타부탐색, 메모리, 싸이클 탐지를 이용한 배낭문제 풀기

고 일 상
조선대학교 경영학과 Tel. (062) 230-6973
koilsang@chollian.dacom.co.kr

Abstract

In solving multi-level knapsack problems, conventional heuristic approaches often assume a short-sighted plan within a static decision environment to find a near optimal solution. These conventional approaches are inflexible, and lack the ability to adapt to different problem structures. This research approaches the problem from a totally different viewpoint, and a new method is designed and implemented. This method performs intelligent actions based on memories of historic data and learning. These actions are developed not only by observing the attributes of the optimal solution, the solution space, and its corresponding path to the optimal solution, but also by applying human intelligence, experience, and intuition with respect to the search strategies. The method intensifies, or diversifies the search process appropriately in time and space. In order to create a good neighborhood structure, this method uses two powerful choice rules that emphasize the impact of candidate variables on the current solution with respect to their profit contribution. A side effect of so-called "pseudo moves," similar to "aspirations," supports these choice rules during the evaluation process. For the purpose of visiting as many relevant points as possible, strategic oscillation between feasible and infeasible solutions around the boundary is applied for intensification. To avoid redundant moves, short-term (tabu-lists), intermediate-term (cycle detection), and long-term (recording frequency and significant solutions for diversification) memories are used. Test results show that among the 45 generated problems (these problems pose significant or insurmountable challenges to exact methods) the approach produces the optimal solutions in 39 cases.

Ⅰ. 다제약 배낭문제의 수학적 정의

다제약 배낭문제는 다음과 같이 정의 된다;

$$\text{Maximige } Z = C'X$$

$$\text{Subject to : } AX \le B$$

$$C' = (c_1, c_2, c_3, \cdots, c_n)$$

$$X = (x_1, x_2, x_3, \cdots, x_n)$$

$$A = \begin{bmatrix} a_{11}, a_{12}, a_{13}, \cdots, a_{1n} \\ a_{21}, a_{22}, a_{23}, \cdots, a_{2n} \\ \vdots \\ a_{m1}, a_{m2}, a_{m3}, \cdots, a_{mn} \end{bmatrix}$$

$$B = (b_1, b_2, b_3, \cdots, b_m)$$

$$X_i = 0 \text{ or } 1 \text{ for alli}$$

이 배낭문제는 프로젝트 선택문제(Senju and Toyoda 1968), 자본예산(Petersen 1974), 자원 할당 문제 등에 응용되고 있다. 이 문제에서는 각 변수들의 다양한 자원소비량들을 반영하는 다수의 제약조건식이 문제를 상당히 풀기 어렵게 만들게 된다. 최적해의 특성들은 어떤 것들이 있으며, 실행 가능해들의 공간, 탐색 여정 등의 특성들에는 어떤 것들이 있는지 등의 자세한 관찰은 최적해를 찾으려는 지능적 행동들의 개발을 가능하게 한다. 이들 특성들을 살펴 보면 다음과 같다.

1. 최적해는 주어진 자원 사용량속에서 목적함수치를 최대로 하는 값이다.
2. 최적해는 실행 가능한 자원의 사용량을 가지며, 실행 가능해들과 실행 불가능해들사이의 경계선에 매우 가까이 있다. 그러나 최적해는 이 경계에서 가장 가까운 해는 아니다.
3. 최적해는 많은 이웃해들을 가지고 있다. 이들 이웃해들은 다음 이동들을 결정하는 선택규칙에 의해서 결정된다. 결과적으로 다양한 선택규칙들은 서로 다른 이웃구조(Neighborhood stracture)들을 만들어 낸다.
4. 최적해는 가능한 많은 경로들을 통하여 접근이 가능하다. 이 경로들은 실행가능 공간의 위치와 선택 규칙들에 의해서 결정된다. 다른 말로 표현하면, 서로 다른 해 공간과 서로 다른 선택규칙들은 서로 다른 탐색경로를 만들어 내게 된다.
5. 여러개의 최적해가 존재할 수 있으며 이들은 동일한 목적함수치를 갖지만 이들은 서로 다른 자원소비량을 갖을 수 있다. 최적해에 있어서 변수들의 수도 서로 다를 수 있다.
6. 최적해에 있어서 변수들의 수는 몇 개인지 결정할 수 없지만, 이 숫자는 추정이 가능하다.

