Discussion Paper No. 106

BRANCH-AND-BOUND STRATEGIES FOR DYNAMIC PROGRAMMING*

by

THOMAS L. MORIN[1] and ROY E. MARSTEN[2]

September, 1974

[1]The Technological Institute
Northwestern University
Evanston, IL 60201

[2]Sloan School of Management
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

* also appears in Working Paper No. WP 750-74, Sloan School of Management, M.I.T.

ABSTRACT

The use of branch-and-bound methods to reduce the storage requirements in discrete dynamic programs is discussed. Relaxations and fathoming criteria are used to eliminate states whose corresponding subpolicies could not lead to optimal policies. The resulting combined dynamic programming/branch-and-bound algorithms are applied to several typical discrete decision problems, including the traveling-salesman problem and the nonlinear knapsack problem. Computational experience, demonstrating the dramatic savings in both computer storage and computational requirements which were effected utilizing this combined approach, is also reported.

1. Introduction

Consider the following functional equations of a dynamic programming algorithm:

$$f(y) = \min \{h(f(y'), y', d) \mid T(y',d) = y\}, \qquad y \in (\Omega - y_0) \tag{1}$$

with the boundary condition

$$f(y_0) = \xi_0 \tag{2}$$

in which, Ω is the finite nonempty state space; $y_0 \in \Omega$ is the initial state; $d \in D$ is a decision, where D is the finite nonempty set of decisions; $T: \Omega \times D \to \Omega$ is the transition mapping, where $T(y',d)$ is the state that is reached if decision d is applied at state y'; $h: \mathbb{R} \times \Omega \times D \to \mathbb{R}$ is the (monotone) cost function, where $h(\xi,y',d)$ is the cost of reaching state $T(y',d) \in \Omega$ by an initial sequence of decisions that reaches state $y' \in \Omega$ at cost $\xi \in \mathbb{R}$ and is then extended by decision d; $\xi_0 \in \mathbb{R}$ represents the initial cost incurred in the initial state $y_0 \in \Omega$; and in (2) we make the convenient, but unnecessary, assumption that return to the initial state is not possible. The functional equations (1) and (2) are simply mathematical transliterations of Bellman's principle of optimality [1].

The recursive solution of these functional equations to determine the cost of an optimal solution and the subsequent (policy) reconstruction process to determine the set of optimal sequences of decisions is straightforward. However, the evaluation of f(y) by (1) necessitates access to f(y') in high-speed (magnetic core, thin film) computer storage for all states $\{y' \mid T(y',d) = y\}$ and the policy reconstruction process necessitates access to the decisions at state y' which result in f(y) for <u>all</u> states in the state space Ω in low-speed (tape,disk) computer storage. It is common knowledge that in real problems excessive high-speed storage requirements

can present a serious implementation problem, so much so that many problems cannot be solved even on the largest present day computers.

This paper presents a general approach to reducing both the high-speed and the low-speed computer storage requirements of dynamic programming algorithms. In order to accomplish this we invoke some of the elementary, but powerful, techniques of branch-and-bound. In particular we demonstrate that in the evaluation of $f(y)$ the costly procedure of storing $f(y')$ in high-speed memory for all states $\{y' | T(y',d) = y\}$ may not always be necessary. That is, if it can be demonstrated that there does not exist a sequence of decisions which when applied to state y' will lead to an optimal solution, then it is not necessary to store $f(y')$ in high-speed memory (nor is it necessary to store the corresponding decision which led to state y' in low-speed memory). Such states y' can be identified by the use of relaxations and fathoming criteria which are commonly employed in branch-and-bound and other enumerative algorithms -- see [11,12,21,23,34] for example. Our approach has been inspired by the ingenious bounding schemes employed within dynamic programming algorithms in [29] and [35], by observations made by the authors in [26], and by the computational success of the resulting hybrid algorithm [22] -- see also [7,28].

The outline of the paper is as follows. Preliminaries and definitions are presented in § 2. The use of fathoming criteria and relaxations within dynamic programming algorithms is then developed in a general setting in §§ 3. The versatility of the results is manifested via application to a number of classes of problems in § 4. A numerical example of the classical traveling-salesman problem is presented and solved using the results of § 3 and computational experience is presented which demonstrates the dramatic savings in both computer storage requirements and computation time which were effected by applying the results of § 3 to nonlinear knapsack problems. The paper concludes with a discussion of other possible applications and extensions § 5.

2. Preliminaries and Definitions

Let $\mathcal{P} = (D,S,\pi)$ denote a <u>discrete optimization problem</u>, where D is the set of all decisions, $S \subseteq \Delta$ is the set of feasible policies, where Δ is the set of all policies (strings or sequences of decisions) which can be formed by concatenating decisions in D and π: $S \to \mathbb{R}$ is the cost function. If $\xi_1 \in \mathbb{R}$ is the cost of an initial sequence of decisions $\delta_1 \in S$ and $\xi_2 \in \mathbb{R}$ is the cost of a subsequent sequence of decisions $\delta_2 \in S$ then we will find it convenient to denote the cost of $\delta_1\delta_2$ (not necessarily in S) by $\tilde{\pi}(\xi_1,\xi_2) = \pi(\delta_1\delta_2)$. (Alternatively, we could employ a composition operator "$\circ$" and write $\pi(\delta_1\delta_2) = \pi_1(\delta_1) \circ \pi_2(\delta_2) = \xi_1 \circ \xi_2 = \tilde{\pi}(\xi_1,\xi_2)$ -- We shall interpret our results in terms of such returns in § 4.1). Let π^* be the value of an optimal solution to $\mathcal{P}$, i.e., $\pi^* = \min\{\pi(\delta) \mid \delta \in S\}$ and let $S^* \subset S$ be the set of optimal solutions to $\mathcal{P}$, i.e., $S^* = \{\delta \mid \delta \in S \wedge \pi(\delta) = \pi^*\}$.

