Branch and Bound Methods for a Search Problem

Alan R. Washburn
Department of Operations Research, Naval Postgraduate School, Monterey, California 93943, USA

Received 14 January 1997; revised 24 October 1997; accepted 24 October 1997

Abstract: The problem of searching for randomly moving targets such as children and submarines is known to be fundamentally difficult, but finding efficient methods for generating optimal or near optimal solutions is nonetheless an important practical problem. This paper investigates the efficiency of Branch and Bound methods, with emphasis on the tradeoff between the accuracy of the bound employed and the time required to compute it. A variety of bounds are investigated, some of which are new. In most cases the best bounds turn out to be imprecise, but very easy to compute. © 1998 John Wiley & Sons, Inc. Naval Research Logistics 45: 243–257, 1998

1. INTRODUCTION

This paper describes several branch-and-bound (B + B) methods for solving a moving-target search problem, and summarizes computational experience. B + B methods often involve a quality/cost tradeoff in calculating bounds, an aspect that is explored. The results may therefore be of some general interest. Additionally, the search problem may be of inherent interest in itself, since optimization for practical applications is on the verge of feasibility.

The search problem is of the type first analyzed by Stewart [9]: Time and space are discrete, the searcher has a location that cannot change too much from time to time, and the target moves randomly from one spatial cell to another as the search proceeds. These assumptions characterize maritime searches for hostile submarines or friendly vessels in distress, as well as terrestrial searches for lost persons. Current maritime software only rarely attempts search optimization in such situations (Wagner [13]), and current terrestrial search-and-rescue software never does, at least not in the author's experience. Part of the explanation for this lack is the need for effective methods for optimizing large problems, the subject of this paper.

Heuristic methods are often effective at solving or nearly solving problems of this type. Dell et al. [4] describe experiments involving several of these. This paper will focus exclusively on B + B methods, since these can provide bounds on the best possible solution.

2. PATH-CONSTRAINED SEARCH FOR A MOVING TARGET

The target track is $X = (X_1, \ldots, X_T)$, where $X_i \in C$ represents the position of the target at time i. C is a finite set of cells, and the single searcher as well as the target must at all times be in one of them. The positive integer T is the fixed amount of time available for search. If the searcher's track is $\sigma = (\sigma_1, \ldots, \sigma_T)$, then the probability of not detecting track X is some given function $\eta(X, \sigma)$. The searcher's object is to minimize the nondetection probability $\Sigma_X f(X)\eta(X, \sigma)$, where $f(\ \ )$ is a given probability mass function. In other words, the probability law governing the target's motion is assumed known.

If there were (say) $|C| = 9$ cells and $T = 10$ detection opportunities, there would in principle be 9^{10} possible tracks, too many to permit even evaluating the objective function, much less optimizing it. Since interesting problems can be even larger, special structures must be imposed to permit optimization. Essentially all work to date has been based on the assumptions that

$$\eta(X, \Psi) = \exp(-Z(X, \Psi)), \tag{1}$$

where

$$Z(X, \Psi) = \sum_{t=1}^{T} W(X_t, t)\Psi(X_t, t), \tag{2}$$

$\Psi(X_t, t)$ being an indicator function for the event $(X_t = \sigma_t)$; i.e., $\Psi(x, t) = 1$ if and only if $x = \sigma_t$. The Ψ and σ notations for a searcher path are equivalent; each is used in the sequel when convenient. The search effectiveness function $W(x, t)$ is assumed to be nonnegative for all $x \in C$, $1 \le t \le T$. Formulas (1) and (2) include the important instance where $W(\cdot, \cdot)$ is a constant W, in which case $Z(X, \Psi)/W$ is the number of times N that the searcher and the target occupy the same cell. Letting $QS = \exp(-W)$, $\eta(X, \Psi)$ is then $(QS)^N$. Thus QS can be identified as the probability of overlooking the target even when the correct cell is searched. The overlook probability can be made to depend on time and location through the function $W(\cdot, \cdot)$, but the independence assumption is inherent in the exponential detection function.

Let $S(\sigma_0, 0)$ be a given nonempty subset of C. Constrain the searcher's path to be in this subset at time 1, and for $1 \le t < T$ require that a searcher in cell x at time t must proceed to some cell in $S(x, t)$ at time $t + 1$. The set $S(x, t)$ is a nonempty subset of C corresponding to the ''forward neighbors'' of x. It is also useful to define a ''backward'' set, $S^*(x, t)$, to be the set of cells at time $t - 1$ from which it is possible to be in x at t, so for $x \in C$ and $1 \le t < T$, $S^*(x, t + 1) = \{y | x \in S(y, t)\}$. For $x \in C$, $y \in C$, and $1 \le t < T$, let $u(x, y, t)$ be 1 if the searcher visits x at t and y at $t + 1$, or otherwise 0; these indicator variables serve to directly trace the searcher's route from one cell to another. Then the problem of minimizing the nondetection probability can be stated as the nonlinear programming problem NLP1, with $\tau = 0$:

$$\min E(\eta(X, \Psi))$$

subject to

$$\sum_{x \in S(\sigma_\tau, \tau)} \Psi(x, \tau + 1) = 1 \quad \text{and} \quad \Psi(x, \tau + 1) = 0 \quad \text{for } x \notin S(\sigma_\tau, \tau), \tag{3}$$

$$\sum_{y \in S(x,t)} u(x, y, t) = \Psi(x, t), \quad \tau < t < T, \quad x \in C, \tag{4}$$

$$\sum_{y \in S^*(x,t+1)} u(y, x, t) = \Psi(x, t+1), \quad \tau < t < T, \quad x \in C, \tag{5}$$

$$\Psi(x, t) \quad \text{and} \quad u(x, y, t) = 0 \quad \text{or} \quad 1, \quad x \in C, \quad y \in C, \quad \tau < t < T. \tag{6}$$

Constraint (3) requires the searcher to start at some cell in $S(\sigma_\tau, \tau)$, while constraints (4)–(6) require the searcher to move from one legal cell to another. The sum in (5), for example, is ''number of times the searcher moves from some cell at time t to cell x at time $t + 1$,'' which must be $\Psi(x, t + 1)$. If $\tau > 0$, it should be understood that $\sigma_0, \ldots, \sigma_\tau$ is specified to be a legal beginning of a searcher track, with only the part after τ to be optimized; the initial cell σ_0 is included for notational convenience. The $E(\ \ )$ in the objective function denotes expected value, so that the objective function is a weighted sum of exponentials of the form (1), one term for each potential target track.

