Three Algorithms for a Class of Multidimensional Assignment Problems

A. B. Poore and N. Rijavec

Department of Mathematics, Colorado State University, Fort Collins, U.S.A.

The assignment problem of matching the elements of two sets at some cost or to some benefit is well known and can be solved in polynomial time. However, many applications, particularly those in remote sensing and computer vision, require matching elements from more than two sets at some cost. Such problems are called multidimensional assignment problems and are known to be NP-hard. For time-critical applications and nontrivial multidimensional assignment problems, fast near-optimal algorithms are the only alternative. This paper compares three such algorithms: greedy, limited branch and bound, and Lagrangian relaxation.

Keywords: Multitarget tracking, multi-dimensional assignment problems, data association, Lagrangian relaxation, Greedy, Branch, Bound

1. Introduction

The central problem in any multitarget/multisensor surveillance system is the data association problem of partitioning the observations into tracks and false alarms [Bar-Shalom (1978), Bar-Shalom and Fortmann (1989), Blackman (1992), Morefield (1977), Reid (1979)]. Current methods for multitarget tracking generally fall into two categories: sequential and deferred logic. Methods for the former include nearest neighbor, one-to-one or few-to-one assignments, and all-to-one assignments as in the joint probabilistic data association (JPDA) [Bar-Shalom and Fortmann (1989)]. For track maintenance, the nearest neighbor method is valid in the absence of clutter and when there is no track contention, i.e., when there is no chance of misassociation. Problems involving one-to-one or few-to-one assignments are generally formulated as (two dimensional) assignment or multi-assignment problems for which there are some excellent optimal or ε-optimal algorithms [Jonker and Volgenant (1987), Bertsekas and Castanon (1991)]. This methodology is real-time but can result in a large number of partial and incorrect assignments, particularly in dense or high contention scenarios, and thus incorrect track identification. The difficulty is that decisions, once made, are irrevocable, so that there is no mechanism to correct misassociations. The use of all observations in a scan (e.g., JPDA) [Bar-Shalom and Fortmann (1989)] to update a track moderates the misassociation problem and has been successful for tracking a few targets in dense clutter.

Deferred logic techniques consider several data sets or scans of data all at once in making data association decisions. At one extreme is batch processing in which all observations (from all time) are processed together, but this is computationally too intensive for real-time applications. The other extreme is sequential processing. Deferred logic methods between these two extremes are of primary interest in this work. The principal deferred logic method used to track large numbers of targets in low to moderate clutter is called multiple hypothesis tracking (MHT) in which one builds a tree of possibilities, assigns a likelihood score based on Bayesian estimation, develops an intricate pruning logic, and then solves the data association problem by explicit enumeration schemes. The use of these enumeration schemes to solve this

NP-hard combinatorial optimization problem in real-time is inevitably faulty in dense scenarios, since the time required to solve the problem optimally can grow exponentially in the size of the problem.

Another important aspect in surveillance systems is the growing use of multisensor data fusion [Bar-Shalom and Fortmann (1989), Bar-Shalom (1990), Deb, Pattipati, and Bar-Shalom (1992)] in which one associates reports from multiple sensors together. Once matched, this more varied information has the potential to greatly enhance target identification and state estimation. The central problem is again that of data association and the principal method employed is multiple hypothesis tracking.

The central problem in these multitarget and multisensor tracking problems is an NP-hard combinatorial optimization that is formulated here as a multidimensional assignment problem. These problems have noisy objective functions due to sensor and model noise, and must be solved in real-time to the noise level of the problem. Greedy algorithms are fast but do not produce solutions of sufficient quality. The only known method for solving the problem optimally is branch and bound, but this method is much too slow for real-time use. We have developed [Poore and Rijavec (1993), Rijavec (1992)] a class of Lagrangian relaxation techniques which solve the problem to the noise level in real-time. Thus the objective in this work is to demonstrate the effectiveness and robustness these algorithms. The performance of the Lagrangian relaxation algorithm is compared to the performance of a greedy algorithm and the relaxation-based branch and bound technique. The model problem used for this demonstration is that multiple targets traveling with constant acceleration in two dimensional space.

This paper is organized as follows: The combinatorial optimization problem governing these data association problems and a simple physical description of the problem are given first. Then overviews of the Lagrangian relaxation algorithm, the greedy algorithm, and the branch and bound procedure are presented, followed by a section on a selection of results from extensive numerical studies.

2. The Assignment Problem

The combinatorial optimization problem that governs the data association problem can be stated mathematically as

Minimize

$$\sum_{i_1=0}^{M_1} \cdots \sum_{i_N=0}^{M_N} c_{i_1\cdots i_N} z_{i_1\cdots i_N}$$

Subject To:

$$\begin{aligned}
&\sum_{i_2=0}^{M_2} \cdots \sum_{i_N=0}^{M_N} z_{i_1\cdots i_N} = 1, \quad i_1 = 1,\ldots,M_1,\\
&\sum_{i_1=0}^{M_1} \cdots \sum_{i_{k-1}=0}^{M_{k-1}} \sum_{i_{k+1}=0}^{M_{k+1}} \cdots \sum_{i_N=0}^{M_N} z_{i_1\cdots i_N} = 1, \qquad (1)\\
&\qquad \text{for } i_k = 1,\ldots,M_k \text{ and } k = 2,\ldots,N-1,\\
&\sum_{i_1=0}^{M_1} \cdots \sum_{i_{N-1}=0}^{M_{N-1}} z_{i_1\cdots i_N} = 1, \quad i_N = 1,\ldots,M_N\\
&\qquad z_{i_1\cdots i_N} \in \{0,1\} \text{ for all } i_1,\ldots,i_N,
\end{aligned}$$

where $c_{0\cdots 0}$ is arbitrarily defined to be zero and is included for notational convenience. The zero index is used for representation purposes such as missing data, tracks that initiate after the first scan, and track terminations. Each cost coefficient is the negative log of a likelihood ratio, and cost coefficients with exactly one nonzero index are zero due to a normalization of this ratio. Here is a simple description of a simple physical problem that gives rise to this problem.

