〈研究論文〉

Journal of the Korean Institute of Industrial Engineers
Vol. 9, No. 1, June 1983

多次元 배낭 문제의 새로운 解法

(A new method for a multi-dimensional Knapsack problem)

朴 淳 達*
朴 盈 滿**

Abstract

The objective of this paper is to present a new method for the multi-dimensional Knapsack problem. Toyoda method and Loulou and Michaelides method are well known for this problem. The new method introduces a new penalty factor for fast convergence and a branching technique for accurate solutions.

The method is tested at IBM370 and shows that the method is slower than Toyoda method, but more accurate than other two methods.

1. 서 론

이 논문은 制約式이 여러 개 존재하는 多次元 배낭 문제를 다룬다. 예를 들면 제한된 자원을 가지고 여러 개의 事業 후보지들 중에서 가장 産出이 큰 사업들을 선택하고자 할 때 이 문제는 다음과 같은 0－1 計劃法으로 표현된다. 즉,

$$P: \quad \begin{aligned} & Max \sum_{j=1}^{n} c_j x_j \\ & s.t. \sum_{j=1}^{n} a_{ij} x_j \le b_i, \quad i=1, \cdots\cdots, m \qquad (1) \\ & x_j = 0 \text{ 또는 } 1 \quad \forall_j \\ & a_{ij} \ge 0, \ b_i \ge 0, \ c_j \ge 0 \quad \forall_{i,j} \end{aligned}$$

이 식은 다시 다음과 같은 식으로 표현될 수도있다. 즉,

$$Q: \quad \begin{aligned} & Max \sum_{j=1}^{n} c_j x_j \\ & s.t \quad \sum r_{ij} x_j \le 1, \quad i=1, \cdots\cdots, m \qquad (2) \\ & x_j = 0 \text{ 또는 } 1 \\ & r_{ij} \ge 0, \ c_j \ge 0 \quad \forall_{i,j} \end{aligned}$$

이들 식에서 $m=1$ 이면 일반적인 배낭 문제라고 한다. 이러한 배낭 문제는 특수한 0－1計劃法 문제로 풀 수 있으며, 특히 發見的 技法중 Greedy 方法이 잘 알려져 있다. 문제 P와 Q는 0－1計劃法 문제의 해법으로 풀 수 있으나 일반적으로 효율성이 낮기 때문에 정확한 解는 아니더라도 효율성이 높은 發見的 技法에 관심을 갖게 된다.

Greedy 方法은 잘 알려져 있는 바와 같이 문제 P에 있어서 $i=1$일 때 각 사업의 무게(weight)를 c_j/a_{ij}로 정의하여 이 무게가 가장 큰 것부터 선택해 나가는 방법이다.

그러나 이러한 방법은 $i>1$일 때는 적용하지 못한다. $i<1$ 일 때의 문제를 多次元 0－1 배낭 문제

* 서울대학교
** 慶南大學校

라고 하며 Toyoda [13], Loulou and Michaelides [9], Kochemberger, et al. [8]의방법이 잘 알려져 있다.

이 들 Greedy類의 技法은 Greedy技法과 같이 각 事業 선택에 있어서 무게를 구하여 선택하는 데 각 사업의 무게를 c_j/V_j의 형태로 구한다. 이 때 V_j를 벌과금(penalty)라 한다. Greedy方法은 V_j로 a_{1j}를 택한 것이다.

Kochenberger, McCarl and Wyman은 P 문제에서 V_j를

$$V_j = \sum_{i=1}^{m} a_{ij} / b_i^* \quad b_i^* = b_i - \sum_{j=1}^{n} a_{ij} x_j \qquad (3)$$

로 사용하고 있다. 이와 같이 벌과금 V_j를 사용하고 남은 자원의 양에 대한 比의 合으로 함으로써 자원을 보호하고자 하는 것이다.

$$V_j = \sum_{i=1}^{m} r_{ij} d_i^\circ / (\sum_{i=1}^{m} d_i^{\circ 2})^{\frac{1}{2}}$$

$$\text{단} \quad d_i^\circ = \sum_{j=1}^{n} r_{ij} x_j$$

로 정의하고 있다.