Ⅱ. 초기실행가능해의 형성

좋은 초기실행가능해를 만들어 내는 것은 최적해에 이르기까지 계산시간을 줄이는데 매우 중요하다. 자원 사용량에 비하여 높은 단위이익을 만들어 주는 변수들은 초기실행가능해에 포함되기에 충분한 매력을 갖는다. 예를 들면, 배낭문제에서 단위이익(C_j/A_j)은 몇몇 발견적 기법들에서 초기 실행가능해를 만드는데 사용된다(Balas and Martin 1980, Martello and Toth 1988). 다제약 배낭문제에서는 각 변수들의 자원 소비량들의 합에 대한 단위이익들이 초기실행가능해를 만들어 내는데 사용되기도 한다(Petersen 1974). 이외에도, 변수들의 이익(C_j)만을 이용하여 초기실행가능 기본해를 만들어 낼수도 있을 것이다. 초기 실행 가능해를 만들어 내기 위해서, 우리의 접근법은 이익과 자원사용 가능량에 의해서 표준화된 최대자원소비량을 사용한다[$\text{Max}\{A_{ij}/(C_j*B_i)\}$].

이 값은 다음부터 최대표준자원 사용량이라고 불리운다. 이 숫자는 한 단위 이익을 산출하기 위해서 필요로하는 각 변수의 상대적인 최대 자원 사용량을 의미한다. 변수들은 최대표준자원 사용량에 기초하여 오름차순으로 정렬된다. 오름차순으로 정렬된 변수들은 차례대로 초기실행가능해속에 포함되어 지는데, 실행불가능해를 만들기 전까지 이 과정은 계속된다. 변수들이 선택되는 순서는 최대표준자원 사용량이 적은 것부터 시작된다.
최소 [변수 | 최대 $\{A_{ij}/(C_j*B_i)\}$]

Ⅲ. 강화

강화전략은 제한된 영역안에서 더 나은 해를 찾으려 시도한다. 다제약 배낭문제속에서, 이 전략은 두 개의 선택규칙인 ADD과 DEL, 전략적 진동, 타부리스트, 가상이동들인 SAVE과 ASPIRE 그리고 싸이클 탐지 등에 의해서 지능적으로 수행된다. 이러한 지능적 행동들은 현재의 공간과 역사적 정보에 따라 시간과 공간속에서 적절하게 활동하게 된다.

1. 선택 규칙들 ADD 과 DEL

지능적인 선택규칙들은 최적해에 이르는 좋은 이웃구조를 만들어 내는데 아주 중요한 역할을 하게 된다. 선택규칙 ADD는 지원변수를 더함으로써 생성되는 해의 최대 표준자원 소비량으로 총 이익을 나눈 비율을 계산한다. 이 규칙은 모든 지원 대상 변수들에 대하여 위의 비율을 계산하고 이중에 최대의 값을 갖는 변수를 다음의 이동으로 선택한다. 자세히 말하면, 현재의 총이익에 변수의 이익을 더하여 새로운 총이익을 계산한다. 다음으로, 변수의 모든 자원소비량들을 현재 해의 총 자원 소비량에 더하고 이 값을 총 사용가능량으로 나누어서 각 자원에 대하여 표준자원 사용량을 계산한다. 계산된 총 표준 자원 사용량들중에서 최대값이 선택되는데 이는 다음의 공식으로 표현된다.

Max{(sum(A_{ij})+A_{ij})/Capacity($_i$)}=총 표준자원 사용량
Sum(A_{ij}) = 현재해의 자원 소비량
A_{ij} = 지원대상 변수의 자원 소비량
Capacity($_i$) = 자원$_i$의 사용가능량

새로 계산된 총이익은 위의 최대 총 표준자원 사용량으로 나누어지고 각 변수들의 근사 총단위 이익들이 계산되어 서로 비교 된다.
마침내 근사 총 단위 이익이 최대인 변수가 다음의 이동으로 선택된다.