3. BRANCH AND BOUND FRAMEWORK

The following basic B + B algorithm is a slight modification of Stewart's [9, 10]. It is assumed that the bound computed in step 2 reduces to the exact nondetection probability when the searcher's entire path is specified ($\tau = T$). $K(\tau)$ is ''the set of continuations yet to be explored,'' and q^* is ''the best nondetection probability yet found.''

1. Set $\tau = 0$, initialize q^* to be any number exceeding 1, and let $\sigma_\tau = \sigma_0$.
2. Obtain a lower bound q on the nondetection probability, subject to the searcher's path following $\sigma_0, \ldots, \sigma_\tau$ up to time τ.
3. If $q < q^*$ and $\tau < T$, then Branch; i.e., let $K(\tau + 1) = S(\sigma_\tau, \tau)$, increment τ, let σ_τ be any cell in $K(\tau)$, and return to 2. Otherwise, the current path has now been fathomed.
4. If $q < q^*$ and $\tau = T$, let $q^* = q$ and save the path $\sigma_0, \ldots, \sigma_T$.
5. If $\tau = 0$, stop. The last saved path is optimal and q^* is its nondetection probability.
6. Delete σ_τ from $K(\tau)$. If $K(\tau)$ is now empty, decrement τ and return to step 5. If not, let σ_τ be any cell in $K(\tau)$ and return to step 2.

If the bound in step 2 is too loose, then no fathoming will occur. The object is to find relaxations of NLP1 that are easy to solve, but still provide tight bounds. Fortunately NLP1 has several relaxations that considerably simplify it. Three of the most important are described below.

Convex. If binary constraints (6) are replaced by simple nonnegativity requirements, NLP1 is a convex program for which Kuhn–Tucker conditions are necessary and sufficient for optimality [11].

Linear. Since $\exp(-Z) \geq 1 - Z$ for $Z \geq 0$, a lower bound can be obtained by minimizing $1 - E(Z(X, \Psi))$, which is equivalent to maximizing $\Sigma_{t=1}^{T} E(W(X_t, t)\Psi(X_t, t))$. The interpretation is that one is maximizing the mean number of detections. This is a longest route problem where the reward for visiting (x, t) is $W(x, t)$ Prob$(X_t = x)$, a relatively simple optimization problem.

A slight tightening of this bound is possible if $Z(X, \Psi)$ is always an integer multiple of some quantity $\Delta > 0$, as will be the case if $W(x, t)$ is always a multiple of Δ. In that case let $f = (1 - \exp(-\Delta))/\Delta$, which is smaller than 1, or otherwise let $f = 1$. Then $\exp(-Z(X, \Psi)) \geq 1 - fZ(X, \Psi)$. The rewards in the longest route problem can be multiplied by f, which will result in a tighter bound.

Distribution of Effort (DOE). Constraints (3)–(5) are network constraints, an exploitable structure, but a further simplification results if the searcher is permitted to occupy any cell at time t that is reachable from the last constrained cell σ_τ. The set S_t of such cells can be obtained from the recurrence $S_{t+1} = \cup_{x \in S_t} S(x, t)$, $t > \tau$, with $S_\tau = \{\sigma_\tau\}$. Then (3)–(5) can be replaced by

$$\sum_{x \in S_t} \Psi(x, t) = 1, \quad \tau < t \leq T. \tag{7}$$

Since (7) is already implied by (3)–(5), the replacement is truly a relaxation. This relaxation has the advantage that the numerous u-variables disappear from the formulation. A further relaxation could be obtained by substituting C for S_t, but there seems to be no computational advantage in doing so.

4. THE MARKOV SPECIALIZATION

The relaxations described in Section 3 cannot in themselves make B + B a practical technique as long as evaluation of the objective function still requires enumeration of all possible target tracks. If NLP1 is to be solvable as a practical matter, either the number of tracks must be constrained or some structure must be imposed that obviates the need to enumerate them. Tactical decision aids have been based on the idea that target motion can be modeled with a track population on the order of 1000, so that proceeding on the former course would be a reasonable choice. Nonetheless, all B + B computational work to date has been based on the Markov specialization of NLP1, a structural assumption that makes it unnecessary to enumerate paths. The rest of this report also concerns that specialization.

The main advantage of the Markov assumption is that it permits the operation of the FAB (Forward and Backward) algorithm. The FAB algorithm uses the two functions

$$P(\Psi, x, t) = \text{Prob}(X_t = x \ \underline{\text{and}} \text{ no detection before } t), \tag{8}$$

and

$$Q(\Psi, x, t) = \text{Prob}(\text{no detection after } t, \ \underline{\text{given}} \ X_t = x), \tag{9}$$

as well as the relation that

$$\text{ND} \equiv \text{Prob}(\text{no detection}) = \sum_x P(\Psi, x, t) \exp(-W(x, t)\Psi(x, t))Q(\Psi, x, t). \tag{10}$$

Formula (10) is valid for $t = 1, \ldots, T$ even without the Markov assumption, but it is especially useful in the Markov case because

a. $P(\Psi, x, t)$ is easily calculated given $\Psi(y, u)$ for $y \in C$ and $u < t$.
b. $Q(\Psi, x, t)$ is easily calculated given $\Psi(y, u)$ for $y \in C$ and $u > t$.
c. Neither $P(\Psi, x, t)$ nor $Q(\Psi, x, t)$ depends on $\Psi(y, t)$ for any $y \in C$.