이 V_j는 앞으로 선택하고자 하는 事業 j의 사용 예정 자원량을 뜻하고 있다. 그런데 자원량이 많은 경우에 많은 량을 사용하는 것과 적은 가용량에서 많은 자원량을 사용하는 것과는 다르기 때문에 이러한 것을 감안하여 $d_i^\circ / (\sum_i d_i^{\circ 2})^{\frac{1}{2}}$ 을 곱해 주고 있다.

적은 가용량에서 많은 자원량을 사용하려면 V_j가 크기 때문에 자연히 c_j/V_j가 작아져 事業 j가 선택될 가능성이 작아진다.

이에 비하여 Loulou and Michaelides는 좀 색다른 벌과금을 사용하고 있다.

Loulou and Michaelides는 Q문제에서

$$V_j = \underset{i}{Max} \ \{d_i^\circ + r_{ij}\} \ (\sum_{k \in Sc} (r_{kj} - r_{ij}) / (1 - d_i^\circ - r_{ij}))$$

Sc : Candidate project 集合

을 사용하고 있다. 여기서 $d_i^\circ + r_{ij}$는 만일 事業 j가 선택되었을 경우의 사용될 총 자원이다. $\sum_k (r_{kj} - r_{ij})$는 미결정 事業들이 요구할 사용량이며 $1 - d_i^\circ - r_{ij}$는 남아있는 자원이다.

그러나 Louliu and Michaelides는 더 나아가 分枝(branching) 개념을 도입하고 있다. 즉 남아 있는 자원 가용량이 남아 있는 사업의 평균 자원 소요량보다 적으면 c_j/V_j 대신에 c_j의 순서로 사업을 선택하는 것이다. 이렇게 함으로써 최종 마무리 단계에서 解를 좀 더 개선시킬 수 있는 것이다.

이 논문에서는 여러가지 방법중에서 Kochenberger et al.의 방법을 개선하고 나아가 解를 개선시키기 위하여 分技 方法을 도입하고자 한다. 일반적으로 解를 구함에 있어 解의 대상을 많이 택하면 택할수록 解가 개선되는 것은 당연하다. 그러나 대신 시간이 걸리게 되며 이것이 지나치면 發見的 기법을 택하는 장점이 없어진다. 이 논문은 실험을 통하여 分技의 정도를 확인하고자 한다.

2. 새로운 技法

이 논문에서 제시할 기법에서는 새로운 벌과금 요소를 도입함과 동시에 分技 기법을 사용한다.

벌과금(penalty)

Kochenberger et el.의 벌과금은 식 (3)에서 보는 바와 같이 남아 있는 자원에 대한 比率의 합이다. 이 경우에는 모든 자원에 꼭 같은 비율을 주고 있다. 그러나 자원 소모율이 가장 큰 것에 더 많은 비중을 둘 수 있다. 즉

$$V_j = \sum_{i=1}^{m} (a_{ij}/b_i^*) + \alpha \ \underset{i}{Max} \ (a_{ij}/b_i^*), \quad \alpha \geqq 0$$

여기서 $\alpha = 0$으로 두면 Kochenberger et al. 方法과 같게 되며, α를 크게하면 최대 자원 소모율이 결정적인 벌과금 요소가 된다. 이렇게 α의 크기를 조정함에 따라 부족한 자원을 더욱 강력하게 보호할 수 있다.

分枝

分枝라는 것은 벌과금을 이용하여 만드는 解이외에 또 다른 解를 만들어 비교함으로써 좋은 解를 선택하여 解의 질을 높이고자 하는 것이다.

먼저 β를 다음과 같이 정의한다.

$$\beta = \underset{j}{Max} \ \underset{i}{Max} \ \{a_{ij} / b_i^*\}, \quad 0 \leqq \beta \leqq 1$$

$Max \ \{a_{ij}/b_i^*\}$는 事業 j에서 각 자원의 最大 소모를 나타내고, β는 事業에서의 최대 자원 소모율을 나타낸다.

여기서 分枝 水準 β^*를 도입해 보자. 즉 해의 정밀도 수준을 높이기 위하여 β가 β^*보다 클 때 지금까지의 방법으로 구한 解 S 1 이외에 c_j의 크가에 따라서 가용 자원이 허용하는 한계 내에서 선택된 사업으로 이루어진 代案의 解 S 2를 구해 간다. 그리하여 마지막 단계에서 $S1$과 $S2$중에서 좋은 解를 선택한다.

