized with the output from the decorrelator. For ϕ_{SD}, M_{rand} is set to 20. GDFD(3) denotes that the group size of the group detection method is set to 3, e.g., $|G_{max}| = 3$.

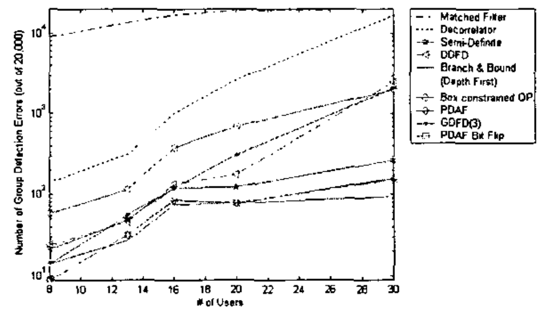

Figure 1: Number of group detection errors vs. Number of users at 15dB

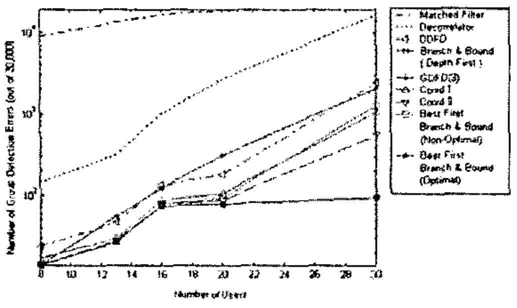

Figure 2: Number of group detection errors vs. Number of users at 15dB

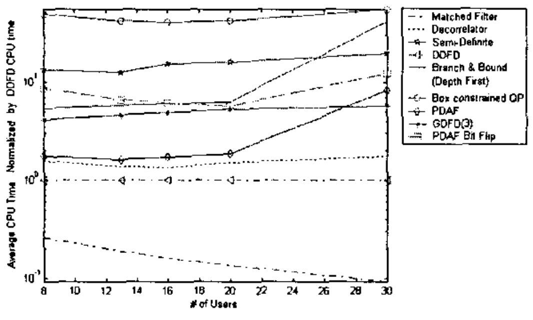

Figure 3: CPU time for various algorithms normalized by DDFD CPU time vs. Number of users at 15dB

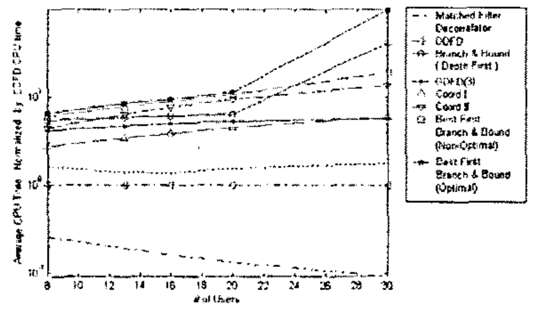

Figure 4: CPU time for various algorithms normalized by DDFD CPU time vs. Number of users at 15dB

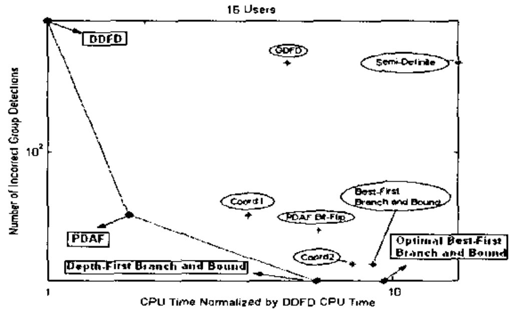

Figure 5: Number of Incorrect Decisions vs. Normalized CPU time at 15dB for 16 Users

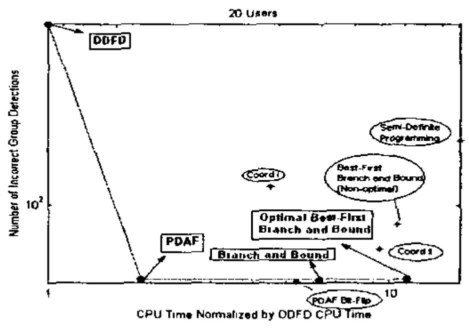

Figure 6: Number of Incorrect Decisions vs. Normalized CPU time at 15dB for 20 Users

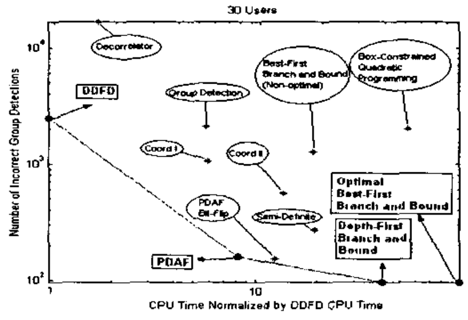

Figure 7: Number of Incorrect Decisions vs. Normalized CPU time at 15dB for 30 Users

From Figure 1, it is clear that the performance of PDAF and PDAF with bit-flips are nearly identical to that of the "depth-first" branch and bound algorithm. Moreover, the bit-flip operation improved the accuracy of PDAF significantly at a modest $O(K^2)$ computational cost. The box-constrained quadratic programming method is able to perform better than DFD when the number of users is large. The performance of the optimal version of "best-first" branch and bound algorithm is close to that of "depth-first" branch and bound, as seen on Figure 2. The CPU time of the family of branch and bound algorithms is greater by almost one to two orders of magnitude when compared to the DDFD's computation time. In comparison, the CPU time of PDAF algorithms is approximately five times that of DDFD. In Figures 5, 6 and 7, the relationship between the CPU time and the accuracy for each algorithm are shown. It is clear that DDFD, the "depth-first" branch and bound and PDAF form an "efficient frontier" for the algorithms. Using the accuracy and computations as performance meters, it is evident that the DDFD, PDAF and branch and bound algorithms dominate the remaining algorithms, viz., the matched filter, decorrelator, coordinate descent I and II, box-constrained quadratic programming method, semi-definite method and "best-first" branch and bound algorithm. This, we believe, is a nice portrayal of the trade-off between accuracy and computational complexity.