In tracking, a common surveillance problem is to estimate the past, current, or future state of a collection of objects (e.g., airplanes) moving in three dimensional space from a sequence of measurements made of the surveillance region by one or more sensors. The objects will be called targets. The dynamics of these targets are generally modeled from physical laws of motion, but there may noise in the dynamics and certain parameters of the motion may be unknown. (The dynamics are often modeled as stochastic differential equations.) At time $t = 0$ a sensor (or sensors) is turned on to observe the region. In an ideal situation measurements are taken at a finite sequence of times $\{t_k\}_{k=0}^{n}$ where $0 = t_0 < t_1 < \cdots < t_n$. (Due to the finite amount of time required for a sensor to sweep the surveillance region, measurements are generally made asynchronously, i.e., not at the same time, so that a time tag is associated with each measurement.) At each time t_k the sensor produces a sequence of measurements $Z(k) = \{z^k_{i_k}\}^{M_k}_{i_k=1}$ where each $z^k_{i_k}$ is a vector of noise contaminated measurements. The actual type of measurement varies with the sensor. For example, a two dimensional radar measures range and azimuth of each potential target ($z^k_{i_k} = (r^k_{i_k}, \theta^k_{i_k})^T$), a three dimensional radar that measures range, azimuth, and elevation ($z^k_{i_k} = (r^k_{i_k}, \theta^k_{i_k}, \phi^k_{i_k})^T$), a three dimensional radar with Doppler measures these and the time derivative of range ($z^k_{i_k} = (r^k_{i_k}, \theta^k_{i_k}, \phi^k_{i_k}, \frac{dr^k_{i_k}}{dt})^T$), and a two dimensional passive sensor measures the azimuth and elevation angle ($z^k_{i_k} = (\theta^k_{i_k}, \phi^k_{i_k})$). Some of the measurements may be false, and the number of targets and which measurement emanates from which target are not known a priori. Given this description, we finally note that the variable $z_{i_1 \cdots i_n} = 1$ means that the measurements $\{z^1_{i_1}, \ldots, z^n_{i_n}\}$ belong to a particular target.

The problems then are to determine the number of targets, which measurements go with which targets and which are false (i.e., the data association problem), and to estimate the state of each target given a sequence of measurements that emanate from that target.

3. Lagrangian Relaxation Algorithm

Having described the N-dimensional assignment problem (1), we now turn to a description of the Lagrangian relaxation algorithm. The description is organized in three subsections. Since the scheme is multilevel, the description of the procedure in the first subsection starts with an n-dimensional problem, relaxes this problem to an $(n-1)$-dimensional one via a Lagrangian relaxation, and concludes with a technique to recover a feasible solution to the n-dimensional problem. The second subsection describes the nonsmooth optimization algorithm that is the crucial part of the relaxation procedure, while the third subsection gives the summary of the algorithm.

3.1. The Lagrangian relaxed problem and the recovery procedure.

Let N be an integer such that $N \geq 3$ and let $n \in \{3, \ldots, N\}$. The n-dimensional assignment problem investigated in this work is

Minimize

$$v_n(z) = \sum_{i_1=0}^{M_1} \cdots \sum_{i_n=0}^{M_n} c^n_{i_1 \cdots i_n} z^n_{i_1 \cdots i_n}$$

Subject To:

$$\begin{aligned}
&\sum_{i_2=0}^{M_2} \cdots \sum_{i_n=0}^{M_n} z^n_{i_1 \cdots i_n} = 1, \quad i_1 = 1, \ldots, M_1, \\
&\sum_{i_1=0}^{M_1} \cdots \sum_{i_{k-1}=0}^{M_{k-1}} \sum_{i_{k+1}=0}^{M_{k+1}} \cdots \sum_{i_n=0}^{M_n} z^n_{i_1 \cdots i_n} = 1, \qquad (2) \\
&\quad \text{for } i_k = 1, \ldots, M_k \text{ and } k = 2, \ldots, n-1, \\
&\sum_{i_1=0}^{M_1} \cdots \sum_{i_{n-1}=0}^{M_{n-1}} z^n_{i_1 \cdots i_n} = 1, \quad i_n = 1, \ldots, M_n \\
&\quad z^n_{i_1 \cdots i_n} \in \{0, 1\} \text{ for all } i_1, \ldots, i_n.
\end{aligned}$$

To ensure that a feasible solution of (2) always exists, all variables with exactly one nonzero index (i.e., variables of the form $z^n_{0 \cdots 0 i_k 0 \cdots 0}$ for $i_k \neq 0$) will not be preassigned and the corresponding cost coefficients are assumed to be well-defined. In the first subsection, an optimal solution of the Lagrangian relaxed problem for (2) will be shown to be obtained from a lower dimensional assignment problem. The second subsection develops a recovery procedure.

The Lagrangian Relaxed Assignment Problem. The n-dimensional assignment problem (2) has n sets of constraints. A $(M_k + 1)$-dimensional multiplier vector associated with the k-th constraint set will be denoted by $u^k = (u^k_0, u^k_1, \ldots, u^k_{M_k})^T$ with $u^k_0 = 0$ and $k = 1, \ldots, n$. The n-dimensional assignment problem (2) is relaxed to a $(n-1)$-dimensional assignment problem by incorporating one of the n sets of constraints into the objective function. Although any constraint set can be relaxed, the relaxation of (2) will be based on the relaxation of the last set of constraints. The *relaxed problem* is

$$\begin{aligned}\Phi_n(u^n) &\equiv \text{Minimize} \quad \phi_n(z^n, u^n)\\ &\equiv \sum_{i_1=0}^{M_1}\cdots\sum_{i_n=0}^{M_n} c^n_{i_1\cdots i_n} z^n_{i_1\cdots i_n}\\ &+\sum_{i_n=0}^{M_n} u^n_{i_n}\Big[\sum_{i_1=0}^{M_1}\cdots\sum_{i_{n-1}=0}^{M_{n-1}} z^n_{i_1\cdots i_n} - 1\Big]\end{aligned}$$

Subject To:

$$\begin{aligned}&\sum_{i_2=0}^{M_2}\cdots\sum_{i_n=0}^{M_n} z^n_{i_1\cdots i_n} = 1,\ i_1 = 1,\ldots,M_1, \qquad (3)\\ &\sum_{i_1=0}^{M_1}\cdots\sum_{i_{k-1}=0}^{M_{k-1}}\sum_{i_{k+1}=0}^{M_{k+1}}\cdots\sum_{i_n=0}^{M_n} z^n_{i_1\cdots i_n} = 1,\\ &\quad \text{for } i_k = 1,\ldots,M_k \text{ and } k = 2,\ldots,n-1,\\ &\quad z^n_{i_1\cdots i_n} \in \{0,1\} \text{ for all } i_1,\ldots,i_n.\end{aligned}$$

An optimal (or suboptimal) solution of (3) can be constructed from that of an $(n-1)$-dimensional assignment problem. To show this, define for each $(i_1,\ldots,i_{n-1})$ an index $j_n = j_n(i_1,\ldots,i_{n-1})$ and a new cost function $c^{n-1}_{i_1\cdots i_{n-1}}$ by

$$\begin{aligned}&j_n(i_1,\ldots,i_{n-1})\\ &\quad = \arg\min\{c^n_{i_1\cdots i_{n-1}i_n}+u^n_{i_n} \mid i_n=0,1,\ldots,M_n\}\\ &c^{n-1}_{i_1\cdots i_{n-1}} = c^n_{i_1\cdots i_{n-1}j_n} + u^n_{j_n} \qquad (4)\\ &\quad \text{for}(i_1,\ldots,i_{n-1}) \neq (0,\ldots,0)\\ &c^{n-1}_{0\cdots 0} = \sum_{i_n=0}^{M_n} \min\{0, c^n_{0\cdots 0 i_n} + u^n_{i_n}\}\end{aligned}$$