여기서 $\beta^* = 0$으로 두면 처음부터 代案의 解를 구하게 되고 $\beta^* = 1$로 두면 分枝가 일어나지 않게 된다.

計算方法

먼저

P : 모든 사업의 集合
PS : 이미 선택된 集合
PC : P/PS
$S1$: 벌과금에 의해 만들어지는 解
$S2$: 分枝에 의해 만들어지는 解
$Z1$: $S1$의 목적 함수 값.
$Z2$: $S2$의 목적 함수 값

라고 하자.

단계 1 : $Z1=Z2=0$, $S1=S2=\phi$, $PS=\phi$라 하고 β^*를 결정한다.

단계 2 : 만일 $PC=\phi$이면 끝낸다. 이때 $Z1$과 $Z2$를 비교하여 큰 값을 가지는 해가 最適解이다. $PC\neq\phi$ 이면 다음 단계로 넘어 간다.

단계 3 : PC의 모든 사업 j에 대해 V_j를 구한다. $\underset{j\in PC}{Max}\ \{c_j/V_j\}=c_s/V_s$가 되는 사업 s를 선택하여 새로운 $s1$을 $S1\cup P_s$으로 한다.

단계 4 : β를 구한다. 만일 $\beta\leq\beta^*$이면 단계 2로 돌아 간다. $\beta>\beta^*$이면 다음 단계로 간다.

단계 5 : 이 分枝가 처음이면 새로운 解 $S3$를 만든다. 즉 $S3$는 $PC/\{s\}$중에서 c_j의 순서로 자원이 허용하는 한도로 사업을 선택한다. 이 解 $S3$의 목적함수 값을 $Z3$라고 한다. 그리고 $S2=S3$ $Z2=Z3$라고 한다.

그리고 두번째 부터는 $Z2=Max\{Z3, Z2\}$라 하고 $S2$는 $S2$와 $S3$중에서 그의 목적 함수값이 큰 것으로 한다. 그리고 단계 2로 돌아 간다.

3. 분석 및 결론

새로운 技法(PB技法)을 Toyoda 技法(TY技法), Loulou and Michaelides 技法(LM技法)과 비교 분석한다. 먼저 해를 구하는 데 소요되는 CPU시간을 비교하고 다음에 解의 정밀도에 대해서 비교한다.

(1) CPU시간

CPU시간을 비교하기 위하여 5가지 크기의 문제 각각에 대해 50 문제씩을 만들어 푸는데 소요되는 시간을 구하여 그 평균 시간을 구하였다. PB技法은 β^*가 1인 경우와 1이 아닌 경우로 구별하였다. 이 계산은 서울大學校의 IBM 370을 사용하였다. CPU시간은 다음 표 1과 같다.

표 1 CPU시간(50문제의 평균시간)

기 법 문제크기($n\times m$)	T Y 技 法	L M 技 法	P S 技 法 $\beta^*=1$	 $\beta^*\neq1$	비 고 ($\beta^*\neq1$의 경우)
10×20	0.74	1.56	1.14	1.58	$\beta^*=0$
20×40	5.04	10.44	6.20	6.98	$\beta^*=0.6$
30×80	26.64	60.28	36.10	37.80	$\beta^*=0.6$
40×120	79.40	176.30	108.20	109.80	$\beta^*=0.74$

(단 PS技法에서 $\alpha=1$로 둠)

이 표를 통하여 PS技法이 分枝를 수행하든 하지 않든 별로 큰 CPU시간의 차이는 없으며 LM 技法보다는 시간상 효과적임을 알 수 있다.

(2) 解의 정밀도

解의 정밀도에 대해서는 PS技法의 경우 $\alpha=0$, $\alpha=10$, $\alpha=100$의 경우를 비교하기로 한다. 그런데 解의 정밀도에 있어서는 絶對的인 解(exact solution)에 대한 오차보다는 그 技法이 만들어내는가장 좋은 解에 대한 相對的 오차를 많이 사용한다. 여기에서는 앞에서와 같이 각 50문제에 대하여 분석해 본 결과 표 2와 같다.