5 Conclusion

We have portrayed the trade-off between the accuracy and speed of the various approaches. From the results, we conclude that an "efficient frontier" is formed by DDFD, PDAF and branch and bound schemes; that is, all others are dominated by these algorithms. From Figures 5, 6 and 7, it is clear that an inaccurate suboptimal method such as the decorrelator cannot cross the "efficient frontier", although it is quick in making its decisions. Accurate but time-consuming methods such as the coordinate descent II, or semi-definite programming appear not to cross the frontier either. By observing these figures, it is clear that the PDAF is both time-efficient and accurate.

Acknowledgement

This work was supported by UCRF grant 444636, by ONR under contracts N00014-98-1-0465 and N00014-00-1-0101, and by NUWC under contract N66604-01-1-1125.

References

[1] B. Alidaee, B. Kochenberger, A. Ahmadian, "0-1 Quadratic programming approach for the optimal solution of two scheduling problems", *Int. Jnl. of Systems Science*, vol. 25, pp. 401-408, 1994.

[2] Y. Bar-Shalom, X. Li, *Estimation and tracking: principles, techniques and software*, Artech House, Dedham, MA, 1993. Reprinted YBS Publishing, 1998.

[3] D. Bertsekas, *"Network optimization, continuous and discrete models"*, Athena Scientific, Belmont, MA, Chap. 10, pp. 483-492, 1998.

[4] J. Krarup, P. Pruzan, "Computer aided layout design", *Mathematical Programming Study*, vol. 9 pp. 75-94, 1978.

[5] T. Coleman, Y. Li "A Reflective Newton method for minimizing a quadratic function subject to bounds on some of the variables", *SIAM Jnl. on Optimization*, vol. 6, no. 4, pp. 1040-1058, Nov. 1996.

[6] G. Gallo, P. Hammer, B. Simeone, "Quadratic knapsack problems", *Mathematical Programming*, vol. 12, pp. 132-149, 1980.

[7] C. Helmberg, R. Rendl, R. Vanderbei, H. Wolkowicz, "An interior-point method for semidefinite programming", *SIAM Jnl. on Opt.*, vol. 6, no. 2, pp. 342-361, 1996.

[8] J. Luo, K. Pattipati, P. Willett, G. Levchuk, "A class of coordinate descent algorithms for multi-user detection", *ICASSP*, Istanbul, Turkey, May 2000.

[9] J. Luo, K. Pattipati, P. Willett, G. Levchuk, "Fast optimal and sub-optimal any-time algorithms for CWMA multiuser detection based on branch and bound", submitted to *IEEE Trans. on Comm.*, July 2000.

[10] J. Luo, K. Pattipati, P. Willett, G. Levchuk, "Optimal grouping algorithm for a group decision feedback detector in synchronous code division multiaccess communications", to appear in *IEEE Trans. on Comm.*, 2001.

[11] J. Luo, K. Pattipati, P. Willett, F. Hasegawa "Near optimal multiuser detection in synchrnous CDMA using Probabilistic Data Association", to appear in *IEEE Comm. Letters*, Sep. 2001.

[12] R. Lupas, S. Verdu, "Linear multiuser detectors for synchronous code-division multiple-access channels", *IEEE Trans. on Info. Theory*, pp. 123-136, Jan. 1989.

[13] W. Ma, T. Davidson, K. Wong, Z. Luo, P. Ching, "Quasi-maximum-likelihood multiuser detection using semi-definite relaxation", Working paper, McMaster University, Hamilton, Ontario, Canada.

[14] R. McBridge, J. Yormark, "An evolutionary heuristic for quadratic 0-1 programming", *Management Science*, vol. 26, no. 3, pp. 282-296, 1980.

[15] Y. Nesterov, "Quality of semidefinite relaxation for nonconvex quadratic optimization", *CORE Discussion paper #9719*, Belgium, Mar. 1997.

[16] P. Pardalos, G. Rodgers, "Computational aspects of a branch and bound algorithm for quadratic zero-one programming", *Computing*, vol. 45, pp. 131-144, 1990.

[17] A. Phillips, J. Rosen, "A quadratic assignment formulation for the molecular conformation", *Jnl. of Global Optimization*, vol. 4, pp. 229-241, 1994.

[18] L. Vandenberghe, S. Boyd, "Semidefinite programming", *SIAM Review*, vol. 38, pp. 49-95, 1996.

[19] M. Varanasi, "Multistage detection for asynchronous code-division multiple-access communications", *IEEE Trans. on Comm.*, pp. 509-519, Apr. 1990.

[20] M. Varanasi, "Group detection for synchronous Gaussian code-division multiple-access channels", *IEEE Trans. on Info. Theory*, pp. 1083-1096, July 1995.

[21] M. Varanasi, "Decision feedback multiuser detection: a systematic approach", *IEEE Trans. on Info. Theory*, pp. 219-240, Jan. 1999.

[22] S. Verdu, *"Multi-user detection"*, Cambridge University Press, NY, 1998.