(If j_n is not unique, choose the smallest such j_n, so that j_n is uniquely defined.) Using the cost coefficients defined in this way, the following $(n-1)$-dimensional assignment problem is obtained:

$$\begin{aligned}\hat{\Phi}_n(u^n) &= \text{Minimize} \quad \hat{\phi}_n(z^{n-1}, u^n)\\ &\equiv v_{n-1}(z^{n-1})\\ &\equiv \sum_{i_1=0}^{M_1}\cdots\sum_{i_{n-1}=0}^{M_{n-1}} c^{n-1}_{i_1\cdots i_{n-1}} z^{n-1}_{i_1\cdots i_{n-1}}\end{aligned}$$

Subject To:

$$\begin{aligned}&\sum_{i_2=0}^{M_2}\cdots\sum_{i_{n-1}=0}^{M_{n-1}} z^{n-1}_{i_1\cdots i_{n-1}} = 1,\ i_1 = 1,\ldots,M_1, \qquad (5)\\ &\sum_{i_1=0}^{M_1}\cdots\sum_{i_{k-1}=0}^{M_{k-1}}\sum_{i_{k+1}=0}^{M_{k+1}}\cdots\sum_{i_{n-1}=0}^{M_{n-1}} z^{n-1}_{i_1\cdots i_{n-1}} = 1,\\ &\quad \text{for } i_k = 1,\ldots,M_k \text{ and } k = 2,\ldots,n-2,\\ &\sum_{i_1=0}^{M_1}\cdots\sum_{i_{n-2}=0}^{M_{n-2}} z^{n-1}_{i_1\cdots i_{n-1}} = 1,\\ &\quad i_{n-1} = 1,\ldots,M_{n-1},\\ &\quad z^{n-1}_{i_1\cdots i_{n-1}} \in \{0,1\} \text{ for all } i_1,\ldots,i_{n-1}.\end{aligned}$$

The next objective is to show that a feasible solution of (5) produces a corresponding feasible solution of (3) that preserves optimality.

Theorem 1. *Let w^{n-1} be a feasible solution to problem (5) and define w^n by*

$$\begin{aligned}w^n_{i_1\cdots i_n} &= w^{n-1}_{i_1\cdots i_{n-1}} && \textit{if } i_n = j_n(i_1,\ldots,i_{n-1})\\ & && \textit{and } (i_1,\ldots,i_{n-1}) \neq (0,\ldots,0)\\ w^n_{i_1\cdots i_n} &= 0 && \textit{if } i_n \neq j_n(i_1,\ldots,i_{n-1})\\ & && \textit{and } (i_1,\ldots,i_{n-1}) \neq (0,\ldots,0)\\ w^n_{0\cdots 0 i_n} &= 1 && \textit{if } c^n_{0\cdots 0 i_n} + u^n_{i_n} < 0\\ w^n_{0\cdots 0 i_n} &= 0 && \textit{if } c^n_{0\cdots 0 i_n} + u^n_{i_n} \geq 0\end{aligned} \qquad (6)$$

Then w^n is a feasible solution of the Lagrangian relaxed problem (3) and $\phi_n(w^n, u^n) = \hat{\phi}_n(w^{n-1}, u^n) - \sum_{i_n=0}^{M_n} u^n_{i_n}$. If, in addition, w^{n-1} is optimal for (5), then w^n is an optimal solution of (3) and $\Phi_n(u^n) = \hat{\Phi}_n(u^n) - \sum_{i_n=0}^{M_n} u^n_{i_n}$.

Problems (2) and (5) differ only in the dimension (n in (2) and $n-1$ in (5)). One can thus construct $\Phi_{n-1}, \Phi_{n-2}, \ldots, \Phi_3$. If $n = N$, the cost coefficients are the original costs. If $n = N-1$, cost coefficients c^{N-1} are functions of the original costs and the multipliers u^N. Since cost coefficients c^{n-1} are computed recursively from the cost coefficients c^n and the multipliers u^n for $n < N$, cost coefficients c^n are functions of the original costs c^N and the multiplier vectors $u^{n+1}, \ldots, u^N$. Thus, for $n < N$ the relaxed objective function Φ_n will be written as

$$\Phi_n(u^n) = \Phi_n(u^n \; ; \; u^{n+1}, \ldots, u^N, c^N)$$

where $u^{n+1}, \ldots, u^N$ are treated as parameters in the formulation of the problem. Of course., $\Phi_N(u^N) = \Phi_N(u^N; c^N)$.

The recovery procedure. The next objective is to explain a *recovery procedure,* i.e., given a feasible (optimal or suboptimal) solution w^{n-1} of (5) (or w^n of (3) constructed via Theorem 1), generate a feasible solution z^n of (2) which is close to w^{n-1} in a sense to be specified. *We first assume that no variables in (2) are preassigned to zero; this assumption will be removed shortly.* The difficulty with the solution w^n is that it need not satisfy the last or relaxed set of constraints in (2). The recovery procedure described here is designed to preserve the 0-1 character of the solution w^{n-1} of (5) as far as possible: If $w^{n-1}_{i_1 \cdots i_{n-1}} = 1$ and at least one $i_l \neq 0$, the corresponding feasible solution z^n of (2) is constructed so that $z^n_{i_1 \cdots i_{n-1} i_n} = 1$ for some $i_n = 0, 1, \ldots, M_n$. By this reasoning, variables of the form $z^n_{0 \cdots 0 i_n}$ can be assigned a value of one in the recovery problem only if $w^{n-1}_{0 \cdots 0} = 1$. However, variables $z^n_{0 \cdots 0 i_n}$ will be treated differently in the recovery procedure in that they can be assigned 0 or 1 independent of the value $w^{n-1}_{0 \cdots 0}$. This increases the feasible set of the recovery problem, leading to a potentially better solution.