The import of c is that $\Psi(\cdot, t)$ can be changed to increase the objective function as long as the searches before and after t remain feasible, thus permitting an iterative (FAB) algorithm for gradually decreasing the objective function. FAB requires repeated evaluations of $P(\cdot, \cdot, \cdot)$ and $Q(\cdot, \cdot, \cdot)$, so that a and b are also important. See [2, 15] for the details of these evaluations. The main use of FAB has been in computing an improving sequence of search plans for NLP1 and its various relaxations. The limiting FAB search plans might reasonably be termed ''locally optimal,'' since a certain class of small perturbations cannot improve the objective function. Brown [2] shows that the limiting FAB plan is optimal for the problem with both the DOE and convex relaxations.

The Markov specialization also permits an effective generalization of the linear bound discussed earlier. Suppose that the searcher's path up to time τ is fixed; let

$$Z_\tau = \sum_{t=1}^{\tau} W(X_t, t)\Psi(X_t, t) \quad \text{and} \quad Z_+ = \sum_{t=\tau+1}^{T} W(X_t, t)\Psi(X_t, t).$$

Then $Z(X, \Psi) = Z_\tau + Z_+$, with Z_τ representing the past and Z_+ the future. Since the target's motion is Markov, Z_τ and Z_+ are independent when X_τ is given, and therefore

$$E(\exp(-(Z_\tau + Z_+)) | X_\tau = x) = E(\exp(-Z_\tau) | X_\tau = x) E(\exp(-Z_+) | X_\tau = x). \quad (11)$$

Let

$$P^+(x, \tau) = E(\exp(-Z_\tau) | X_\tau = x) \ \text{Prob}(X_\tau = x). \quad (12)$$

$P^+(x, \tau)$ can be obtained from the FAB function $P(\Psi, x, \tau)$ by multiplying by $\exp(-W(x, \tau)\Psi(x, \tau))$; the + superscript conveys the idea that the effect of search at time τ is included. Then the nondetection probability is

$$\text{ND} = E(\exp(-Z_\tau + Z_+)) = \sum_{x \in C} P^+(x, \tau) E(\exp(-Z_+) | X_\tau = x). \quad (13)$$

But $\exp(-Z_+) \geq 1 - fZ^+$, where $f \leq 1$ is the factor introduced earlier in describing the linear relaxation. Therefore,

$$\text{ND} \geq \sum_{x \in C} P^+(x, \tau) - f \sum_{t=\tau+1}^{T} \sum_{x \in C} E(W(X_t, t)\Psi(X_t, t) | X_\tau = x) P^+(x, \tau). \quad (14)$$

But

$$E(W(X_t, t)\Psi(X_t, t) | X_\tau = x) = W(\sigma_t, t) \ \text{Prob}(X_t = \sigma_t | X_\tau = x). \quad (15)$$

Now let

$$R(\sigma_t, t) \equiv fW(\sigma_t, t) \sum_{x \in C} \text{Prob}(X_t = \sigma_t | X_\tau = x) P^+(x, \tau) \tag{16}$$

be the searcher's reward for visiting cell σ_t at time t. A lower bound on ND can now be obtained by minimizing the right-hand side of (14), which amounts to finding the search path continuation that maximizes the total reward at times $\tau + 1, \ldots, T$, a longest path optimization problem.

If the DOE and linear relaxations are combined, then σ_t should simply maximize $R(\sigma_t, t)$ subject to being in the set S_t, for $t = \tau + 1, \ldots, T$. The need for solving a longest path problem is removed, but the bound is less tight on account of the additional relaxation. A bound of this type (PROP) is tested in Section 7, along with an even simpler bound that is developed in the next section. All three of these will be referred to as linear bounds, since the linear relaxation is central.

5. AN ''ERGODIC'' BOUND

The ergodic bound introduced in this section applies only in the circumstance where the target's motion is governed by an ergodic Markov chain, and exploits the fact that such a chain must have stationary probabilities (e.g., Feller [6]). The theorem below gives a sense in which every transition brings such a chain closer to stationarity.

THEOREM 1: Consider an ergodic Markov chain on a countable state space C with transition probability matrix (P_{ij}) and stationary distribution π. Let $\nu(j, t)$ be the probability that the state is j at time t, $j \in C$. Let S be a nonempty subset of C, let $N_T = S$, and for $t < T$ let

$$N_t \equiv \{i | P_{ij} > 0 \quad \text{for some } j \in N_{t+1}\}.$$

Let $r(S, t) = \max_{j \in N_t} \nu(j, t)/\pi(j)$ for $t \leq T$. Then $r(S, t + 1) \leq r(S, t)$ for $t < T$.

PROOF: Since the chain is Markov,

$$\nu(j, t + 1) = \sum_{i \in C} P_{ij} \nu(i, t), \quad t < T, j \in C. \tag{17}$$

Since the chain is ergodic, $P_{ij} = \pi(j) q_{ji}/\pi(i)$, where (q_{ji}) is the transition matrix of the inverse Markov chain (Feller [6]). Therefore,

$$\nu(j, t + 1)/\pi(j) = \sum_{i \in C} q_{ji} \nu(i, t)/\pi(i), \quad t < T, j \in C. \tag{18}$$

If $j \in N_{t+1}$, then $P_{ij} = q_{ji} = 0$ unless $i \in N_t$, so it follows from (18) that

$$\nu(j, t + 1)/\pi(j) = \sum_{i \in N_t} q_{ji} \nu(i, t)/\pi(i), \quad t < T, j \in N_{t+1}. \tag{19}$$

Since $\Sigma_{i \in N_t} q_{ji} = 1$ for all $j \in N_{t+1}$, the right-hand side of (19) is bounded above by $r(S, t)$ regardless of j. Therefore, $r(S, t + 1) \leq r(S, t)$, as was to be shown.

COROLLARY: $r(C, t+1) \leq r(C, t)$ for all t.

PROOF: Take $S = C$.