Max{변수[Sum(C_j)+C_j)/총 표준자원 사용량]}
Sum(C_j) = 현재 해의 총 이익
C_j = 대상 변수의 이익

반면에, 선택규칙 DEL은 근사 총단위이익을 계산하여 봄으로써 새로이 만들어 지는 해의 총자원 소비의 효율성을 최대화하여 주는 변수를 탈락변수로 결정한다. 이 경우에, 새로운 총이익은 현재의 총이익으로부터 지원대상 변수의 이익을 빼어 계산된다. 각 변수의 자원 소비량은 현재 해의 총자원 소비량으로부터 빼어지고, 계산된 새로운 총자원 소비량은 자원 사용가능량에 의해 나누어진다. 이 계산의 결과는 모든 자원에 대한 총 표준자원 사용량이 된다. 총 표준자원 사용량중에 최대값들이 선택되고, 새로운 해의 근사 총 단위 이익을 계산하기 위해 사용된다. 최대 총표준자원 사용량은 다음과 같이 주어진다.

Max {(Sum(A_{ij})-A_{ij})/Capacity($_i$)}=총 표준자원 사용량
Sum(A_{ij}) = 현재해의 자원 소비량
Aij=지원 대상변수의 자원 소비량
Capacity($_i$)=자원$_i$의 사용가능량

모든 지원대상 변수들의 근사총 단위 이익들이 계산되고 비교된다. 이 근사총 단위이익을 최대로 하는 변수가 다음의 이동을 위해 탈락변수로 선택된다.

Max{변수[(Sum(C_j) - C_j)/최대표준자원 사용량]}
Sum(C_j)= 현재해의 총 이익
C_j=지원대상 변수의 이익

2. 실행가능성을 이용한 전략적 진동

실행가능해와 실행불가능해사이의 경계를 중심으로 하는 전략적 진동은 탐색과정중에 많은 적합한 점들을 방문할 수 있게 해준다. 일반적인 가정은 최적해는 경계선에 가까이 있지만, 가장 가까이 있는 해가 아닐 수도 있다는 것이다. 만일에 현재의 해가 실행가능하다면, 탐색과정은 현재해가 실행 불가능이 될 때까지, 변수들을 하나씩 추가하여 보면서 경계선에 더욱 가까이 가보는 것이다. 만일에 현재의 해가 실행불가능하다면, 탐색과정은 현재해가 실행가능이 될 때까지 변수들을 차례대로 빼내어, 경계선에 가까운 최적해일지도 모르는 적당한 점을 방문하고자 노력한다. 우리는 진동의 수를 증가시킴으로써, 보다 더 나은 해에 도달할 수 있다. 진동의 횟수를 결정하는 간단한 접근법은 문제속에 있는 변수의 수를 고려하는 것이다. 예를 들면, 어떤 문제가 30개의 변수를 가졌다면, 30번의 진동을 하는 것이 바람직해 보인다. 이 숫자는 강화과정속에서 희망적으로 각 변수를 한 번씩은 1에 고정시킬 수 있게 하여 준다.

3. 타부리스트의 길이

타부리스트(진입타부리스트와 진출 타부리스트)는 최근의 해에 더하여 졌거나 최근의 해로부터 탈락된 변수들을 가지고 있게 된다. 최근에 더해진 변수 또는 탈락된 변수는 잠시동안 탈락되거나 더해지는 것이 금지 된다. 타부리스트는 현재해와 목적함수치의 변화에 의한 효과를 다음의 몇몇 반복시도속에서 계속되도록 유지하여 준다. 타부리스트는 "단기메모리"라고 불리우며, 이어지는 몇몇의 이동들 속에서 과거의 변화들이 유지되도록 한다. 전략적 진동과 타부리스트의 주의 깊은 결합은 해의 질을 향상시키는데 크게 공헌한다.

4. 가상이동들 SAVE과 ASPIRE

1) 가상이동 SAVE

가상이동 SAVE의 목적은 지원 대상 변수들을 평가하는 도중에 현재까지 발견된 최선의 해보다 더 좋은 목적 함수치를 만들어 낼 수 있는 실행가능 이동들을 기억하는 부수효과를 만들어 줌으로써 선택규칙 ADD와 DEL을 측면에서 지원하게 된다. 보다 자세히 말하면, 어떤 가능이동이 선택규칙 ADD와 DEL에 의해서 선택된 최선의 이동이 아닐지라도, 이 이동이 실행가능한 더 좋은 목적 함수치를 만들어 낼 수 있다면, 가상이동 SAVE는 이 새로운 최선해를 발견하여 최선의 목적함수치와 최선의 해를 기록하게 된다(마치 탐색과정속에서 이 이동이 실제로 행하여진 이동인 것 처럼). 몇몇의 경우에서는, 이 가상이동은 선택되어야 할 최선의 이동이 아니나, 이러한 이동을 단순히 기록하는 것만으로도 최적해를 찾으려는데 중요한 공헌을 할 수 있다.