Let $\{(i^k_1, i^k_2, \ldots, i^k_{n-1})\}^{M_0}_{k=1}$ be an enumeration of indices of w^{n-1} (or the first $n-1$ indices of w^n constructed in Theorem 1) such that $w^{n-1}_{i^k_1 i^k_2 \cdots i^k_{n-1}} = 1$ and $(i^k_1, i^k_2, \ldots, i^k_{n-1}) \neq (0, \ldots, 0)$. Set $(i^0_1, \ldots, i^0_{n-1}) = (0, \ldots, 0)$ for $k = 0$ and define

$$h_{kl} = c^n_{i^k_1 i^k_2 \cdots i^k_{n-1} l} \quad \text{for} \quad l = 0, \ldots, M_n \quad \text{and} \quad k = 0, \ldots, M_0. \tag{7}$$

Let Y denote the solution of the two dimensional assignment problem

$$\text{Minimize} \quad \sum_{k=0}^{M_0} \sum_{l=0}^{M_n} h_{kl} y_{kl}$$

$$\text{Subject To :}$$

$$\sum_{l=0}^{M_n} y_{kl} = 1, \quad k = 1, \ldots, M_0$$

$$\sum_{k=0}^{M_0} y_{kl} = 1, \quad l = 1, \ldots, M_n$$

$$y_{kl} \in \{0, 1\}, \quad k = 0, \ldots, M_0, \quad l = 0, \ldots, M_n \, . \tag{8}$$

The recovered feasible solution z^n of (2) corresponding to the multiplier u^n is then defined by

$$z^n_{i_1 \cdots i_n} = \begin{cases} 1 & \text{if } (i_1, ..., i_{n-1}) = (i^k_1, ..., i^k_{n-1}) \\ & \quad \text{for some } k = 0, \ldots, M_0 \\ & \quad \text{and } Y_{k i_n} = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

This recovery procedure is valid as long as all cost coefficients c^n are defined and all zero one variables in z^n are free to be assigned. Modifications are necessary for sparse problems; however, the recovery procedure is still valid if the cost coefficients $c^n_{i^k_1 i^k_2 \cdots i^k_{n-1} l}$ are defined and the zero-one variables $z^n_{i^k_1 i^k_2 \cdots i^k_{n-1} l}$ are free to be assigned to zero or one for $k = 0, \ldots, M_0$ and $l = 0, \ldots, M_n$; otherwise, either the definition of the cost coefficient h_{kl} in (7) may not be valid or (8) may not have a feasible solution. We next present a method for dealing with this difficulty.

Observe that if $z^n_{i^k_1 i^k_2 \cdots i^k_{n-1} l}$ is preassigned to zero, then y_{kl} is also preassigned to zero. If (7) is not a valid definition for h_{kl}, set $h_{kl} = \infty$ and preassign the corresponding variable $y_{kl} = 0$ in (8). Now note that a feasible solution of (8) exists if h_{k0} ($k = 0, \ldots, M_0$) and h_{0l} ($l = 0, \ldots, M_n$) are defined and the corresponding zero-one variables y_{k0} and y_{0l} are free to be assigned. By an earlier assumption, the variables $z^n_{0 \cdots 0 i_n}$ are not preassigned for all $i_n = 0, \ldots, M_n$, so that all variables of the form y_{0l} are free to be assigned; however, some zero-one variable y_{k0} ($k = 0, \ldots, M_0$) in (8) may be preassigned to 0. In this case (8) may not have a feasible solution. To resolve this problem, each such y_{k0} is freed and the corresponding cost h_{k0} is redefined to be some large number. The optimal solution of (8) then exists and can be computed. If the computed solution contains a variable y_{k0} that was originally preassigned to 0, then the variable $z^n_{i^k_1 i^k_2 \cdots i^k_{n-1} 0}$

was also preassigned to zero. In this case the constraints in the original problem are satisfied by setting $z^n_{0\cdots i^k_m\cdots 0} = 1$ for $m = 1, \ldots, n-1$ when $i^k_m \neq 0$.

3.2. Maximization of the nonsmooth function $\Phi_n(u^n)$

The next key part of the final algorithm is the solution of the problem

$$\text{Maximize } \{\Phi_n(u^n \; ; \; u^{n+1}, \ldots, u^N, c^N) : u^n \in \mathbb{R}^{M_n+1}\} \quad (10)$$

The evaluation of $\Phi_n(u^n \; ; \; u^{n+1}, \ldots, u^N, c^N)$ requires the optimal solution of the corresponding minimization problem (3). This is achieved whenever $n = 3$ by solving a two dimensional assignment problem optimally or when the problem is so small that it can be solved by branch and bound. However, if $n > 3$ and due to real-time needs, (3) is only solved approximately, so that $\Phi_n(u^n \; ; \; u^{n+1}, \ldots, u^N, c^N)$ is not computed accurately. To resolve this major difficulty, a new function $\Psi_n(u^n)$, which is a lower approximation to $\Phi_n(u^n)$, is defined and is used as a merit function to guide the solution of (10). We begin with some properties of Φ_n.

Theorem 2. *Let u^n be any multiplier vector associated with the last constraint set of (2), let Φ_n be as defined in (3), let z^n be the recovered feasible solution of (2), and let $\bar{z}^n$ be an optimal solution of (2). Then, $\Phi_n(u^n \; ; \; u^{n+1}, \ldots, u^N, c^N)$ is piecewise affine, concave and continuous in u^n and*

$$\Phi_n(u^n; u^{n+1}, \ldots, u^N, c^N) \leq v_n(\bar{z}^n) \leq v_n(z^n) \,. \quad (11)$$

As discussed above, the algorithm yields an upper approximation, $\bar{\Phi}_n$, to $\Phi_n(u^n \; ; \; u^{n+1}, \ldots, u^N, c^N)$. Unfortunately, $\bar{\Phi}_n$ does not satisfy the inequality in the Theorem 2. (In fact, $\bar{\Phi}_n$ can be either bigger or smaller than $v_n(\bar{z}^n)$, so that $\bar{\Phi}_n$ cannot be used as an approximation of Φ_n in the course of the nonsmooth optimization algorithm.) The following definition constructs the aforementioned function that will be used as a lower approximation of Φ_n and that will satisfy the inequalities in Theorem 2.

Definition 1. Let $n \geq 3$ and let $\bar{u}^3, \bar{u}^4, \ldots, \bar{u}^{n-1}$ be the multiplier vectors obtained in the course of the nested relaxation algorithms in solving (5). Let u^n be given. The function Ψ_n is defined recursively via

$$\Psi_n(\bar{u}^3, \ldots, \bar{u}^{n-1} \; ; \; u^n) = \begin{cases} \Phi_n(u^n \; ; \; u^{n+1}, \ldots, u^N, c^N) \\ \quad \text{if (5) is solved optimally} \\ \Psi_{n-1}(\bar{u}^3, \ldots, \bar{u}^{n-2}; \bar{u}^{n-1}) \\ \quad - \sum_{i_n=0}^{M_n} u^n_{i_n} \quad \text{otherwise,} \end{cases} \quad (12)$$

Note that Ψ is well-defined since (5) can always be solved optimally if $n = 3$.