Theorem 1 is useful in bounding the reward $R(\sigma_t, t)$ obtainable at time t by the searcher, for $\tau < t \leq T$. For this purpose, let S be S_t, the set of states feasible for the searcher at time t, and take $\nu(j, t) = \text{Prob}(X_t = j$ and no detection at time τ or before$)$. The searcher's reward at time t is then $fW(\sigma_t, t)\nu(\sigma_t, t)$. But $\nu(\sigma_t, t)$ is bounded above by $r(S_t, \tau+1)\pi(\sigma_t)$, so the total reward (16) is bounded by

$$\sum_{t=\tau+1}^{T} R(\sigma_t, t) \leq B(\sigma, \tau) \equiv f \sum_{t=\tau+1}^{T} W(\sigma_t, t) r(S_t, \tau+1)\pi(\sigma_t). \tag{20}$$

This is the ergodic bound. Evaluation of $B(\sigma, \tau)$ does not require any further Markov calculations once the stationary distribution is known.

If $\pi(j)$ and $W(j, t)$ are independent of j for $t > \tau$, then $B(\sigma, \tau)$ is independent of σ. Otherwise, the need to solve a longest path problem can be avoided by letting $r = \max_{t>\tau} r(S_t, \tau+1)$, in which case

$$B(\sigma, \tau) \leq fr \sum_{t=\tau+1}^{T} W(\sigma_t, t)\pi(\sigma_t). \tag{21}$$

Maximizing the right-hand side of (21) still involves a longest path problem, but it is now a problem that only needs to be solved once because the distribution of the state at time $\tau + 1$ is no longer involved. The inequality in (20) states that the ergodic bound $B(\sigma, \tau)$ is less tight than the linear bound. The relative advantage of the ergodic bound is in ease of computation.

6. REVIEW OF PREVIOUS COMPUTATIONAL EXPERIENCE WITH B + B

All discussions in this section concern the Markov specialization. Stewart [9] considers the linear relaxation, but finds the resulting bounds ''· · · too weak to be usefully effective · · ·.'' He is then led to the DOE relaxation, employing the FAB algorithm to ''solve'' it. He acknowledges that the resulting B + B solutions are potentially nonoptimal because FAB solutions of the DOE relaxation are themselves nonoptimal, but rejects making the additional convex relaxation (FAB solutions would then be optimal) because the resulting bounds are again weak. Stewart [10] gives computational results for a one-dimensional problem where the searcher has two options (right or left) at each time, and $T = 10$. He finds the true optimal solution in 101 out of 105 test problems.

Eagle and Yee [5] use the convex relaxation of NLP1 to obtain bounds. This relaxation has a nonlinear objective function and network constraints. It is solved by an iterative method where at each stage a linear approximation to the objective function is made. An attractive feature of this method is that each of these minimizations results in a solution feasible in NLP1, thus permitting q^* to be quickly reduced.

Martins [7] pursues the idea of using bounds that are easily evaluated, rather than tight. He uses the linear relaxation in the form of Eqs. (13)–(16), so calculating a bound takes the form of a longest path problem. The resulting procedure does more branching than the

Eagle–Yee procedure on the same test problems, but still has run times that are smaller by a factor of about 4. Dell et al. [4] reinforce the conclusion that the linear bound is computationally superior to the convex bound.

7. A COMPUTATIONAL EXPERIMENT

All of the experiments reported here are for $C = \{1, \ldots, N\}$, a one-dimensional set of cells. In boundary cells 1 and N, the target moves inward with probability D, or remains stationary with probability $1 - D$. In interior cells, the target moves right and left with probability D, or remains stationary with probability $1 - 2D$. The target's initial cell is specified, as is the cell that the searcher must examine at time 1. First the searcher examines the given cell, then each party moves to a new position, then a second search is made at time 2, etc., until finally the last search is made at time T. If the searcher's current position is $x \in C$, then the searcher's next position can be any $y \in C$ such that $|x - y| \leq 1$. The overlook probability $QS \equiv \exp(-W(x, t))$ is constant for all $x \in C$, $1 \leq t \leq T$. Four lower bounds are tested, listed below in order of computational difficulty.

1. ***ERGO2.*** The random walk chosen for the target has stationary distribution $(1/N, \ldots, 1/N)$, and $W(x, t)$ is constant, so the ergodic bound $B(\sigma)$ can be obtained from (20). $P(\Psi, x, t)$ is calculated up to time $\tau + 1$, the last time at which $P(\Psi, x, t)$ is determined by $(\sigma_1, \ldots, \sigma_\tau)$, but not beyond.
2. ***PROP.*** This is the DOE/linear bound described at the end of Section 4. The forward function $P(\Psi, x, t)$ is ''propagated'' (hence the name) for $t > \tau + 1$ using the Markov rule, making no correction for the effects of search. PROP is a tighter bound than ERGO2.
3. ***MEAN.*** This is the same bound utilized by Martins [7]. Only the linear relaxation is made, so a longest path computation is necessary. MEAN is a tighter bound than PROP.
4. ***FABC.*** This is the FAB bound for NLP1 with both the convex and DOE relaxations [14]. This is the only bound that requires FAB's backward function $Q(\cdot, \cdot, \cdot)$. Since the backward calculation of $Q(\cdot, \cdot, \cdot)$ is performed after the forward calculation of $P(\cdot, \cdot, \cdot)$, the calculations are normally applied iteratively until $P(\cdot, \cdot, \cdot)$ and/or $Q(\cdot, \cdot, \cdot)$ ''converges'' in some sense. FABC applies only a single iteration, saving time at the expense of tightness. Even so, the need for utilizing exponential and logarithmic functions makes FABC the most time-consuming computation among the four bounds.

Certain other bounds were considered but not tested. The convex bound is not tested because the MEAN bound appears to be preferable (Martins [7] and Dell et al. [4]). An ergodic bound ERGO based on the lemma to Theorem 1 is even simpler to compute than ERGO2, which is based on (20), but ERGO's bounds are crude when the searcher is far away from the target. ERGO2 is tighter than ERGO, and not much more difficult to compute. Washburn [16] describes some tests involving ERGO, and also introduces a FAB bound on the unrelaxed NLP1, but the latter bound does not appear to be competitive.