Assume that the discrete optimization problem $\mathcal{P}$ is represented [17-19] by the <u>finite dynamic program</u> $\mathcal{D}$ (alternatively, a terminating or finite sequential decision process [6,19]). $\mathcal{D}$ is specified by the triple $(\mathcal{A},h,\xi_0)$ where $\mathcal{A} = (\Omega,D,y_0,T,\Omega_F)$ is a <u>finite automaton</u> [4,30]. Recall that Ω is the finite non-empty set of states, D is the finite non-empty set of decisions, $y_0 \in \Omega$ is the initial state, $T: \Omega \times D \to \Omega$ is the transition mapping, $h: \mathbb{R} \times \Omega \times D \to \mathbb{R}$ is the cost function and $\xi_0 \in \mathbb{R}$ is the cost incurred in the initial state y_0. $\Omega_F \subset \Omega$ is the set of final states.

Δ is the set of all subpolicies and policies obtained by concatenating decisions in D. Let $\delta \in \Delta$ denote a subpolicy or policy and let e denote the null decision, i.e. $(\forall\delta \in \Delta), \delta e = e\delta = \delta$. The transition mapping T can be extended to $\Omega \times \Delta \to \Omega$ inductively by

$$(\forall y' \in \Omega),\ T(y',e) = y', \text{ and}$$

$$(\forall y' \in \Omega)(\forall \delta \in \Delta)(\forall d \in D),\ T(y',\delta d) = T(T(y',\delta),d).$$

Similarily, the cost function h can be extended to $\mathbb{R} \times \Omega \times \Delta \to \mathbb{R}$ inductively by

$$(\forall \xi \in \mathbb{R})(\forall y' \in \Omega),\ h(\xi,y',e) = \xi, \text{ and}$$

$$(\forall \xi \in \mathbb{R})(\forall y' \in \Omega)(\forall \delta \in \Delta)(\forall d \in D),\ h(\xi,y',\delta d) = h(h(\xi,y',\delta),T(y',\delta),d).$$

For $y' \in \Omega$ let $\Delta(y')$ denote the set of feasible subpolicies (or policies if $y' \in \Omega_F$) which when applied to y_0 result in state y', i.e., $\Delta(y') = \{\delta \mid T(y_0,\delta) = y'\}$, let $\Delta^*(y') \subseteq \Delta(y')$ denote the set of optimal subpolicies (or policies, if $y' \in \Omega_F$) for state y', i.e., $\Delta^*(y') = \{\delta \mid f(y') = h(\xi_0,y_0,\delta)\}$, and let $\Delta(\Omega_F) = \bigcup_{y' \in \Omega_F} \Delta(y')$ be the set of feasible policies for $\mathcal{D}$. Finally, we define a <u>completion set</u> of feasible subpolicies for each state $y' \in \Omega$ as $\chi(y') = \{\delta \mid T(y',\delta) \in \Omega_F\}$. Notice that $\chi(y_0) = \Delta(\Omega_F)$ = the set of strings accepted by the automaton $\mathcal{A}$.

The optimal value for $\mathcal{D}$ is $f^* = \min\{f(y) \mid y \in \Omega_F\}$ and the set of optimal policies for $\mathcal{D}$ is $\Delta^* = \{\delta \mid f^* = h(\xi_0,y_0,\delta)\}$. A sufficient condition on h to insure that the dynamic programming algorithm (1) and (2) finds the set of optimal policies for the finite dynamic program $\mathcal{D}$ is that h is monotone [6,19,23,27] in the sense that

$$(\forall(\xi_1,\xi_2) \in \mathbb{R} \times \mathbb{R})(\forall \delta \in \Delta)(\forall y' \in \Omega),\ \xi_1 \leq \xi_2 \Rightarrow h(\xi_1,y',\delta) \leq h(\xi_2,y',\delta).$$

Finally, since $\mathcal{D}$ is a representation of $\mathcal{P}$ we have $f^* = \pi^*$ and $\Delta^* = S^*$.

3. Fathoming Criteria and Relaxations

Let $\mathcal{U}$ be an upper bound on the value π^* of any optimal solution to the discrete optimization problem $\mathcal{P}$. Then, since the finite dynamic program $\mathcal{D}$ is a representation of $\mathcal{P}$, it follows that $\mathcal{U} \geq f^*$. For each state $y' \in \Omega$ define a (lower bound) mapping $\ell: \Omega \to \mathbb{R}$ with the property that

$$\tilde{\pi}(f(y'), \ell(y')) \leq h(f(y'), y', \delta) \qquad \forall \delta \in \chi(y') \qquad (3)$$

Our main result is stated in the form of the following theorem.

Theorem 1. For any $y' \in \Omega$ and all $\delta'\delta \in \Delta(\Omega_F)$ such that $\delta' \in \Delta^*(y')$ and $\delta \in \chi(y')$, we have

$$\tilde{\pi}(f(y'), \ell(y')) > \mathcal{U} \Rightarrow \delta'\delta \notin S^* = \Delta^* \qquad (4)$$

Proof. (3) $\wedge$ (4) $\Rightarrow h(f(y'), y', \delta) > f^* \qquad \forall \delta \in \chi(y)$. □

Any state $y' \in \Omega$ which satisfies the Theorem is said to be fathomed [12]. If $y' \in \Omega$ is fathomed then $f(y')$ does not have to be placed in high-speed memory and the corresponding last decision in δ' does not have to be placed in low-speed memory since no feasible completion of δ' can be an optimal policy. That is, in the evaluation of $f(y)$ with the functional equations, information on previously fathomed states is irrelevant. Notice also that no feasible completion of any subpolicy $\delta'\delta'' \in \Delta(y'')$, where $T(y', \delta'') = y'' \notin \Omega_F$, can be in Δ^*. Therefore, as a consequence of Theorem 1, we have the following additional fathoming criterion:

Corollary 1. Let $y' \in \Omega$, $\delta' \in \Delta(y')$ and $\delta \in \chi(y')$ satisfy (4). Then all successive states $y'' \in \Omega$ such that $\Delta(y'') = \{\delta'\delta'' \mid T(y', \delta'') = y''\}$ can also be fathomed.