Although the dependence of Ψ_n on $\bar{u}^3, \ldots, \bar{u}^{n-1}$ has been made explicit, the dependence on $u^{n+1}, \ldots, u^N$ has been omitted. (Both Φ_n and Ψ_n depend on the cost coefficients c^n and the multipliers $u^{n+1}, \ldots, u^N$, but Φ_n does not depend on $\bar{u}^3, \ldots, \bar{u}^{n-1}$.) The following theorem establishes the fact that Ψ_n is a lower approximation to Φ_n.

Theorem 3. *Given Definition 1,*

$$\Psi_k(\bar{u}^3, \ldots, \bar{u}^{k-1} \; ; \; u^k) \leq \Phi_k(u^k \; ; \; u^{k+1}, \ldots, u^N, c^N) \quad \text{for all } k = 3, \ldots, n. \quad (13)$$

Most of the algorithms for non-smooth optimization are based on generalized gradients called *subgradients*, given by the following definition.

Definition 2. The set $\delta\Phi_n(u)$ is called a *subdifferential* of Φ_n at u and is defined by

$$\delta\Phi_n(u) = \{q \in \mathbb{R}^{M_n+1} \mid \Phi_n(w \; ; \; u^{n+1}, \ldots, u^N, c^N) - \Phi_n(u \; ; \; u^{n+1}, \ldots, u^N, c^N) \leq q^T(w-u) \; \forall \, w \in \mathbb{R}^{M_n+1}\} \,. \quad (14)$$

A vector $q \in \delta\Phi_n(u)$ is called a *subgradient*.

If z^n is an optimal solution of (3) computed during evaluation of $\Phi_n(u^n \; ; \; u^{n+1}, \ldots, u^N, c^N)$, differentiating Φ_n with respect to $u^n_{i_n}$ yields the following i_n-th component of a subgradient g of $\Phi_n(u^n \; ; \; u^{n+1}, \ldots, u^N, c^N)$:

$$g_0 = 0$$

$$g_{i_n} = \sum_{i_1=0}^{M_1} \cdots \sum_{i_{n-1}=0}^{M_{n-1}} z^n_{i_1\cdots i_n} - 1 \quad (15)$$

$$\text{for } i_n = 1, \ldots, M_n.$$

If z^n is the unique optimal solution of (3), $\delta\Phi_n(u^n) = \{g\}$, and Φ_n is differentiable at u^n. If the optimal solution is not unique, then there are finitely many such solutions, say $z^n(1), \ldots, z^n(S)$. Given the corresponding subgradients, $g^1, \ldots, g^S$, the subdifferential $\delta\Phi(u^n)$ is the convex hull of $\{g^1, \ldots, g^S\}$ [Goffin (1977)]. When $n > 3$, the algorithms used in this work only compute high quality suboptimal solutions of (3), so that (15) is only an approximation $\hat{g}$ to the actual subgradient g.

One of the most widely used methods for nonsmooth optimization is the subgradient algorithm [Fisher (1981), Geoffrion (1974), Goffin (1977), Held, Wolfe and Crowder (1974)], which is the nonsmooth analogue to the steepest ascent method. Analogous to conjugate gradient methods for smooth optimization is the class called "bundle methods" [Lemarechal (1978)]. This includes the space dilation method of Shor (1985), the conjugate subgradient method of Wolfe (1975), and the "bundle - trust region" method due to Schramm and Zowe (1992). We currently use the conjugate subgradient method.

3.3. Summary of the Lagrangian relaxation algorithm

Having described each of the algorithm parts, we now return to the multidimensional assignment (2) and summarize the algorithm in this subsection. Starting with the N-dimensional assignment problem (2), i.e. $n = N$, the developed algorithm is recursive in that the N-dimensional assignment problem is relaxed to a $(N-1)$-dimensional one by incorporating one set of constraints into the objective function using a Lagrangian relaxation of this set. Ideally, this problem is then maximized with respect to the Lagrange multipliers and the corresponding solution is used to recover a feasible solution of the N-dimensional problem. Each $(N-1)$-dimensional problem is then solved in a similar manner, and the process is repeated until it reaches the two-dimensional problem, which is solved by the an algorithm that produces an optimal solution of this two-dimensional problem. Here we describe one loop in this procedure. Thus, assume $N \geq 3$ and let $n \in \{3, \ldots, N\}$

3.4. The Lagrangian Relaxation Algorithm for the n-Dimensional Assignment Problem

To obtain a high quality solution of the n-dimensional assignment problem (2), construct a sequence of multipliers $\{u_k^n\}_{k=0}^{\infty}$ in the course of maximizing $\Phi_n(u^n)$ defined in (3) and a corresponding sequence of feasible solutions $\{z_k^n\}_{k=0}^{\infty}$ of the n-dimensional problem as follows:

A. Initialization: Choose an initial approximation u_0^n. For the n-dimensional assignment problem (2), set the lower bound of the optimal solution value to $\bar{\Psi}_n = -\infty$, set the best computed solution value $\bar{V}_n = \infty$, and the best solution $Z^n = \emptyset$.

B. Given u_k^n, form the relaxed problem (3) and the equivalent $(n-1)$-dimensional assignment problem (5). Compute an optimal or good suboptimal solution z_k^{n-1} of (5). Compute the lower approximation $\Psi_n(u_k^n)$ of $\Phi_n(u_k^n)$ as defined in Definition 1.

C. Recover a feasible solution z_k^n of the n-dimensional assignment problem (2) using the procedure described in first subsection and compute the objective function value $v_n(z_k^n)$ for (2).

D. If $\bar{\Psi}^n < \Psi_n(u_k^n)$, set $\bar{\Psi}_n = \Psi_n(u_k^n)$. If $\bar{V}_n > v_n(z_k^n)$, set $\bar{V}_n = v_n(z_k^n)$ and $Z^n = z_k^n$.

E. Use a non-smooth optimization technique to take a step from u_k^n to an updated multiplier u_{k+1}^n in the solution of

$$\text{Maximize } \{\Phi_n(u^n) \ : \ u^n \in \mathbb{R}^{M_n+1}\}$$

(Ψ_n is used as a merit function in that the acceptance of an updated u_{k+1}^n requires $\Psi_n(u_k^n) < \Psi_n(u_{k+1}^n)$.)

F. If the termination criteria for the non-smooth optimization algorithm have not been met, set $k = k + 1$ and return to B.

If $n > 3$, solving the $(n-1)$-dimensional assignment problem (5) to evaluate $\Psi_n(u_k^n)$ will require another Lagrangian relaxation algorithm and is likely to be very expensive. Most line search algorithms assume that evaluating the subgradient is more expensive than evaluating the function, and will make repeated function evaluations to compute the best step length. Since the reverse is true in this case, the line search is integrated into the algorithm and the multiplier u^n_{k+1} is accepted as long as $\Psi_n(u^n_k) < \Psi_n(u^n_{k+1})$ is satisfied.