In the B + B procedure outlined in Section 3, q^* can actually be set to any feasible nondetection probability in step 1. Using a low value is, of course, best because it makes fathoming easier, provided that calculation of that value is not too time-consuming. The FAB algorithm applied to NLP1 (unrelaxed) is used here in all cases to quickly provide a

Table 1. B + B solution times in seconds for a search problem with $T = 15$ time periods, $N = 25$ cells, and where searcher and target both start in cell 13.

	$QS = .1$			$QS = .5$			$QS = .9$		
	$D = .1$	.2	.3	$D = .1$	.2	.3	$D = .1$	.2	.3
Exhaustion	97.11	97.11	97.11	97.11	97.11	97.11	97.11	97.11	97.11
ERGO2	18.67	9.50	16.10	6.26	4.89	11.69	.39	.83	1.21
PROP	20.04	9.55	15.22	6.54	4.78	11.09	.28	.50	.61
MEAN	43.01	20.27	32.51	13.89	10.16	23.95	.55	1.10	1.26
FABC	15.77	36.03	109.47	2.74	9.61	53.17	.11	.33	.27

good starting path for the searcher. The associated q^* is usually not optimal, but usually not too far off.

7.1. Basic Results

The results of a comparison on a problem with 15 time periods and 25 cells are shown in Table 1. The times shown are for a FORTRAN implementation running on a 120 MHz 486 PC. The run time for exhaustively examining every path (which, of course, depends on neither QS nor D) is shown for comparison. Some features worthy of note are:

- Run times decrease strongly with QS in all four cases, particularly for FABC.
- Although ERGO2 has an occasional victory, PROP seems to be the best of the three linear bounds.
- FABC has a strong preference for problems where D is small and QS is large. In several cases it is the best bound. There is also one case where its time exceeds that of exhaustion.
- When QS is small, the three linear bounds are fastest when $D = .2$, rather than when $D = .1$.

The first bullet above may be counterintuitive, since the target is easiest to find when QS is small, rather than large. The first look will detect the target with probability $1 - QS$, and the remaining 14 looks can at best increase the number to 1, a marginal improvement when QS is small. Finding the *best* marginal improvement, however, is a difficult problem when QS is small, as the solution times in Table 1 attest. In practice, one would look for an ε-optimal solution in such circumstances (see Section 8).

The last bullet may also be counterintuitive, since $D = 0$ corresponds to a stationary-target search problem, a relatively easy type. But the linear bounds do not exploit stationarity, and the linear approximation becomes crude in problems where the target is almost certain to be detected. As D becomes smaller, the increasing lethargy of the target makes it easier to detect, and B + B with a linear bound takes increasingly long to find an optimal solution. When $D = 0$ and $QS = .5$, PROP takes 97 s to find an optimal detection probability of .997. FABC accomplishes the same task in a fraction (.05) of a second after completely evaluating only 29 searcher paths. The optimal solution (never move from cell 13) is obvious.

Here are some details for the case in the middle of Table 1 with $QS = .5$, $D = .2$, and with both searcher and target starting in cell 13, hereafter the ''central'' case. The optimal searcher path is (13,13,13,12,13,14,15,14,13,12, 11,12,13,14,15), and the resulting detection

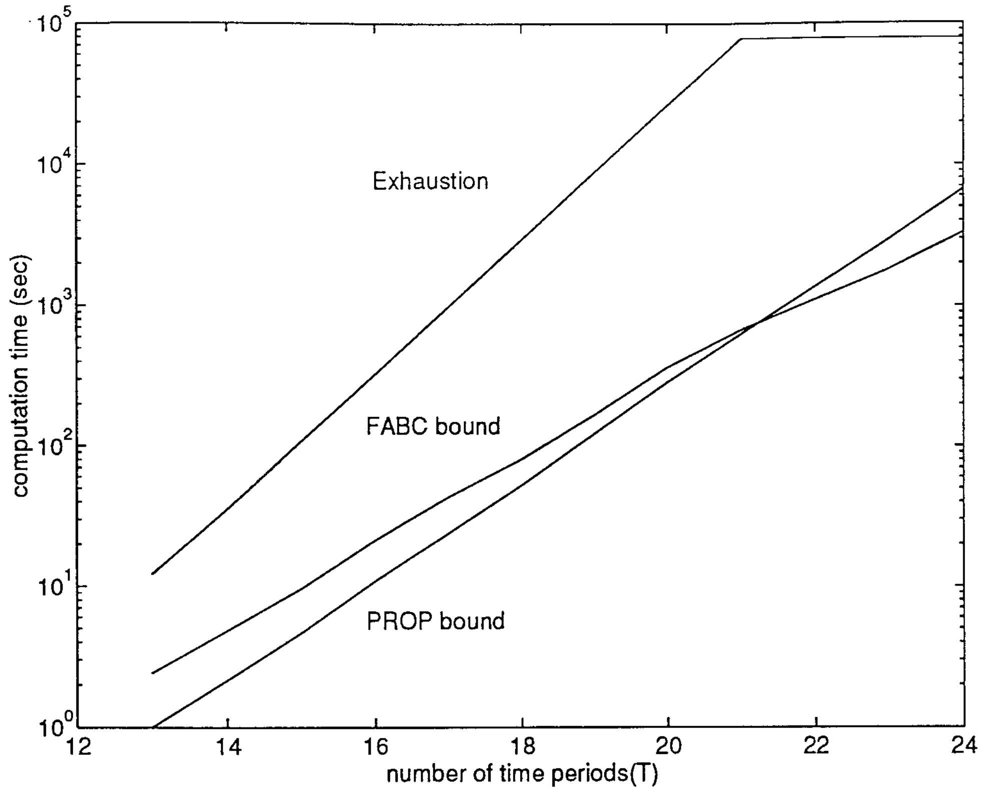

Figure 1. Exponential growth of computation time.

probability is .905594. This path is one of 1,594,321 examined by the exhaustion procedure. The numbers of bounding attempts by the four B + B procedures in coming to the same conclusion are ERGO2 (54,384), PROP (26,115), MEAN (25,977), and FABC (14,844). The fact that the number of attempts is almost the same for PROP and MEAN indicates that the DOE relaxation used by PROP has only a slight effect on bound tightness, and explains why the simpler PROP has the smaller computation time.