An obvious upper bound $\mathcal{U}$ on the cost π^* of an optimal solution to the original optimization problem $\mathcal{P}$ is the cost, $\pi(\delta)$, of any feasible solution $\delta \in S$ to $\mathcal{P}$. The lower bounds $\ell(y')$ can be obtained by solving a relaxed [12] version of the residual problem whose initial state is y'. If the cost functions h and π are nonnegative then an immediate lower bound for any state $y \in \Omega$ is 0 in the additive case and 1 (if all costs are ≥ 1) in the multiplicative case corresponding to a total relaxation. However, much better lower bounds can usually be obtained by judicious choice of the relaxation as will be demonstrated in the following section.

Notice that our approach includes conventional dynamic programming as a special case in which $\mathcal{U}$ is some sufficiently large real number and a total relaxation is employed.

4. Examples

In this section the results of §3 are explicated and illustrated vis-á-vis a number of examples. The first example of § 4.1 includes the second two examples: the traveling-salesman problem of § 4.2 and the nonlinear knapsack problem of § 4.3, as special cases.

4.1 Example 1: Separable Cost Functions [6,23,27].

Before addressing specific examples, we shall interpret the results of § 3 for a general class of problems which can be approached with dynamic programming. Consider any discrete optimization problem $\mathcal{P}$ whose cost function π belongs to the class of monotone cost functions $\Pi = \{\pi \mid (\forall \delta_1, \delta_2 \in S),\ \pi(\delta_1\delta_2) = \pi_1(\delta_1) \circ \pi_2(\delta_2)\}$ where "$\circ$" represents a composition operator [6,23,27]. Specifically, let "$\circ$" denote any associative, commutative, isotonic binary operator; common examples of which are "+", addition, "$\times$", multiplication, and the infix operators [2] "$\vee$", disjunction, and "$\wedge$", conjunction. For any $\pi \in \Pi$ and $y' \in \Omega$ we can write (3) as

$$f(y') \circ \ell(y') \leq h(f(y'), y', \delta) \qquad \forall \delta \in \chi(y')$$

Further, for any $\pi \in \Pi$ we can write h as

$$h(f(y'), y', \delta) = f(y') \circ I(y', \delta)$$

where $I(y', \delta)$ denotes the incremental cost of applying policy δ at state y'. Hence, for $\pi \in \Pi$ and any $y' \in \Omega$, (3) reduces to

$$\ell(y') \leq I(y', \delta) \qquad \forall \delta \in \chi(y')$$

and Theorem 1 can be stated as follows:

Proposition 1. If $\pi \in \Pi$ then for any $y' \in \Omega$ and all $\delta'\delta \in \Delta(\Omega_F)$ such that $\delta' \in \Delta^*(y')$ and $\delta \in \chi(y')$ we have

$$f(y') \circ \ell(y') > \mathcal{U} \Rightarrow \delta'\delta \notin S^* = \Delta^*.$$

Where we assume that $f(y)$, $\ell(y)$ and $\mathcal{U}$ are nonnegative in the multiplicative case ("$\circ$" = "$\times$").

The remaining examples of this section have cost functions belonging to the additive subclass $\Pi_+ \subset \Pi$, where $\Pi_+ = \{\pi \mid (\forall \delta_1, \delta_2 \in S),\ \pi(\delta_1 \delta_2) = \pi_1(\delta_1) + \pi_2(\delta_2)\}$.

For $\pi \in \Pi_+$ we can write the functional equation (1) as

$$f(y) = \min \{ I(y', d) + f(y') \mid T(y', d) = y \} \qquad y \in (\Omega - y_0).$$

4.2 Example 2: The Traveling-Salesman Problem [3]

Let $\mathcal{G}_N = (V, E)$ be the directed graph whose vertex set is $V = \{1, 2, .., N\}$. Each edge $(i,j) \in E$ is assigned a nonnegative weight c_{ij}. A tour $t \in \mathcal{T}$ is a cycle which passes through each vertex exactly once. Tour $t = (i_1, i_2, \ldots, i_n, i_1)$ can be given the ordered pair representation $t' = [(i_1, i_2), (i_2, i_3), \ldots, (i_{n-1}, i_n), (i_n, i_1)] \in \mathcal{T}'$. The traveling-salesman problem is to find a tour of minimum weight, i.e., find $t \in \mathcal{T}$ so as to

$$\min_{t' \in \mathcal{T}'} \sum_{(i,j) \in t'} c_{ij} \tag{5}$$

If we think of the vertices as cities and the edge weights as the distances between cities, the problem (5) is to start at some home city, visit all the N-1 other cities and return home in the minimum total distance.

Consider the following dynamic programming algorithm [14] for (5). Let $f(S, j)$ denote the weight of a minimum weight path (subgraph) which starts

at (contains) vertex 1, visits (contains) all vertices in $S \subseteq \{1,2,3,\ldots, N\}$, and terminates at vertex $j \in S$. Then, for all $S \neq \phi$, we have the following functional equations of a dynamic programming algorithm for the traveling-salesman problem

$$f(S,j) = \min \{c_{ij} + f(S-j,i) \mid i \in S-j\} \tag{6}$$

with the boundary condition

$$f(\phi,-) = 0 \tag{7}$$

Notice that (6) and (7) are equivalent to (1) and (2) with $y_0 = (\phi,-)$, $y = (S,j) \in \Omega = \{[(S,j) \mid j \in S \subseteq \{2,3,\ldots, N\}] \cup (\{1,2,\ldots, N\},1)\}$, $y' = (S'-j, i)$, $d = j$, $T(y',d) = T((S-j,i),j) = (S,j)$, $\{(y',d) \mid T(y',d) = y\}$ $= \{((S-j,i),j) \mid i \in S-j\}$, $h(f(y'), y',d) = f(S-j,i) + c_{ij}$, and $\xi_0 = 0$.