2) 가상이동 ASPIRE

"타부" 상태에 있는 변수들은 다음의 몇몇 시행들에서 현재해로부터 더해지거나 빼지는 것이 방지된다. 어떤 경우에는, 타부리스트에 있는 이동들이 지금까지 발견된 최선의 해보다 더 좋은 목적함수치를 갖는 실행가능해를 만들기도 한다.

각 시행기간 동안에 타부리스트에 있는 변수들은 이러한 가능성이 있는지 평가될 수 있으며, 만일 해당되는 타부변수가 있다면, 이 이동은 새로운 가장 좋은 목적함수치와 함께 기록된다. 이러한 종류의 이동이 또다른 "가상이동"이며 최적해를 방문하는데 아주 중요한 역할을 하게 된다. 결과적으로 타부리스트에 있는 변수가 현재까지 발견된 최선의 값보다 더 좋은 목적 함수치를 갖는 실행가능해를 만들어 낼 수 있다면, 가상이동 ASPIRE는 이 가능성을 감지하고 해를 기록하게 된다.

5. 강화속에서의 멈춤규칙들

1) 미리 할당된 전략적 진동의 횟수

탐색강화 과정을 멈추게 하는 가장 간단한 멈춤규칙은 미리 할당해 놓은 전략적 진동의 횟수를 이용하는 것인데, 경험에 의하면 이 숫자는 배낭문제의 변수의 수와 같게 하는게 좋다. 이 숫자는 서로 다른 변수들을 가지고 만들어 낼 수 있는 가능한 이동들의 수를 예측하여 만든 것이다. 전략적 진동의 횟수가 정해진 숫자보다 크게 되면, 현재의 탐색강화과정은 멈추게 된다.

2) 싸이클 탐지

결정적 선택 규칙들을 사용하는 경우에, 탐색강화과정중에서 정해진 전략적 진동의 숫자보다는 싸이클 탐지가 보다 더 지능적인 멈춤 규칙을 제공한다. 탐색과정 중에 만일에 똑 같은 해들을 주기를 가지고 다시 방문하게 된다면, 보다 더 나은 해를 방문할 가능성이 없게 된다. 싸이클을 감지하는 것은 아주 효과적인 멈춤규칙이 될 수 있다. 싸이클이란 "같은 목적함수치를 가지는 동일한 해를 일정한 숫자의 시행뒤에 반복적으로 다시 방문하는 것을 말한다." 싸이클을 감지하기 위해서 같은 정보에 대하여 최소한 두 개의 리스트를 유지할 필요가 있는데, 하나는 전체 정보(예를 들면, 해 또는 이동들의 전체순서)이며, 다른 하나는 전체정보중 중요한 부분정보(예를 들면, 최근의 5개의 이동들 또는 5개의 최근해)들이다. 이 부분정보는 최근의 정보를 반영하기 위해서 각 시행마다 변하게 된다. 싸이클 탐지의 정확도는 이 부분정보의 양과 질에 전적으로 의존하게 된다. 싸이클 감지를 위하여 불완전한 정보가 사용될 수 있다. 예를 들면, 다음과 같은 두 개의 리스트 List-A 와 List-B가 있다고 가정하자.

List-A =(X_1, X_3, X_4, X_2, X_5, X_6, X_7, X_1, X_3, X_2, X_5, X_6, X_7, X_1)

List-B = (X_2, X_5, X_6, X_7, X_1)

이 예에서 List-A는 강화과정속에서 모든 이동들의 리스트이며, 변수들은 진입 또는 진출변수들의 순서를 나타낸다. List-B는 최근의 5개의 이동들의 순서를 나타낸다. 매 시행에서, 선택된 변수의 도입에 의해서 길이가 6이 되면 맨 처음변수를 빼므로써 길이를 5로 유지한다. 싸이클을 감지하기 위해서, 우리는 List-A에 있는 5개의 순차적 변수들이 List-B와 똑같은가 비교하게 된다. 현재의 List-B가 List-A의 마지막 부분인 (X_2, X_5, X_6, X_7, X_1)을 제외하고도 확실히 List-A의 한 부분이다. 하나의 싸이클이 감지된다. 어떤 경우들에는 이것은 싸이클이 아닐 수 있다. List-AA와 List-BB처럼 보다 나은 싸이클 탐지를 위해서 목적함수치와 같은 추가적 정보가 동시에 기록되어야 한다.