Figure 1 shows the computation time for three procedures on the central case, except that the number of time periods varies from 13 to 24. The computation time by exhaustion is almost exactly 97.11 $(3)^{T-15}$ s over the range tested ($13 \leq T \leq 21$). This is exponential growth on a base of 3 because all interior cells have 3 possible successors. The computation time for PROP is very close to 4.78 $(2.22)^{T-15}$ seconds. This is still exponential growth, but at least the B + B procedure reduces the base from 3 to 2.22. The other two linear bounds are not shown, but exhibit exponential growth on the same base. The growth of the FABC computation time with T is visibly not a straight line, but still it is clear that the time grows more slowly than for PROP. The FABC time is roughly 9.61 $(1.95)^{T-15}$ s. By time 24, FABC is easily the fastest of the five tested procedures.

Figure 2 shows computation times for the same three procedures, but with the QS and D parameters scaled to reflect the idea that a fixed time interval is simply being subdivided into T equal parts. The scaling formula for D is $D = .2(15/T)$, so the total variance of the target's location over T transitions remains fixed at $.2 \times 15 = .3$. The scaling formula

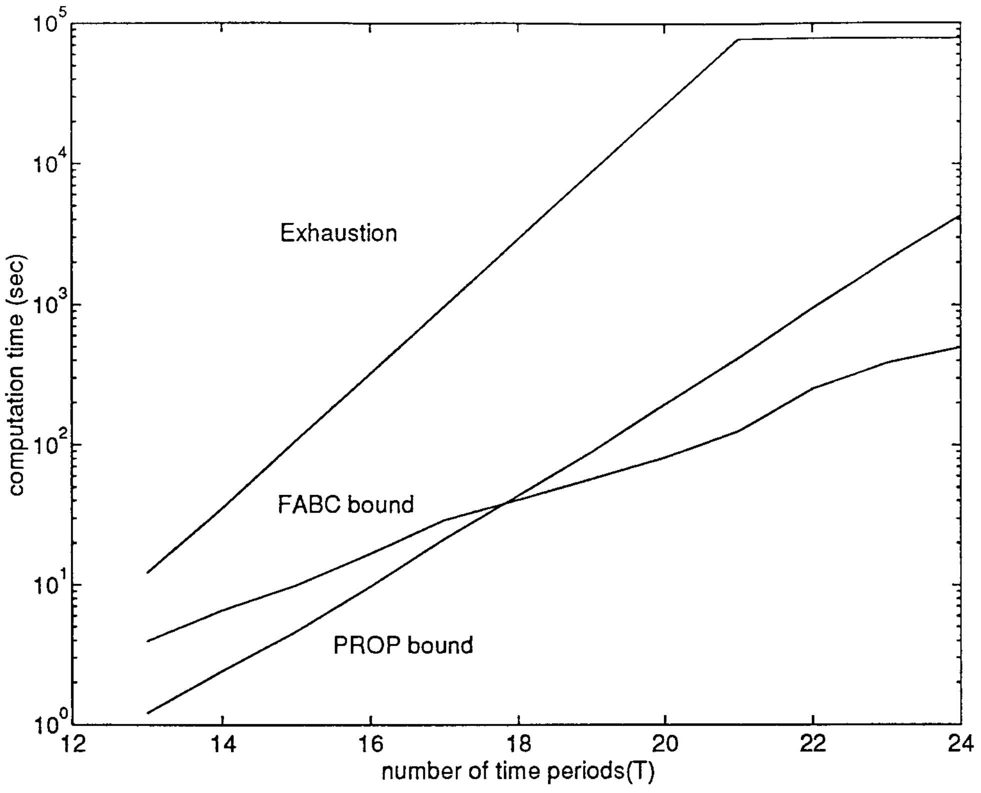

Figure 2. Growth of computation time in a scaled problem.

for QS is $QS = (.5)^{15/T}$, so the nondetection probability of T looks in the correct cell is fixed at $(.5)^{15}$ (it is, of course, unlikely that all looks will be in the cell occupied by the target—the formula simply fixes the total search effort). Both PROP and FABC benefit from the scaling in the sense of having lower computation times when T is large, particularly FABC. The base for PROP decreases from 2.22 to 2.14, while the base for FABC decreases dramatically from roughly 1.95 to roughly 1.55. It was mentioned earlier that FABC is particularly efficient on problems where QS is large and D is small. Figure 2 is further confirmation. The optimized detection probability is approximately .9 for all values of T in the scaled problem.

7.2. Hybrid Bounds

The FABC bound is so time-consuming to compute that it is tempting to try PROP or some other simpler bound first. If the simple bound succeeds in fathoming, then the FABC computations can be avoided; if not, then little has been lost. In any case, the (PROP, FABC) ''hybrid'' is certainly tighter than either bound alone, so it is reasonable to hope that the hybrid might be computationally superior to either of its components.

Of course, it is also possible that the hybrid will *not* be superior. On the central case where $QS = .5$, the time for the (PROP, FABC) hybrid is 6.65 s, larger than the time for PROP and smaller than the time for FABC (Table 1). The inclusion of FABC as a secondary

. **Table 2.** Comparison of hybrid and pure bounds

	Time (s)	Primary attempts (1000s)	Primary successes (1000s)	Secondary attempts (1000s)	Secondary successes (1000s)
PROP	147.26	668	445	—	—
FABC	159.56	192	128	—	—
(PROP, FABC)	135.34	152	57	95	44

bound makes it impossible to have a cheap failure of the primary bound, and the effect is essentially to slow down PROP. Change QS to .9 and the hybrid bound has approximately the same computation time (.33 s) as FABC. FABC is so tight in this case that PROP usually fails, and the net effect of introducing a primary bound that is easy to compute, but which usually fails, is small. Similar tests with ERGO2 and MEAN as the primary bound come to the same conclusion: hybrid bounds offer no decrease in computation time when one of the pure components is clearly superior to the other.