The major roadblock to solving even moderate size ($N \geq 20$) traveling salesman problems by the dynamic programming algorithm, (6) and (7), is the excessive high-speed computer storage requirement. The storage bottleneck occurs approximately halfway through the recursion when $n+1 = |S| = [N/2]$, since the evaluation of $f(S,j)$ by (6) requires high speed access to $n\binom{N-1}{n}$ values of $f(S-j,i)$. For $N = 20$, this amounts to a high-speed storage requirement of 92,378 locations. However, we can reduce these requirements by fathoming certain states, thereby expanding the range of traveling-salesman problems which can be solved by dynamic programming.

An upper bound on (5) is easily obtained as the weight of any tour $t' \in \mathcal{T}'$. For example, we could take the minimum of the weights of i) the tour $(1,2,\ldots, N-1, N,1)$ and ii) the (nearest-neighbor) tour constructed by connecting vertex 1 to vertex k where $c_{1k} \leq c_{1j}$ $\forall j \neq 1$, then connecting vertex k to vertex m where $c_{km} \leq c_{kj}$ $\forall j \neq 1$, $j \neq k$, and so on until all vertices in $\{1,2,..,N\}$ are connected, and finally connecting the last vertex to vertex 1. Alternatively, we could take the weight of a tour generated by some efficient heuristic as our upper bound $\mathcal{U}$.

The lower bound $\ell(S,j)$ on the weight of a path $j \rightarrow 1$ which includes all vertices $k \notin S$ is also easily obtained. Since the classical assignment problem is a relaxation of (5) we can set $\ell(S,j)$ equal to the value of the minimum weight assignment to an assignment problem whose weight (cost) matrix $\hat{C}$ is obtained from C by deleting all rows $i \in \{1 \cup (S - j)\}$ and columns $j \in S$. A relaxation of the traveling-salesman problem which can be solved by a "greedy" algorithm is the minimum weight 1 - tree problem which is a minimum (weight) spanning tree on the vertex set $\{2,\ldots, N\}$ with the two edges of lowest weight adjoined at vertex 1 [15,16]. Other relaxations have been discussed in [3,5,34].

Suppose that we have calculated the bounds $\mathcal{U}$ and $\ell(S,j)$. Then Theorem 1 can be stated for the traveling-salesman problem as follows:

Proposition 2. T-S Fathoming Criterion: If

$$f(S,j) + \ell(S,j) > \mathcal{U} \tag{8}$$

then any tour $t \in \mathcal{T}$ which contains a path between vertex 1 and vertex j that connects all vertices in S - j **cannot** be a minimum weight tour, i.e., $t \notin \mathcal{T}^*$.

The use of Proposition 2 is demonstrated on the following simple numerical example.

4.2.1 Example: Wagner [34, p.472]

Consider the directed graph $\mathcal{G}_5$ on the vertex set $\{1,2,\ldots, 5\}$ whose weight matrix C is

$$C = \begin{bmatrix} L & 10 & 25 & 25 & 10 \\ 1 & L & 10 & 15 & 2 \\ 8 & 9 & L & 20 & 10 \\ 14 & 10 & 24 & L & 15 \\ 10 & 8 & 25 & 27 & L \end{bmatrix}$$

where $L \in \mathbb{R}_+$ is sufficiently large.

The calculations involved in the solution of the functional equation (6) and (7), are summarized in Table I. The weight of the optimal tour $\{1,5,2,3,4,1\}$ is 62.

TABLE I. Summary of Calculations for the Conventional DP Solution of the Traveling-Salesman Example Problem

$S = \phi$

(S,j)	f(S,j)	i*
$(\phi,-)$	0	-

$|S| = 1$

(S,j)	f(S,j)	i*
$(\{2\},2)$	10	1
$(\{3\},3)$	25	1
$(\{4\},4)$	25	1
$(\{5\},5)$	10	1

$|S| = 2$

(S,j)	f(S,j)	i*
$(\{2,3\},2)$	34	3
$(\{2,3\},3)$	20	2
$(\{2,4\},2)$	35	4
$(\{2,4\},4)$	25	2
$(\{2,5\},2)$	18	5
$(\{2,5\},5)$	12	2
$(\{3,4\},3)$	49	4
$(\{3,4\},4)$	45	3
$(\{3,5\},3)$	35	5
$(\{3,5\},5)$	35	3
$(\{4,5\},4)$	37	5
$(\{4,5\},5)$	40	4

$|S| = 3$

(S,j)	f(S,j)	i*
$(\{2,3,4\},2)$	55	4
$(\{2,3,4\},3)$	45	2
$(\{2,3,4\},4)$	40	3
$(\{2,3,5\},2)$	43	5
$(\{2,3,5\},3)$	28	2
$(\{2,3,5\},5)$	30	3
$(\{2,4,5\},2)$	47	4
$(\{2,4,5\},4)$	33	2
$(\{2,4,5\},5)$	37	2
$(\{3,4,5\},3)$	61	4
$(\{3,4,5\},4)$	55	3
$(\{3,4,5\},5)$	59	3

$|S| = 4$

(S,j)	f(S,j)	i*
$(\{2,3,4,5\},2)$	65	4
$(\{2,3,4,5\},3)$	57	2 or 4
$(\{2,3,4,5\},4)$	48	3
$(\{2,3,4,5\},5)$	55	3 or 4

$|S| = 5$

(S,j)	f(S,j)	i*
$(\{1,2,3,4,5\},1)$	62	4

We next solve this example problem with (6) and (7), incorporating the fathoming criterion (8). The upper bound $\mathcal{U}$ is 62: the minimum of the weight (65) of the tour (1,2,3,4,5,1) and the weight (62) of a nearest-neighbor tour (1,5,2,3,4,1). The lower bounds $\ell(S,j)$ will be calculated by solving the assignment problem relaxation.