4. Greedy Algorithms

A simple greedy algorithm for solving (1) can be summarized as follows:

A. Order the assignment problem variables in some way. Set all the variables to *free*.

B. Find the first *free* variable in the list and set it to one. Set all the conflicting variables to zero to keep the constraints satisfied.

C. Repeat step B until no *free* variables are left.

A common way of ordering the variables of (1) is by the cost coefficients. The greedy algorithm thus assigns the variable with the smallest cost first, and so on. The advantage of the greedy algorithm is the low computational cost. The disadvantage is that the computed solution is often very far from optimal. Since all decisions are purely local (made on the basis of the cost coefficients alone), the last few variables remaining *free* will often have very large costs. Indeed, if the assignment problem (1) is sparse (i.e., some variables $z_{i_1 \cdots i_N}$ are preassigned to 0), greedy algorithm will often fail to find a feasible solution in the sense that the computed solution will not satisfy the constraints.

Two strategies are employed to address these shortcomings. First, to assure that a feasible solution will be computed for a sparse problems, the problem is modified so that all the variables $z_{0\cdots 0 i_k 0 \cdots 0}$ with a single nonzero index are feasible (i.e., not preassigned to 0). If necessary, such variables (called *singleton* variables) can be freed by giving them extremely large costs. Second, to achieve better solution quality, multiple instances of greedy algorithm can be run. The algorithm can be organized in the following way:

A. Sort the variable list by cost. Split the list into two parts, the second part containing singleton variables that have higher costs than any non-singleton variable. Set the list of excluded variables to empty. Set the saved solution to empty.

B. Construct the working list of variables by concatenating the part one of the variable list, then the excluded list and then the part two of the variable list.

C. Use the greedy algorithm on the working list of variables.

D. If the computed solution is better that the saved solution, save it.

E. Move some variables from the part one of the main variable list to the excluded list, according to some criterion.

F. Repeat the steps B-E a predefined number of times, or until all the variables have been moved to the excluded list.

The idea behind this algorithm is to produce as many different solutions as possible, in hopes that the best one will be close to optimal. To achieve this goal, the variables that are removed to the excluded list must be part of the current solution, otherwise their removal has no effect. A good strategy is to exclude a predefined number of solution variables with smallest coefficients. The variables on the excluded list are considered before the trailing singleton variables (second part of the main variable list), since the trailing singletons are expensive and are added just to construct a feasible solution.

5. Limited Branch and Bound Algorithm

The branch and bound algorithm [Breu and Brudet (1974), Nemhauser and Wolsey (1988), Rijavec (1992)] is the only known algorithm for solving the multidimensional assignment problem optimally. For problems of any size, however, this algorithm is impractical, since the time required for computing the optimal solution grows exponentially with the size of the problem. The purpose of this section is to give a brief description of a limited branch and bound algorithm, a suboptimal version of the full branch and bound, based on the Lagrangian relaxation.

The limited branch and bound algorithm constructs a sequence of subproblems from the original problem (1). Each subproblem has one or more variables from (1) preassigned to 0 or 1, but is otherwise identical to (1). The preassigned variables are called *branching variables*.

Each subproblem is then solved using the Lagrangian relaxation algorithm [Poore and Rijavec (1993), Rijavec (1992)]. The relaxation algorithm returns a feasible suboptimal solution to the subproblem and also the lower bound on the optimal solution. The branch and bound algorithm is organized as follows:

A. Push the original assignment problem (1) onto the problem stack. Set the best solution value to ∞.

B. Take the top subproblem from the problem stack and solve it using the Lagrangian relaxation algorithm. If the computed lower bound on the optimal solution is bigger than the best solution value found so far, go to E.

C. If the computed solution is better than the best solution found so far, save the solution and record the solution value as the new best solution. If the lower bound is equal to the solution value, the optimal solution along this branch has been found: go to E.

D. Pick a non-singleton variable in the solution with the highest cost coefficient. Preassign this variable to one and push the subproblem onto the top of the stack. Next, preassign this variable to zero and push the subproblem back onto the stack.

E. Repeat steps B-D until either no more subproblems remain on the stack or predetermined number of subproblems have been solved.

Step D is the branching step. Preassigning to zero the variable with the largest cost from the solution not only guarantees a new subproblem with a different solution than the current subproblem, but also offers the best hope of improvement. So that the algorithm remains valid, a problem with the branching variable set to 1 must also be constructed, but the problems are put onto the stack in such order that the problem containing the zero branch is considered first.

Step C is the bounding step. Preassigning a variable to either 0 or 1 reduces the feasible region and increases the lower bound. Thus, if the computed lower bound for some subproblem is bigger than the value of the best solution computed so far, no matter which variables are further preassigned to either 0 or 1, the optimal solution of all the subproblems obtained from the current subproblem will have optimal solution that is worse than the current best solution. The current subproblem can be discarded since further branching will not yield a better solution.

If the above algorithm runs until the stack is empty, an optimal solution of (1) is produced. This approach is similar in spirit to the highly successful work of Held and Karp (1970,1971) on the traveling salesman problem.

To control the execution time, the algorithm is normally terminated after a set number of branches have been taken. Since the zero branch is always considered before the one branch, the algorithm is guaranteed to produce as many different solutions as possible.

6. Numerical Studies

This section will give a comparison of the three different assignment algorithms on a six dimensional assignment problem arising from the tracking problems. Two studies will be presented. For the first one hundred (100) random sparse problems were generated, with $M_k \leq 11$ ($k = 1, \ldots, 6$) and with the average of 150 feasible variables. It should be noted that such problems are considered small since the Lagrangian relaxation algorithm has been used successfully on problems with as many as a quarter of a million feasible variables. However, small problems allowed the studies to be conducted over wider range of the algorithm parameters.

The results of the first study are presented in the first two tables with the objective function values and solution times for 27 test runs for the different algorithms and parameters. The first column identifies the algorithm and parameters that were used to obtain the result, while the second column lists the result. The algorithms are defined as follows:

G_<cycles>_<excluded>: Greedy algorithm, where <cycles> represents the number of times a simple greedy algorithm has been run, while <excluded> is the number of variables that were added to the excluded list each time. For example, G_1_0 represents a single run of the simple greedy algorithm with no excluded variables.

B_<branches>: Branch and bound algorithm, where <branches> is the maximum number of branches that were taken (i.e., the maximum number of subproblems that were solved).