List-AA = (73, 75, 71, 68, 74, 70, 69, 73, 75, 68, 74, 70, 69, 73)

List-BB = (68, 74, 70, 69, 73)

그러나 싸이클 탐지를 위한 계산시간은 두배가 되며, 경제적인 비효율성 때문에 이의 효과가 퇴색되기 쉽다. 따라서 두가지 정보를 동시에 사용하는 것보다는 둘중에 하나의 불완전한 정보를 사용하는 것이 좋을 듯 싶다. 다른 방안으로는 탐색과정중에 두가지 정보를 모두 유지하지만, 가끔씩 싸이클 탐지를 행하는 방법이다. 이 경우에는, 이러한 탐지행동을 언제하여야 할 것인가가 또 다른 어려운 일이 된다. 멈춤규칙으로 싸이클 탐지를 이용하는 경우에 있어서도, 수백의 이동들을 기록하는 것을 막음으로써 계산시간의 효율성을 실현시키기 위해서 미리 정한 전략적 진동 횟수와 같은 추가적인 멈춤규칙을 세워 두는 것이 경제적이다.

Ⅳ. 다양화

다양화전략은 탐색과정으로 하여금 실행가능공간중에서 덜 개발된 지역을 방문할 수 있도록 하기 위해서 사용되어 진다. 다양화의 기본 구조는 최적해나 근사 최적해를 찾기 위해서 새로운 지역에서 탐색과정을 자주 재시작하는 것이다. 이러한 목적을 위하여, 장기 메모리 장치(빈도 횟수와 같은)가 거의 가보지 못한 지역을 찾거나 최적해가 있을 만한 매력적인 지역을 찾기 위해서 사용된다. 최근의 개선해들 중에서 가장 좋은 3개의 해에 속해 있는 변수들의 빈도를 FREQUENCY-3BEST로, 모든 개선해들 속에서의 변수들의 빈도를 FREQUENCY-IMPROVING으로, 그리고 모든 실행가능 및 실행 불가능해들 속에서의 변수들의 빈도를 FREQUENCY-ALL로 단순히 기록하여 사용함으로써 우리는 탐색다양화과정을 지능적으로 행할 수 있다. 개선해들을 기록하기 위해서 IMPROVING-ISET과 일부의 초기 실행 가능해들을 기억하기 위해서 ISET-LIST라는 메모리가 만들어 진다. 다음은 이들 메모리를 사용하여 다양화를 추구하는 추론기관의 예이다.

단계 0

최대표준자원사용량이 낮은 변수들을 이용하여 초기 실행가능해를 만든다. 이 해를 ISET-LIST에 기억시킨다. IMPROVIMG-ISET을 만들어 비워둔다. 탐색강화과정을 실행한다. IMPROVING-ISET에 개선해들을 기억시킨다. IMPROVING-ISET이 비워 있지 않으면 단계 1로 간다. 비워 있으면 FREQUENCY-ALL을 조정하고 단계 4로 간다.

단계 1

FREQUENCY-3BEST와 FREQUENCY-ALL을 조정한다. IMPROVING-ISET을 비워 초기화 한다. FREQUENCY-3BEST의 높은 빈도를 갖는 변수들을 이용하여 초기 실행가능해를 만든다. 만일 새로만든 초기해가 ISET-LIST에 있으면, 단계 2로 간다. 그렇지 않으면 만들어진 초기해를 ISET-LIST에 집어 넣는다. 만들어진 초기해로부터 탐색다양화과정을 실행한다. IMPROVING-ISET이 비워있지 않으면 단계 1을 반복한다. 만일 IMPROVING-ISET이 비워 있으면 FREQUENCY-ALL을 조정하고 단계 2로 간다.