For $|S| = 1$, we have the following calculations:

(S,j)	$f(S,j)$	i^*	$\ell(S,j)$
$(\{2\},2)$	10	1	55
$(\{3\},3)$	25	1	42
$(\{4\},4)$	25	1	40
$(\{5\},5)$	10	1	50

Notice that we can fathom states $\{(\{2\},2),(\{3\},3),(\{4\},4)\}$ immediately by (8), since we have

$$f(S,j) + \ell(S,j) > \mathcal{U}$$

Therefore, the maximum high speed storage for $|S| = 2$ is 1 location as opposed to 4 locations in the conventional DP approach -- only information on state $(\{5\},5)$ is relevant to the calculation of $f(S,j)$ for $|S| = 2$.

Furthermore, by Corollary 1 we can fathom states $\{(\{2,3\},2), (\{2,3\},3), (\{2,4\},2), (\{2,4\},4), (\{2,5\},5), (\{3,4\},3), (\{3,4\},4), (\{3,5\},5), (\{4,5\},5)\}$ before evaluating either $f(S,j)$ or $\ell(S,j)$ since they clearly could not lead to optimal tours. Therefore, for $|S| = 2$ only 3 of the 12 possible states remain. The calculations for these states are presented below:

(S,j)	$f(S,j)$	i^*	$\ell(S,j)$
$(\{2,5\},2)$	18	5	44
$(\{3,5\},3)$	35	5	31
$(\{4,5\},4)$	37	5	28

Therefore, we can fathom states $\{(\{3,5\},3), (\{4,5\},4)\}$ immediately by (8), so that the maximum high-speed storage for $|S| = 3$ is only 1 location as opposed to the 12 locations required in the conventional DP approach.

Again, by Corollary 1 we can also fathom all states except for states $\{(\{2,3,5\},3), (\{2,4,5\},4)\}$ prior to the evaluation of $f(S,j)$ for $|S| = 3$, i.e., we only have to evaluate $f(S,j)$ for 2 of the 12 possible states for $|S| = 3$. The calculations for the 2 remaining states are presented below:

(S,j)	$f(S,j)$	i^*	$\ell(S,j)$
$(\{2,3,5\},3)$	28	2	34
$(\{2,4,5\},4)$	33	2	32

Therefore, we can fathom state $(\{2,4,5\},4)$ by (8), so that the maximum high-speed storage requirement for $|S| = 4$ is 1 location as opposed to the 12 locations required in the conventional DP approach. However, we don't even have to store $f(\{2,3,5\},3)$ because by Corollary 1 the optimal solution must be (1,5,2,3,4,1)!

Notice that, although the computation of the lower bound $\ell(S,j)$ required additional calculations in excess of those required in the convential DP solution, this was offset by the reduction in the number of calculations as a result of the states (S,j) which were fathomed by the Corollary 1 prior to the evaluation of $f(S,j)$.

The drastic reductions in both storage and computational requirements which were effected by employing the fathoming criteria with the DP algorithm are summarized in Table II.

Reference to Table II reveals that the high-speed storage requirement was reduced by 11/12 and the low-speed storage requirement was reduced by 26/33 when the fathoming criterion were used with DP in the solution of this example problem! However, this was partly attributable to the very good (in fact, optimal)

TABLE II. Summary of Reduction in Both Storage and Computational Requirements Effected by Employing Fathoming Criteria in the DP Solution of the Traveling-Salesman Example Problem.

Conventional DP Solution

|S|	No. of States for which f(S,j) is evaluated	Evaluation of f(S,j) No. of Additions	No. of Comparisons
1	4	0	0
2	12	12	0
3	12	24	12
4	4	12	4
5	1	4	1
Σ =	33	48	17

max high-speed storage requirement = 12

total low-speed storage requirement = 33

D.P. with Fathoming Criterion

|S|	No. of States for which f(S,j) is evaluated	Max no. of unfathomed states	Evaluation of f(S,j) No. of Additions	No. of Comparisons	No. of bounds ℓ(S, calculated
1	4	1	0	0	4
2	3	1	3	0	3
3	2	1	2	0	2
4	-	-	-	-	-
5	-	-	-	-	-
Σ =	9	3	5	0	9

max high-speed storage requirement = 1

total low-speed storage requirement = 7

initial upper bound. Only further computational experience with the approach can indicate if it is competitive with other relatively efficient algorithms, such as [16,17] for the traveling-salesman problem. In any case, the approach will significantly enlarge the range of traveling-salesman problems which can be solved with dynamic programming.

4.3 Example 3: The Nonlinear Knapsack Problem [26]

The nonlinear knapsack problem (NKP) can be stated as follows: find $x \in \mathbb{R}_+^N$ so as to

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{N} r_j(x_j) \\ \text{subject to} \quad & \sum_{j=1}^{N} g_{ij}(x_j) \le b_i \qquad i = 1,2,\ldots, M \\ & x_j \in S_j \qquad\qquad\quad j = 1,2,\ldots, N \end{aligned} \tag{9}$$

where $(\forall j)\ S_j = \{0,1,\ldots, K_j\}$ and $r_j: S_j \to \mathbb{R}_+$ is nondecreasing with $r(0) = 0$, $(\forall ij)\ g_{ij}: S_j \to \mathbb{R}_+$ with $g_{ij}(0) = 0$, and $b = (b_1,b_2,\ldots, b_M) \ge 0$. The (NKP) includes both the classical (one-dimensional) knapsack problem [31] and the multidimensional 0/1 knapsack problem [35] as special cases for which $(\forall j)\ r_j(x_j) = c_j x_j$, $(\forall ij)\ g_{ij}(x_j) = a_{ij}x_j$ and K_j is taken as the smallest integer such that $g_{1j}(K_j + 1) > b_1$ or $(\forall j)\ K_j = 1$, respectively. The (NKP) also includes a number of other variants [8] of the classical knapsack problem and the "optimal distribution of effort problem" [20,34], as well as various discrete nonlinear resource allocation problems [7,9] graph-theoretic problems [10,33] and nonlinear integer programming problems [11,13,32], as special cases.