However, hybrid bounds can be winners. Table 2 shows the results of a test involving the (PROP, FABC) hybrid and its two pure components on the central case modified to have $QS = .7$ and $T = 20$. Now 38% of the hybrid calls result in fathoming by PROP, so that only 95,000 calls to FABC are needed, half as many as with pure FABC. In addition, 46% of the hybrid FABC calls are successful, thus reducing the number of hybrid PROP calls to a level much smaller than when PROP is employed alone. The net result is that the hybrid procedure is superior to either of its components.

And yet the hybrid procedure is only slightly superior, even on a case deliberately selected to favor it. One might hope to find another case or a different hybrid where the success rates of the components are as large as the rates of 67% for the pure procedures. This turns out to be a vain hope. Intuitively, the problem is that failure of one component is evidence that the fathoming problem is so hard that the other component will probably also fail. Formally, consider the graphical tree where each vertex is one of the paths considered by the B + B procedure (Figure 3). Terminal vertexes (‘‘leaves’’) correspond to paths that are fathomed, and other vertexes (‘‘nodes’’) branch once for each successor cell available to the searcher. If there are L leaves and $N \geq 1$ nodes, and if each node branches $B \geq 1$ times, then necessarily $(L - 1)/N = B - 1$, a proposition that can be easily proved by

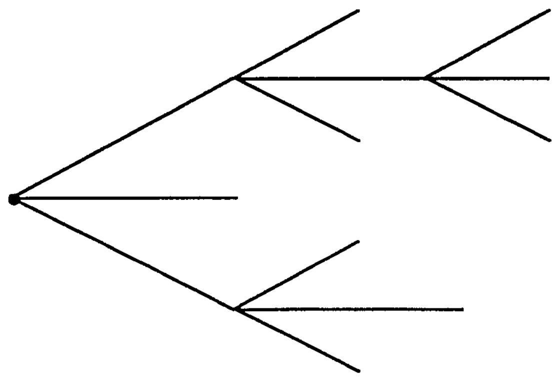

Figure 3. B + B pictured as a tree with $B = 3$, $L = 9$, and $N = 4$.

induction on N (the crucial observation is that turning any leaf into a node adds one node and $B - 1$ leaves to the tree). Now, B is 3 for all of the procedures in Table 2, since all paths are fathomed before getting to any edge cell where only two successors are feasible. Therefore, the ratio of successful attempts (L) to attempts ($L + N$) must be very nearly $\frac{2}{3}$. This is true even for the hybrid procedure in Table 2 if one divides total successes (101,000) by total hybrid attempts (152,000). If f_1 and f_2 are the fractions of successful component attempts in a hybrid procedure, then necessarily $f_1 + (1 - f_1)f_2 = \frac{2}{3}$, so certainly f_1 and f_2 cannot both be $\frac{2}{3}$. In this sense there is a fundamental limit on the efficiency of any hybrid procedure.

In spite of these discouraging theoretical and computational results concerning hybrid bounds, there is still something to be said for making a slow bound like FABC be secondary to a fast primary like PROP in robust software that must be prepared for a variety of problems. Little will be lost in problems where the secondary would be better employed alone, and on other problems the hybrid may be considerably faster than the lone secondary. There is also something to be said for employing the slow secondary calculation only when the primary bound ‘‘almost’’ succeeds in fathoming, although the gains from employing this device are not dramatic [16].

8. SUMMARY AND PROGNOSIS

The search problem considered here happens to be one where a variety of bounds are available, some emphasizing tightness and some emphasizing speed. To an extent that has strained the author’s intuition, victory in the sense of total computation time usually has gone to bounds that emphasize speed. The MEAN bound seems almost hopelessly crude at first sight, since it approximates the probability of at least one detection by the mean number of detections, but still MEAN has performed very well in previous computational tests. The PROP bound is even less tight than MEAN, since the searcher’s path is not sufficiently constrained in PROP, but PROP outperforms MEAN by about a factor of 2 in the tests reported here. Even ERGO2, which achieves a further increase in speed by disposing of much of the detail in the motion model, is still competitive within the class of linear bounds.

Furthermore, the FABC bound tested here is actually the fastest member of a class, since the FAB algorithm can be iterated as many times as desired in attempting to achieve tightness. The best number of iterations turns out to be 1, in spite of the implication that the $Q(\cdot, \cdot, \cdot)$ function must be based on a previously considered path, rather than the current one. Again, tightness is sacrificed for speed.

There is a limit to this preference for speed, of course. Per attempt, the fastest lower bound on nondetection probability is simply 0, which requires no computation but will never fathom a path that is not complete. This bound corresponds to exhaustion, a procedure that becomes increasingly unattractive as the size of the problem increases. Even so, speed is an important property of a bound. Most feasible solutions are so bad that it is easy to prove them nonoptimal, and it is important for a B + B procedure not to spend a lot of computational time doing so.

The potential attractiveness of sacrificing tightness for speed may have some general application, but there remains the specific question of how B + B should be applied to search problems where both target and searcher move as the search proceeds. To address this practical question, it is first necessary to distinguish between terrestrial and maritime searches.