단계 2

FREQUENCY-IMPROVING의 높은 빈도를 갖는 변수들을 이용하여 초기 실행가능해를 만든다. 만들어진 초기 실행가능해가 과거 초기해들의 집합인 ISET-LIST에 있으면 단계 3으로 간다. 그렇지 않으면, 초기해를 ISET-LIST에 넣는다. 만들어진 초기해로부터 강화과정을 시행한다. IMPROVING-ISET이 비어있지 않으면 단계 1로 간다. IMPROVING-ISET이 비어 있으면, FREQUENCY-ALL을 조정하고, 단계 3으로 간다.

단계 3
FREQUEBCY-ALL의 높은 빈도를 갖는 변수들을 이용하여 초기 실행 가능해를 만든다. 만들어진 초기 실행가능해가 ISET-LIST에 있으면, 단계 4로 간다. 그렇지 않으면 만들어진 초기해를 ISET-LIST에 집어 넣는다. 이 초기해로부터 시작하는 탐색강화과정을 실행한다. IMPROVING-ISET가 비어 있지 않으면, 단계 1로 간다. IMPROVING-ISET이 비어 있으면, FREQUENCY-ALL을 조정하고 단계 4로 간다.
단계 4
무작위 추출을 이용하여 초기 실행가능해를 만든다. 이 초기해로부터 시작되는 탐색강화과정을 시행한다. IMPROVING-ISET이 비어있지 않으면, 단계 1로 간다. IMPROVING-ISET이 비어 있으면, FREQUENCY-ALL을 조정하고 단계 5로 간다. 이 단계에서 초기해들은 기억되지 않는다.
단계 5
FREQUENCY-ALL의 높은 빈도를 갖는 변수들을 이용하여 초기 실행가능해를 만든다. 만들어진 초기해가 ISET-LIST에 있으면, 단계 4로 간다. 없으면 이 초기해로부터 출발하는 탐색강화과정을 진행한다. 그 결과로 IMPROVING-ISET이 비어 있지 않으면, 단계 1로 간다. IMPROVING-ISET이 비어 있으면, FREQUENCY-ALL을 조정하고 단계 4로 간다.

V. 결론

위에서 제안된 방법의 효율성을 측정하기 위해서 다섯 종류의 다제약 배낭문제들이 만들어 졌다. 이들은 무상관, 약상관, 중상관, 강상관, 극상관 배낭문제들인데 각각의 경우에 변수가 20개 제약조건이 10개인 문제가 3개, 변수가 40개 제약 조건이 20개인 문제가 3개, 변수가 60개 제약조건식이 30개인 문제가 3개씩 모두 9개가 만들어 졌으며, 총 45개의 문제들이 위의 방법에 의하여 실험되었다. 타부리스트의 길이는 1, 2, 3이 고정된 값으로 사용되었다. 45문제 중에서 39개의 경우에 있어서 최적해가 발견되었다. 실험결과는 타부리스트가 3인 경우에 다른 두 경우보다 최적해에 더 많이 이르렀음을 보여준다. 배낭문제에서 최적해, 실행가능영역, 탐색경로 등의 특성들에 대하여 면밀한 분석을 통하여 개발된 타부탐색의 확장 모형은 우리로 하여금 작은 수의 시행을 통하여 최적해나 근사해에 효과적으로 도달하게 해준다. 이 글에서 제안된 탐색방법은 인간의 지능을 모방하여 개발된 지능적 행동들을 시간과 공간에 적합하게 수행하여 최적해를 찾으려 최선을 다한다. 이 방법은 학습효과를 얻어내기 위해서, 단기, 중기, 장기 메모리를 탐색과정속에서 적절히 사용하고 있으며, 이들의 메모리에 기초한 학습은 탐색강화와 탐색다양화 전략과 조화를 이룬다. 이 연구에서 제안된 지능적 행동들은 배낭문제와 같은 NP-Complete 한 문제들을 푸는데 유용하게 사용되어 질 수 있을 것이다.

참고문헌

Abelson, H., G. J. Sussman, and J. Sussman, Structure and Interpretation of Computer Programs, The MIT press, McGraw-Hill Book Company, 1985.

Balas, E. and E. Zemel, "An Algorithm for Large Zero-One Knapsack Problems," Operations Research, Vol. 28, No. 5, September-October 1980, 1131-1154.