Consider the following (imbedded state space) dynamic programming algorithm, M&MDP [26], for the solution of (9). Let $f(n,\beta)$ denote the maximum objective function value of an undominated feasible solution to (9) in which at most all of the first n variables $(x_1,x_2,\ldots, x_n)$ are positive and whose resource consumption vector does not exceed $\beta = (\beta_1,\beta_2,\ldots,\beta_M)$, i.e., $(\forall i)\ \beta_i \ge \sum_{j=1}^{n} g_{ij}(x_j)$. For $n = 1,2,\ldots, N$, the feasible solution $x = (x_1, x_2,\ldots, x_n)$ is

said to be <u>dominated</u> by the feasible solution $\hat{x} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n)$ if we have both $\sum_{j=1}^{n} r_j(x_j) \leq \sum_{j=1}^{n} r_j(\hat{x}_j)$ and $\sum_{j=1}^{n} g_{ij}(x_j) \geq \sum_{j=1}^{n} g_{ij}(\hat{x}_j)$ with strict inequality holding in at least one of the (M+1) inequalities. For $1 \leq n \leq N$, let F_n be the (domain) set of resource consumption vectors, β, of all undominated feasible solutions $(x_1, x_2, \ldots, x_n)$ to the following subproblem

$$\max \sum_{j=1}^{n} r_j(x_j) \tag{10}$$

$$\text{subject to} \quad \sum_{j=1}^{n} g_{ij}(x_j) \leq b_i \qquad i = 1,2,\ldots, M$$

$$x_j \in S_j \qquad j = 1,2,\ldots, n$$

Also, for $0 \leq n \leq N$, let V_n be the (domain) set of resource consumption vectors $g_n(k) = (g_{1n}(k), g_{2n}(k), \ldots, g_{Mn}(k))$, of all undominated feasible values of $x_n = k \in S_n$. We can demonstrate [26] that for $1 \leq n \leq N$, $F_n \subseteq (V_n \circledast F_{n-1})$ with $F_0 = \phi$, where $(V_n \circledast F_{n-1})$ denotes the set obtained by forming all sums of one element of V_n with one element of F_{n-1}.

Therefore, for $1 \leq n \leq N$ we can recursively generate all feasible candidates for F_n from F_{n-1} and V_n with the following functional equation of M&MDP:

$$f(n,\beta) = \{ r_n(k) + f(n-1, \beta - g_n(k)) \,|\, g_n(k) \in V_n \wedge (\beta - g_n(k)) \in F_{n-1} \wedge \beta \leq b \} \tag{11}$$

with the boundary condition

$$f(0,0) = 0 \tag{12}$$

If the $f(n,\beta)$ corresponding to dominated solutions are eliminated at the end of (or during) each stage n, then (11) and (12) will yield the set of objective function values $\{f(N,\beta) \,|\, \beta \in F_N\}$ of the <u>complete family</u> of undominated feasible solutions to the (NKP). Notice that the functional equations, (11) and (12 together with the dominance elimination, are equivalent to (1) and (2) with $y_0 = (0,0)$, $y = (n,\beta) \in \Omega = \{(n,\beta) \,|\, \beta \in F_n, n = 0,1,\ldots, N\}$, $y' = (n-1, \beta - g_n(k))$, $d = k$, $T(y',d) = T((n-1, \beta - g_n(k)), k) = (n,\beta) \in \Omega$, $\{(y',d) \,|\, T(y',d) = y\} = \{(n-1, \beta - g_n(k)) \,|\, g_n(k) \in V_n \wedge (\beta - g_n(k)) \in F_{n-1} \wedge \beta \leq b\}$, $h(f(y'), y', d) = f(n-1, \beta - g_n(k)) + r_n(k)$, and $\xi_0 = 0$.

M&MDP is basically a one-dimensional recursive search (with dominance elimination) over the sequence of sets $F_0, F_1, \ldots, F_N$ which are all imbedded in the M-dimensional space $B = \{\beta \mid (\forall i)\ \beta_i \in [0, b_i]\}$. As a result of the "imbedded state space" approach [25], M&MDP is not overcome by the "curse of dimensionality" which inhibits the utility of conventional dynamic programming algorithms on higher dimensional problems. However, both the high-speed storage requirements and the computational requirements are directly related to the length $|F_n|$ of the list F_n of undominated feasible solutions at stage n. Specifically, the high-speed storage requirement is on the order of $(2M + 6)\bar{n}$ where $\bar{n} = \max_n \{|F_n|\}$ = maximum list length and, although a very efficient dominance elimination scheme (which eliminates sorting) is employed in M&MDP, the computational time tends to increase exponentially with $\bar{n}$ -- in fact, problems in which $\bar{n} \geq 10{,}000$ might consume hours of computer time. Fortunately, we can employ the fathoming and relaxation results of §3 (with minima replaced by maxima, upper bounds replaced by lower bounds, and vice versa) to great advantage in reducing the list lengths.