Search theory is one of the original Operations Research techniques, born in WWII on account of the importance of locating maritime (particularly submarine) targets [1]. The continued importance of maritime targets, as well as the clear organizational responsibility for finding them, has by now led the U.S. Navy and the U.S. Coast Guard to develop several tactical decision aids for the search function, including CASP [8], NODESTAR [12], and others. These computer programs require the user to identify the search mechanism (eyeball, sonar, $\cdots$) and the nature of target motion, but parameters such as QS and D in the foregoing are computed, rather than input. As a result, the designer is free to utilize a fine subdivision of space and time, if he chooses, without imposing a tremendous input burden on the user. If the subdivision is fine, then there will be many opportunities for detection, all with a large nondetection probability, and it is reasonable to expect the FAB algorithm to play an important role in optimization. In particular, FABC, or possibly a hybrid where FABC is secondary to one of the Linear bounds, is a natural candidate for a bounding mechanism in a B + B algorithm (recall Figure 2).

It is not meant to imply that time *should* be finely subdivided merely because the capability is there. The analyst's desire to have searches in neighboring time intervals be independent of each other may mitigate an otherwise strong desire for a fine subdivision. Indeed, the best subdivision of time is probably the central question in planning (maritime) search software. Still, to the extent that time is finely subdivided, some version of the FAB algorithm is indicated as an optimization technique.

Terrestrial search for lost humans shares with maritime search the feature that the target moves while search proceeds, but there are also some strong differences. Terrain plays a vital role in both target motion and detection, and diurnal effects play a stronger role than in the maritime case. Most important, at least in the United States, is that there is no national organization with the responsibility for carrying out these searches. As a result, the available software is relatively primitive. CASIE 3 [3] is typical in requiring that detection probabilities be directly input by the user, one for each cell—the option of computing them is not available because there is no built-in sensor model. If only to minimize the number of such inputs required, the user is likely to subdivide time into large periods and estimate correspondingly large detection probabilities. There may still be nontrivial optimization problems; indeed, CASIE 3 already includes a single-period optimization capability. B + B could be useful in solving an optimal searcher path problem, but the indicated bounding technique would be one of the linear bounds, rather than FABC, since FABC performs poorly on problems where nondetection probabilities are small (recall Table 1).

But which linear bound is best? MEAN is the tightest of the three, but nonetheless loses to ERGO2 and PROP in the calculations reported here because of the computational cost of longest path calculations. To make MEAN look good, let the target have a predictable zigzag motion, rather than a random walk: PROP would track the target even if doing so were not a feasible searcher path, resulting in a very loose bound when compared to MEAN. To make ERGO2 look good, adopt a stationary Markov motion model where all transition probabilities are nonzero, thus slowing down PROP and MEAN without affecting ERGO2. Still, PROP usually wins the race in the computations reported here, and its conceptual simplicity is appealing. The best choice must evidently depend on circumstances, but in problems where the target's motion is roughly a random walk, the author's first choice would be PROP.

Finally, as a last point in defense of the idea of using B + B on problems of this type, it should not be forgotten that the basic B + B procedure of Section 3 can be easily adapted to find an ε-optimal solution by simply replacing q by $q + \varepsilon$ in step 3. The reward for being willing to accept a nearly optimal solution can be a dramatic decrease in computation time [16].

REFERENCES

[1] Benkoski, S.J., Monticino, M.G., and Weisinger, J.R., ''A Survey of the Search Theory Literature,'' *Naval Research Logistics,* **38,** 469–494 (1991).
[2] Brown, S.S., ''Optimal Search for a Moving Target in Discrete Time and Space,'' *Operations Research,* **28,** 1275–1289 (1980).
[3] CASIE 3, distributed by NASAR (P.O. Box 3709, Fairfax, VA 22038, 703-352-1349) and ERI (4537 Foxhall Dr. NE, Olympia, WA 98506, 360-491-7785).
[4] Dell, R.F., Eagle, J.N., Martins, G.H.A., and Santos, A.G., ''Using Multiple Searchers in Constrained-Path, Moving-Target Search Problems,'' *Naval Research Logistics,* **43,** 463–480 (1996).
[5] Eagle, J.N., and Yee, J.R., ''An Optimal Branch-and-Bound Procedure for the Constrained Path, Moving Target Search Problem,'' *Operations Research,* **38,** 110–114 (1990).
[6] Feller, W., *An Introduction to Probability Theory and Its Applications,* Volume 1, Edition 2, Wiley, New York, 1957, Chap. 15.
[7] Martins, G., ''A New Branch-and-Bound Procedure for Computing Optimal Search Paths,'' Master's Thesis, Naval Postgraduate School, 1993.
[8] Richardson, H., and Dicenza, J., ''The United States Coast Guard Computer-Assisted Search Planning System (CASP),'' *Naval Research Logistics,* **27,** 659–680 (1980).
[9] Stewart, T.J., ''Search for a Moving Target When Search Motion is Restricted,'' *Computers and Operation Research,* **6,** 129–140 (1979).
[10] Stewart, T.J., ''Experience With a Branch-and-Bound Method for Constrained Searcher Motion,'' in B. Haley and L. Stone (Eds.), *Search Theory and Applications,* Plenum, New York, 1980, pp. 247–253.
[11] Stone, L.D., ''Necessary and Sufficient Conditions for Optimal Search Plans for Moving Targets,'' *Mathematics of Operations Research,* **4,** 431–440 (1979).
[12] Stone, L., and Corwin, T., ''NODESTAR: A Nonlinear, Discrete, Multiple-target, Correlator-tracker: Part 1,'' *U.S. Navy Journal of Underwater Acoustics,* **45,** 525–540, (1995).
[13] Wagner, D.H., ''Naval Tactical Decision Aids,'' Military Operations Research Lecture Notes, NPSOR-90-01, Naval Postgraduate School, Monterey, CA, 1989.
[14] Washburn, A.R., ''An Upper Bound Useful in Optimizing Search for a Moving Target,'' *Operations Research,* **29,** 1227–1230 (1981).
[15] Washburn, A.R., ''Search for a Moving Target: The FAB Algorithm,'' *Operations Research,* **31,** 739–751 (1983).
[16] Washburn, A., ''Branch and Bound Methods for Search Problems,'' NPSOR-95-003, Naval Postgraduate School, Monterey, CA, 1995.