Balas, E. and C. H. Martin, "Pivot and Complement - A Heuristic for 0-1 Programming," Management Science, Vol. 26, No. 1, January 1980, 86-96.

Freville, A. and G. Plateau, "An Efficient Preprocessing Procedure for the Multidimensional 0-1 Knapsack Problem," Proceedings of the conference "Viewpoints on Optimization" Grimentz, September 1990, Switzerland.

Garey, M. and D. Johnson, Computers and Intractability: A Guide to the theory of NP-Completeness, Freeman, San Francisco, 1979.

Geoffrion, A. M., "An Improved Implicit Enumeration Approach for Integer Programming," Operations Research, Vol. 17, No. 3, 1969, 437-454.

Glover, F., "A Multiphase-Dual Algorithm for the Zero-One Integer Programming Problem," Operations Research, Vol. 13, No. 3, 1965, 879-919.

Glover, F., "Heuristics for Integer Programming using Surrogate Constraints," Decision Sciences, Vol. 8, 1977, 156-166.

Glover, F., "Artificial Intelligence, Heuristic Frameworks and Tabu Search," Managerial and Decision Economics, Vol. 11, 1990a, 365-375.

Glover, F., "Tabu Search: A Tutorial," Center for Applied Artificial Intelligence, University of Colorado, February 1990b.

Glover, F., "Simple Tabu Thresholding in Optimization," Graduate School of Business, University of Colorado, October 1991.

Glover, F., D. Klingman, and N. V. Phillips, Network Models in Optimization and Their Applications in Practice, A Wiley-Interscience Publication, 1992.

Karp, R., "Reducibility Among Combinatorial Problems," in Complexity of Computer Computations, Miller, R. E. and J. W. Thatcher (eds), Plenum Press, New York, 1972. 85-103.

Kelly, J. P., B. L. Golden, and A. A. Assad, "Large-scale Controlled Rounding using Tabu Search with Strategic Oscillation," Annals of Operations Research, 1993.

Kochenberger, G. A., B. A. McCarl, and F. P. Wyman, "A Heuristic for General Integer Programming," Decision Sciences, Vol. 5, No. 1, 1974, 36-44.

Lee, J. S. and M. Guignard, "An Approximate Algorithm for Multidimensional Zero-One Knapsack Problems-A Parametric Approach," Management Science, Vol. 34, No. 3, 1988, 402-410.

Luger, G. F. and W. A. Stubblefield, Artificial Intelligence and the Design of Expert Systems, The Benjamin/Cummings Publishing Company, Inc. 1989.

Magazine, M. J. and O. Oguz, "A Heuristic Algorithm for the Multidimensional Zero-One Knapsack Problem," European Journal of Operational Research, Vol. 16, 1984, 319-326.

Martello, S. and P. Toth, "A New Algorithm for the 0-1 Knapsack Problem," Management Science, Vol. 34, No. 5, 1988, 633-644.

Nemhauser, G. L., and L. A. Wolsey, Integer and combinatorial Optimization, A Wiley-Interscience Publication, 1988

Petersen, C. C., "A Capital Budgeting Heuristic Algorithm Using Exchange Operation," AIIE Transactions, Volume 6, No. 2, June 1974, 143-150.

Petersen, C. C., "Computational Experience with Variants of the Balas Algorithm Applied to the Selection of R&D Projects," Management Science, Vol. 13, No. 9, May 1967, 736-750.

Rich, E., Artificial Intelligence, McGraw-Hill, Inc. 1983

Senju, S. and Y. Toyoda, "An Approach to Linear Programming with 0-1 Variables," Management Science, Vol. 15, 1968, 196-207.

Tanimoto, S. L., The Elements of Artificial Intelligence - An Introduction Using LISP, Computer Science Press, Inc, 1987.

Toyoda, Y., "A simplified Algorithm for Obtaining Approximate Solutions to 0-1 Programming Problems," Management Science, Vol. 21, No. 12, 1975, 1417-1427.

Wilensky, R., Common LISP craft, W. W. Norton Company, 1986.

Winston, P. H., Artificial Intelligence, the second edition, Addison-Wesley Publishing Company, 1984.

Winston, P. H., and B. K. P. Horn, LISP, the second edition, 1984.