Let $\mathcal{L}$ be a lower bound on the value of an optimal solution $x^* \in S^*$ to the (NKP). For any $(n,\beta) \in \Omega$, $f(n,\beta)$ is the value of an optimal solution $\delta' = x' = (x_1'^*, x_2'^*, \ldots, x_n'^*) \in \Delta^*(n,\beta)$ to subproblem (10) for $b = \beta$. Consider the following subproblem for any $(n,\beta) \in \Omega$:

Find $(x_{n+1}, x_{n+2}, \ldots, x_N) \in \mathbb{R}_+^{N-n}$ so as to

$$\max \sum_{j=n+1}^{N} r_j(x_j) \tag{13}$$

subject to

$$\sum_{j=n+1}^{N} g_{ij}(x_j) \leq b_i - \beta_i \qquad i = 1,2,\ldots, M$$

$$x_j \in S_j \qquad j = n+1, n+2, \ldots, N$$

Let $X(n,\beta)$ denote the set of all feasible solutions to subproblem (13). Clearly, for $\delta = \hat{x} = (\hat{x}_{n+1}, \hat{x}_{n+2}, \ldots, \hat{x}_N) \in X(n,\beta)$, $\delta'\delta = x'\hat{x} = (x_1'^*, x_2'^*, \ldots, x_n'^*, \hat{x}_{n+1}, \hat{x}_{n+2}, \ldots, \hat{x}_N)$ is a feasible solution to the (NKP) and we have

$$\sum_{j=1}^{n} r_j(x_j'^*) + \sum_{j=n+1}^{N} r_j(\hat{x}_j) = f(n,\beta) + \sum_{j=n+1}^{N} r_j(\hat{x}_j) \leq \sum_{j=1}^{N} r_j(x_j^*) \text{ for any } x^* \in S^* \quad (14)$$

where S^* is the set of optimal solutions to the (NKP).

Let $u(n,\beta)$ be an upper bound on the value of an optimal solution to subproblem (13). Then, Theorem 1 can be stated for the (NKP) as follows:

Proposition 3. (NKP) Fathoming Criterion: If for any $(n,\beta) \in \Omega$, we have

$$f(n,\beta) + u(n,\beta) < \mathcal{L} \quad (15)$$

then $\nexists\ \hat{x} \in X(n,\beta)$ such that $x'\hat{x} \in S^*$.

Any $\beta \in F_n$ for which (15) holds (for the corresponding $(n,\beta) \in \Omega$) can be eliminated from F_n, thereby reducing the lengths of the lists $F_n, F_{n+1}, \ldots, F_N$. Since $\beta \in F_n$ may give rise to $\prod_{j=n+1}^{k} (K_j + 1)$ members in list $F_k (n+1 \leq k \leq N)$ the effect of fathoming can be significant.

The choice of bounds and relaxations is dependent on the form of both the return functions and the constraints and is discussed in detail in [22]. The objective function value of any feasible solution to the (NKP) provides a lower bound $\mathcal{L}$. One easily computed upper bound $u(n,\beta)$ for any state $(n,\beta) \in \Omega$ is $\sum_{j=n+1}^{N} r_j(K_j)$; another is $\min_{1 \leq i \leq M} \{\rho_{in}(b_i - \beta_i)\}$ where

$$\rho_{in} = \max_{n+1 \leq j \leq N} \left(\max_{1 \leq k \leq K_j} \{r_j(k)/g_{ij}(k)\} \right)$$

We note that for linear problems, as a result of our solution tree structure, linear programming relaxations can be used to particular advantage. That is, in our procedure there is no backtracking and the same LP relaxation of (13) applies to all $(n,\beta) \in \Omega$; with only the $(b-\beta)$ vector changing value. Notice that for fixed n the dual objective function value of any dual feasible solution of this relaxation is an upper bound for all $(n,\beta) \in F_n$. Also, the primal LP relaxation for n+1 is equivalent to that for n with the column corresponding to n+1 deleted, i.e., the LP relaxations are "nested".

The "hybrid" (M&MDP with fathoming) algorithm was coded in FORTRAN IV and applied to several of the "hard" linear problems which were solved with M&MDP [26]. The computational experience with both the hybrid algorithm and the straight M&MDP are presented in Table III. The times presented are the execution (CPU) times in seconds of the codes on M.I.T.'s IBM 370-165.

In all problems of Table III, the decision variables were sequenced in nonincreasing order of $r_j(1)$ for input to both algorithms. The simple $\min_{1\le i\le M}\{\rho_{in}(b_i-\beta_i)\}$ upper bounds were used in the hybrid algorithm. When Problem 8 was solved with the LP relaxations employed at every state in the hybrid algorithm the $\max_n|F_n| = 3$ and the execution time was 4.99 seconds -- Since 0 is a member of the F_n list the savings were on the order of 10^3 (2052 vs 2) and the computational savings were on the order of 10^1 (150.00 seconds vs. 4.99 seconds) with the hybrid algorithm.

Reference to Table III reveals the dramatic reduction in both computer storage requirements (which are a function of $\max_n|F_n|$) and computation time which were effected by incorporating the fathoming criteria and relaxations within M&MDP. In fact, on the basis of the additional computational experience reported in [22], the hybrid algorithm appears to be competitive with even the best available linear 0/1 integer programming algorithm for this special case of the (NKP) and the hybrid algorithm has the ability to solve a much broader class of problems.

TABLE III. Computational Comparison of the Hybrid Algorithm vs. M&MDP

Problem	Problem Size			M&MDP		Hybrid Algorithm	
No.	N	M	K	$\max_n \lvert F_n \rvert$	execution time (sec.)	$\max_n \lvert F_n \rvert$	execution time time (sec.)
4	15	10	1	391	6.36	16	0.29
5	20	10	1	526	6.26	22	0.31
6	28	10	1	814	24.18	62	0.81
7	39	5	1	1043	11.27	51	0.92
8	50	5	1	2053	150.00	49	1.33
12	34	5	1	540	8.28	29	0.38
13	50	5	1	571	12.48	12	0.42

5. Discussion

It has been demonstrated that both the computer storage requirements and also the computational requirements of dynamic programming algorithms can be significantly reduced through the use of branch-and-bound methods. We envision that this hybrid approach may open the doorways to the solutions of many sizes and classes of problems which are currently "unsolvable" by conventional dynamic programming algorithms.