이 표들을 보아 PS技法이 TY, LM技法보다 상대오차에 있어서 우수하다는 것을 알 수 있다. 그리고 PS技法에서 $\beta^*=0$ 즉 分枝를 수행하고 특히 分枝水準을 낮추면 해가 훨씬 우수해 짐을 알 수 있다. 그리고 $\alpha=10$인 경우가 일반적으로 우수함을 알 수 있다.

이상을 종합해 볼 때 PS技法이 Toyoda 技法보다는 CPU시간이 많이 걸리지만 이러한 CPU시간은 解의 정밀도 견지에서 충분히 보상됨을 알 수 있다.

표 2 解의 比較表

문제크기	기 법	TY技法	LM技法	PS (α=0)		PS (α=10)		PS (α=100)	
				β*=0.6	β*=1	β*=0.6	β*=1	β*=0.6	β*=1
10×20	평균상대오차	3.28	3.52	1.47	3.41	0.89	2.11	0.77	2.80
	표 준 편 차	3.77	4.54	2.87	4.05	1.82	3.47	1.59	3.36
20×40	평균상대오차	3.87	2.44	1.44	2.25	1.04	1.60	1.08	1.97
	표 준 편 차	3.53	2.60	1.88	2.79	1.53	2.03	1.75	2.13
30×80	평균상대오차	3.11	1.59	1.1	1.71	0.53	1.00	0.92	1.41
	표 준 편 차	2.10	1.14	1.04	1.22	0.75	1.14	1.12	1.34
40×120	평균상대오차	2.92	1.01	0.94	1.22	0.66	0.78	0.84	1.11
	표 준 편 차	1.83	0.88	0.84	0.94	0.69	0.69	0.78	0.86

REFERENCES

1. E. Balas, "An Additive Algorithm for Solving Linear Programs with Zero-One Variables," Operations Research 13 (1965) 517-546.
2. E. Balas & C. H. Martin, "Pivot and Complement-A Heuristic for 0-1 Programming," Management Science 26 (1980), 86-96.
3. E. Balas & E. Zemel, "An Algorithm for Large Zero-One Knapsack Problem," Operations Research 28 (1980), 1130-1150.
4. J. L. Balintfy & G. T. Ross & P. Sinha & A. A. Zoltners, "A Mathematical Programming System for Preference and Compatibility Maximized Menu Planning and Scheduling," Mathematical Programming 15 (1978), 63-76.
5. B. H. Faaland & F. S. Hiller, "Interior Path Methods for Heuristic Integer Programming Procedures," Operations Research 26 (1978), 1069-1087.
6. F. S. Hiller, "Efficient Heuristic Procedures for Integer Linear Programming with an Interior," Operations Research 17 (1976), 600-537.
7. R. M. Karp, "On the Computational Complexity of Combinatorial Problems", Network 5 (1975), 45-68.
8. G. A. Kochenberger & B. A. McCarl & F. P. Wyman, "A Heuristic for General Integer Programming," Decision Science 5 (1974), 36-44.
9. R. Loulou & E. Michaelides, "New Greedy-like Heuristics for the Multidimensional 0-1 Knapsack Problem," Operations Research 26 (1978), 1101-1114.
10. K. Murty, Linear and Combinatorial Programming, Wiley, New York, 1976.
11. S. Sahni & E. Horowitz, "Combinatorial Problems: Reducibility and Approximation," Operations Research 26 (1978) 718-759.
12. S. Senju & Y. Toyoda, "An Approach to Linear Programming with 0-1 Variables," Management Science 15 (1968), B196-B207.
13. Y. Toyoda, "A Simplified Algorithm for Obtaining Approximate Solutions to Zero-One Programming Problems", Management Science 21 (1975), 1417-1427.
14. H. M. Weingartner & D. N. Ness, "Methods for the Solution of the Mutidimensional 0-1 Knapsack Problem" Operations Research 15 (1967), 83-103.
15. S. H. Zankis, "Heuristic 0-1 Linear Programming: An Experimental Camparison of Three Methods", Management Science 24 (1977), 81-104.
16. 박영만, 많은 制約式을 가진 0-1 Knapsack 問題에 대한 Heuristic 技法, 서울대학교 석사논문, 1982.