L_<steps>: Lagrangian relaxation algorithm. <steps> is the number of steps taken in the non-smooth maximization loop that lies in the core of the relaxation algorithm [Poore and Rijavec (1993), Rijavec (1992)]. Since the relaxation algorithm employed within the branch and bound took four steps, L_4 is the same algorithm as B_1.

Table 1 presents the objective function values computed by each run. The values are sorted in each column, and columns are sorted from left to right.

algorithm	v	algorithm	v	algorithm	v
B_50	515.10	L_4	521.30	G_10_1	573.91
B_20	515.58	G_60_1	566.50	G_10_3	580.95
B_10	518.80	G_60_10	566.74	G_5_1	581.86
B_5	518.88	G_60_3	567.23	G_10_6	584.20
L_10	519.57	G_60_6	567.56	G_10_10	584.22
L_100	519.89	G_30_1	569.94	G_5_3	584.23
L_50	519.91	G_30_10	571.53	G_5_10	588.86
L_25	519.91	G_30_6	572.34	G_5_6	588.84
B_2	520.58	G_30_3	572.88	G_1_0	590.56

Table 1: Sorted solution values

It is obvious that the Lagrangian relaxation based algorithms (B and L) give significantly better values than the greedy algorithms, and that it is more beneficial to take several branches in the branch and bound than do a large number of nonsmooth optimization steps. The best computed functional value is within 2.35% of the optimal solution, since the best average lower bound that could be found using the relaxation algorithm was 503.01. This performance, however comes at some computational cost. Table 2 shows the total solution time in seconds for each test run, again sorted by columns. The times were obtained on an IBM RS/6000-550 workstation.

Table 2 shows that limited branch and bound is also by far the most expensive algorithm, taking up to 0.6 seconds. The shortest times achieved by greedy algorithms are less than 0.005 seconds, and show in the tables as zero. However, the current implementation of the branch and bound is inefficient. Every time a branch is taken, the resulting assignment problem is solved from scratch, including the preprocessing, data structure setup, etc. Given a sufficient programming effort, the branch and bound and the relaxation algorithms could be tightly integrated, yielding significant speedups. To some extent, the relaxation algorithm itself could also be coded in a more efficient manner. In contrast, greedy algorithms are relatively simple and it is unlikely that the better coding could result in much shorter execution times.

algorithm	t	algorithm	t	algorithm	t
G_1_0	0.00	G_30_1	0.01	L_10	0.03
G_5_1	0.00	G_30_3	0.01	B_2	0.06
G_5_3	0.00	G_30_6	0.01	L_25	0.09
G_5_6	0.00	G_30_10	0.01	L_50	0.09
G_5_10	0.00	L_4	0.02	B_5	0.10
G_10_1	0.00	G_60_1	0.02	L_100	0.12
G_10_3	0.00	G_60_3	0.02	B_10	0.17
G_10_10	0.00	G_60_6	0.02	B_20	0.30
G_10_6	0.01	G_60_10	0.02	B_50	0.60

Table 2: Sorted total solution times in seconds

Study of Table 2 shows that the fastest relaxation algorithm takes no more time than some greedy algorithms, yet it can be seen in Table 1 that its solution is far superior. Our conclusion is that the best algorithms for the construction of real-time and near-optimal solutions of the noisy multidimensional assignment problems (1) are based on Lagrangian relaxation [Poore and Rijavec (1993), Rijavec (1992)]. It is possible that improvements in the various components of this method as described the third section will significantly improve these relaxation based algorithms.

The second study consists of a selection of results from extensive parametric studies that were done in the course of testing the Lagrangian relaxation algorithm. The studies were run on a variety of tracking problems, with 100 targets and various levels of measurement errors, probability of detection and false alarms. For comparison purposes, the problems were also solved using the simple greedy algorithm (G_1_0) and the branch and bound algorithm with 10 branches (B_10). The Lagrangian relaxation algorithm was run with 4 steps in the non smooth optimization loop (L_4). The full report on the results of these parametric studies exceeds the scope of this paper and can be obtained by contacting the authors. Table 3 represents a small selection of results designed to illustrate the algorithm behavior on large problems.

The results in Table 3 were obtained from four scenarios with varying amount of measurement errors. These were six scan track initiation problems, yielding six dimensional assignment problems. Since the tracking problems were constructed using a simulator, the true solution was known, in the sense of knowing which observations were generated by each target and which represented false alarms. This true solution could also be scored, yielding the column "true" in Table 3.

As before, the numbers in Table 3 represent an average over 100 randomly generated problems. The column "size" gives the number of variables in the problem. For each algorithm, the solution time, the computed solution value and the solution quality are given. Solution quality represents the percentage of the targets that were correctly identified by the algorithm.

The problems represented in Table 3 are medium in size and difficulty, but considerably larger than the problems analyzed in Tables 1 and 2. It can be seen that the solutions for the relaxation and branch and bound produced identical answers, i.e., no better solution was found within 10 branches. The greedy algorithm is far less competitive than on the smaller problems. While the greedy algorithm is slightly faster than the Lagrangian relaxation, the solution is markedly worse. The greedy algorithm requires sorting, which increases the computational effort for larger problems.

Comparing the computed values with the true value, it can be seen that the Lagrangian relaxation consistently computes solutions that are in a sense more probable than the true solutions (true solution is not the optimal solution due to the effects of the measurement errors). This means that there is no sense in trying to compute even better solutions (e.g., by taking more steps), since the solution of the *tracking* problem might not be any better. Greedy algorithm, on the other hand, obtains solutions that have value considerably larger than the true values. The effects can be seen in the only true measure of the solution quality - percentage of the correctly identified targets.

Table 3 shows that for larger problems, the greedy algorithm can not provide the necessary solution quality, while the Lagrangian relaxation solves the problems to below the noise level. It remains to discuss the issue of real time requirements. In many common tracking applications based on the radar, each problem must be solved in 6-12 seconds, the amount of time it takes the radar to complete a full sweep of the space. While all the algorithms in Table 3 could be called "real time" for such problems, normally only a fraction of the time can be devoted to solving the data association problems. Problem formulation, solution post-processing, track updates and input/output require most of the resources. Times of both Lagrangian relaxation and greedy algorithms are fast enough that solving the data association problems becomes only a minor part of the total computational effort required to solve the whole tracking problem.

7. Concluding Remarks

The central problem in multitarget tracking and multisensor data fusion is the data association

Problem			Relaxation			Branch & Bound			Greedy		
scenario	size	true	t	v	quality	t	v	quality	t	v	quality
1	5378	1168	0.16	1160	100	4.18	1160	100	0.14	1612	96
2	7348	1173	0.24	1143	100	5.81	1143	100	0.20	1584	95
3	7547	1145	0.24	1136	100	5.78	1136	100	0.20	1598	94
4	11896	1145	0.37	1118	99	8.75	1118	99	0.35	1564	93

Table 3: Algorithm comparison for a medium size problem

problem that has been posed here as a multidimensional assignment problem. This NP-hard combinatorial optimization problem occupies such a central place in these areas that the need for fast algorithms will continue for some time in the future. Air traffic control, robot vision, and pattern recognition are but a few of the potential applications.