REFERENCES

1. BELLMAN, R. E., Dynamic Programming, Princeton University Press, Princeton, N.J., 1957.

2. BELLMAN, R. E., and ZADEH, L. A., "Decision-Making in a Fuzzy Environment," Management Science, 17 (1970), B141-B164.

3. BELLMORE,M. and NEMHAUSER, G. L., "The Traveling-Salesman Problem: A Survey," Operations Research, 16 (1968), 538-558.

4. BOOTH, T.L., Sequential Machines and Automata Theory, John Wiley and Sons, New York, N.Y., 1967.

5. CHRISTOFIDES, N., "Bounds for the Traveling-Salesman Problem," Operations Research, 20 (1972), 1044-1056.

6. DENARDO, E. V. and MITTEN, L. G., "Elements of Sequential Decision Processes," Journal of Industrial Engineering, XVIII (1967), 106-112.

7. DHARMADHIKARI, V., "Discrete Dynamic Programming and the Nonlinear Resource Allocation Problem," Technical Report CP-74009, Department of Computer Science and Operations Research, Southern Methodist University, Dallas, Texas, March, 1974.

8. FAALAND, B., "Solution of the Value-Independent Knapsack Problem by Partitioning,"Operations Research, 21 (1973), 333-337.

9. FOX, B., "Discrete Optimization Via Marginal Analysis," Management Science, 13 (1966), 210-216.

10. FRANK, H., FRISCH, L. T., VAN SLYKE, R., and CHOU, W. S., "Optimal Design of Centralized Computer Networks," Networks, 1 (1971), 43-57.

11. GARFINKEL, R. S. and NEMHAUSER, G. L., Integer Programming, Wiley-Interscience, New York, N.Y., 1972.

12. GEOFFRION, A. M. and MARSTEN, R. E., "Integer Programming Algorithms: A Framework and State-of-the-Art-Survey," Management Science, 18 (1972), 465-491.

13. GRAVES, G. and WHINSTON, A., "A New Approach to Discrete Mathematical Programming," Management Science, 15 (1968), 177-190.

14. HELD, M. and KARP, R. M., "A Dynamic Programming Approach to Sequencing Problems, Journal SIAM, 10 (1962), 196-210.

15. HELD, M. and KARP, R. M., "The Traveling-Salesman Problem and Minimum Spanning Trees," Operations Research, 18 (1970), 1138-1162.

16. HELD, M. and KARP, R. M., "The Traveling-Salesman Problem and Minimum Spanning Trees - Part II," Mathematical Programming, 1 (1971), 6-25.

17. IBARAKI, T., "Finite State Representations of Discrete Optimization Problems," SIAM Journal on Computing, 2 (1973), 193-210.

18. IBARAKI, T., "Solvable Classes of Discrete Dynamic Programming," Journal of Mathematical Analysis and Applications, 43 (1973), 642-693.

19. KARP, R. M. and HELD, M., "Finite-State Processes and Dynamic Programming," SIAM Journal on Applied Mathematics, 15 (1967), 693-718.

20. KARUSH, W., "A General Algorithm for the Optimal Distribution of Effort," Management Science, 9 (1962), 229-239.

21. LAWLER, E. L. and WOOD, D. E., "Branch and Bound Methods: A Survey," Operations Research, 14 (1966), 699-719.

22. MARSTEN, R. E. and MORIN, T. L., "A Hybrid Approach to Discrete Mathematical Programming," Working Paper, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1974.

23. MITTEN, L. G., "Composition Principles for Synthesis of Optimal Multi-stage Process," Operations Research, 12 (1964), 610-619.

24. MITTEN, L. G., "Branch-and-Bound Methods: General Formulation and Properties," Operations Research 18 (1970), 24-34.

25. MORIN, T. L., and ESOGBUE, A. M. O., "The Imbedded State Space Approach to Reducing Dimensionality in Dynamic Programs of Higher Dimensions," Journal of Mathematical Analysis and Applications, 48 (1974).

26. MORIN, T. L. and MARSTEN, R. E., "An Algorithm for Nonlinear Knapsack Problems," Technical Report No. 95, Operations Research Center, Massachusetts Institute of Technology, Cambridge, Massachusetts, May, 1974, to appear in Management Science.

27. NEMHAUSER, G. L., Introduction to Dynamic Programming, John Wiley and Sons, New York, N.Y., 1966.

28. NEMHAUSER, G. L., "A Generalized Permanent Label Setting Algorithm for the Shortest Path Between Specified Nodes," Journal of Mathematical Analysis and Applications, 38 (1972), 328-334.

29. PROSCHAN, F. and BRAY, T. A., "Optimal Redundancy Under Multiple Constraints," Operations Research, 13 (1965), 800-814.

30. RABIN, M. D. and SCOTT, D., "Finite Automata and their Decision Problems," IBM Journal of Research & Development, 3 (1959), 114-125.

31. SALKIN, H. M. and DE KLUYVER, C. A., "The Knapsack Problem: A Survey," Technical Memorandum No. 281, Department of Operations Research, Case Western Reserve University, Cleveland, Ohio, December, 1972.

32. SCHWARTZ, R. E. and DYM, C. L., "An Integer Maximization Problem," Operations Research, 19 (1971), 548-550.

33. VAN SLYKE, R. and KERSHENBAUM, A., "Network Analysis by Accumulation of Trees," Lecture Notes for the course, Mathematical Programming for Management Decision: Theory and Application, Massachusetts Institute of Technology, Cambridge, Massachusetts, July, 1973.

34. WAGNER, H. M., Introduction to Operations Research, Prentice-Hall, Englewood Cliffs, N.J., 1969.

35. WEINGARTNER, H. M. and NESS, D. N., "Methods for the Solution of the Multi-Dimensional 0/1 Knapsack Problem," Operations Research, 15 (1967), 83-103.