Acknowledgment:

This work was partially supported by the Air Force Office of Scientific Research through AFOSR Grant Number F49620-93-1-0133 and by the IBM Corporation in Boulder, CO and Owego, NY.

References

Y. BAR–SHALOM (1978), Tracking Methods in a Multitarget Environment, *IEEE Transactions on Automatic Control,* Vol. AC–23, pp. 618–626.

Y. BAR–SHALOM, T. E. FORTMANN (1988), *Tracking and Data Association*, Academic Press, Boston, MA.

Y. BAR–SHALOM (1990), ed., *Multitarget-Multisensor Tracking: Advanced Applications,* Artech House, Dedham, MA.

Y. BAR–SHALOM (1992), ed., *Multitarget-Multisensor Tracking: Applications and Advances,*, Artech House, Dedham, MA.

M. S. BAZARAA , J. J. JARVIS (1989), *Linear Programming and Network Flows, Second Edition*, John Wiley and Sons, New York.

D. P. BERTSEKAS, D. A. CASTANON (1991), and Tsaknakis, H., Reverse auction and the solution of inequality constrained assignment problems, preprint.

S. S. BLACKMAN (1992), Association and fusion of multiple sensor data, in *Multitarget-Multisensor Tracking: Applications and Advances,* Y. Bar–Shalom, ed., Artech House, Dedham, MA.

R. BREU, C. A. BURDET (1974), Branch and bound experiments in 0-1 programming, in Balinski, M.L., ed., *Mathematical Programming Study 2: Approaches to Integer Programming,* North Holland Publishing Company, Amsterdam.

S. DEB , K. R. PATTIPATI and Y. BAR-SHALOM (1992), A Multisensor-Multitarget Data Association Algorithm for Heterogeneous Systems, preprint.

M. L. FISHER (1981), The Lagrangian Relaxation Method for Solving Integer Programming Problem, *Management Science*, Vol. 27, No. 1, pp. 1–18.

A. M. FRIEZE, J. YADEGAR (1981), An Algorithm for Solving 3–Dimensional Assignment Problems with Application to Scheduling a Teaching Practice, *Journal of the Operational Research Society*, Vol. 32, pp. 989–995.

M. R. GARVEY , D. S. JOHNSON (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., CA.

A. M. GEOFFRION (1974), Lagrangean Relaxation for Integer Programming, in Balinski, M.L., ed., *Mathematical Programming Study 2: Approaches to Integer Programming*, North Holland Publishing Company, Amsterdam.

J. L. GOFFIN (1977), On Convergence Rates of Subgradient Optimization Methods, *Mathematical Programming 13*, pp. 329–347

M. HELD, R. M. KARP (1970), The traveling salesman problem and minimal spanning trees, *Operations Research 18*, pp. 1138–1162.

M. HELD, R. M. KARP] (1971), The Traveling Salesman Problem and Minimum Spanning Trees: Part II, *Mathematical Programming* 1, pp. 6–25.

M. HELD, P. WOLFE and H. P. CROWDER (1974), Validation of Subgradient Optimization, *Mathematical Programming 6*, pp. 62–88.

A. JONKER, T. VOLGENANT (1987), A Shortest Augmenting Path Algorithm for Dense and Sparse Linear Assignment Problems, *Computing 38*, pp. 325–340.

A. W. KUHN (1955), The Hungarian method for the assignment problem, *Naval Res. Log. Quart. 2*, pp. 83.

C. LEMARECHAL (1978), Bundle Methods in Nonsmooth Optimization, Lemarechal C. and Mifflin, R., Eds, *Nonsmooth Optimization*, IIASA Proceedings 3, Pergamon, Oxford, pp. 79–102.

C. L. MOREFIELD (1977), Application of 0–1 Integer Programming to Multitarget Tracking Problems, *IEEE Transactions on Automatic Control,* Vol. AC–22, No. 3, pp. 302–312.

G. L. NEMHAUSER, L. A. WOLSEY (1988), *Integer and Combinatorial Optimization*, John Wiley and Sons, New York.

A. B. POORE, N. RIJAVEC (1993), A Lagrangian Relaxation Algorithm for Multidimensional Assignment Problems Arising from Multi-target Tracking, *SIAM Journal on Optimization*, Vol. 3, No. 3, pp. 545–563.

A. B. POORE, N. RIJAVEC, T. BARKER and M. MUNGER (1993), Data association problems posed as multidimensional assignment problems: algorithm development, in I. Kadar and V. Libby, eds, *Signal Processing, Sensor Fusion, and Target Recognition II*, SPIE, pp. 172–183.

D. B. REID (1979), An Algorithm for Tracking Multiple Targets, *IEEE Transactions on Automatic Control,* Vol. AC–24, No. 6, December pp. 843–854.

N. RIJAVEC (1992), A Lagrangian Relaxation Algorithm for Some Multidimensional Assignment Problems, Ph.D. Thesis, Colorado State University, Fort Collins, CO.

H. SCHRAMM, J. ZOWE (1992), A Version of the Bundle Idea for Minimizing a Nonsmooth Function: Conceptual Idea, Convergence Analysis, Numerical Results SIAM Journal on Optimization, Vol 2, pp. 121–152.

J. F. SHAPIRO (1979), A Survey of Lagrangian Techniques for Discrete Optimization, *Annals of Discrete Mathematics 5*, pp. 113–138.

N. Z. SHOR (1985), *Minimization Methods for Non-Differentiable Functions*, Springer-Verlag, Berlin.

P. WOLFE (1975), A Method of Conjugate Subgradients for Minimizing Nondifferentiable Functions, *Mathematical Programming Study 3*, pp. 147–173

P. WOLFE (1976), Finding the Nearest Point in a Polytope, *Mathematical Programming 11*, pp. 128–149.

Received: September, 1993
Accepted: March, 1994

Contact address:

Colorado State University
Fort Collins, CO 80523, U.S.A.
Telephone (303) 491-6695
Fax: (303) 491-2161
e-mail: poore@math.colostate.edu

AUBREY POORE received his Ph.D. in Applied Mathematics from the California Institute of Technology in 1973. He is on the editorial board of Computational Optimization and Applications and the IMACS Technical Committee on Optimization, and is currently Professor of Mathematics at Colorado State University.

NENAD RIJAVEC received his Ph.D. in Mathematics at the Colorado State University in 1992. He is currently on a postdoctoral appointment at the Colorado State University.