한국지능시스템학회 논문지 2010, Vol. 20, No. 6, pp. 755-760
DOI : 10.5391/JKIIS.2010.20.6.755

다중 배낭 문제를 위한 라그랑지안 휴리스틱

A Lagrangian Heuristic for the Multidimensional 0-1 Knapsack Problem

윤유림* · 김용혁**+
Yourim Yoon and Yong-Hyuk Kim

* 서울대학교 컴퓨터공학부
** 광운대학교 컴퓨터소프트웨어학과

요 약

일반적으로 이산 최적화에서의 라그랑지안 방법은 제약조건을 쉽게 다루기 위한 기법이다. 이 방법은 전형적으로 분지한계법에서 상한을 찾을 때 사용한다. 본 논문은 여러 개의 제약조건이 있는 다중 배낭 문제를 위한 새로운 라그랑지안 방법을 제안한다. 기존 라그랑지안 접근법과는 달리 제안한 방법은 라그랑지안 벡터의 새로운 특징에 기초하여 품질 좋은 하한(즉, 가능 해)을 효율적으로 찾을 수 있다. 잘 알려진 큰 규모의 벤치마크 데이터에서 실험을 하였고 제안한 라그랑지안 방법은 기존 방법의 성능을 개선하였다.

키워드 : 다중 배낭 문제, 라그랑지안 휴리스틱

Abstract

In general, Lagrangian method for discrete optimization is a kind of technique to easily manage constraints. It is traditionally used for finding upper bounds in the branch-and-bound method. In this paper, we propose a new Lagrangian search method for the 0-1 knapsack problem with multiple constraints. A novel feature of the proposed method different from existing Lagrangian approaches is that it can find high-quality lower bounds, i.e., feasible solutions, efficiently based on a new property of Lagrangian vector. We show the performance improvement of the proposed Lagrangian method over existing ones through experiments on well-known large scale benchmark data.

Key Words : The multidimensional 0-1 knapsack problem, Lagrangian heuristic.

1. 서 론

다중 배낭 문제는 NP-완전(complete) 문제로 잘 알려진 배낭 문제의 확장된 형태로 이해할 수 있다[1]. 배낭 문제는 크기(size)와 가치(profit)를 갖는 물건(object)의 집합이 주어지면 배낭의 용량(capacity)을 넘지 않는 선에서 가치가 최대가 되도록 물건을 배낭에 담는 방법을 찾는 문제이다. 다중 배낭 문제의 경우에는 용량에 대한 제약조건이 둘 이상이 된다. 예를 들면, 배낭에 담을 수 있는 물건의 총 크기 외에도 담을 수 있는 물건의 총 무게 등이 제약조건으로 추가될 수 있다.

배낭 문제(knapsack problem)는 오직 하나의 제약조건이 있는 문제로 그 문제 형태가 단순하여 단순한 휴리스틱 알고리즘도 매우 잘 작동하며 지금껏 근사최적해(near-optimal solution)를 찾기 위한 효율적인 근사(approximation) 알고리즘에 대한 연구도 많이 이루어져 왔다[2-5]. 이 논문에서는 탐색이 그리 쉽지 않은, 둘 이상의 제약조건이 있는 다중 배낭 문제(multidimensional 0-1 knapsack problem)를 다룬다.

다중 배낭 문제를 해결하기 위한 방법들이 과거에 많이 연구되어 왔으나 [6-13] 이들 연구는 이산 해 공간을 바로 다루는 접근 방식이 대부분이었다. 이 논문에서는 이산 해 공간을 직접 다루는 대신에 문제의 탐색 공간을 실수 공간으로 변환한 다음에 그 변환된 해 공간을 다루어 간접적인 탐색을 수행한다. 다중 배낭 문제는 여러 개의 제약식이 있는 최적화 문제이다. 이 문제는 제약식 개수에 해당하는 만큼의 라그랑지 승수(Lagrange multiplier)들을 사용하여 이산적인 해 공간을 연속적인 실수 공간으로 바꿀 수 있다[14]. 하지만 이런 변환 방법은 원래의 해 공간이 연속 공간이 아니었으므로 많은 한계가 있게 된다. 지금껏 연구된 이산 해 공간을 위한 라그랑지안 방법들은 쌍대성(duality) 정리에 의해 얻은 특징에 기초하여 좋은 상한(upper bound)을 찾는데 주로 이용되어 왔다[15,16]. 하지만 본 논문에서는 라그랑지안 방법으로 좋은 가능 해(feasible solution)를 찾는데 초점을 둔다.

그동안 이산 최적화 문제를 풀기 위한 라그랑지안 기법에 관한 많은 연구가 있어 왔다[17-22]. 이 중에서 다중 배

접수일자 : 2010년 7월 12일
완료일자 : 2010년 11월 20일
+ 교신저자
감사의 글 : 이 논문은 2008년 정부(교육과학기술부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2008-331-C00048).

낭 문제를 풀기 위한 연구도 여럿 있었다. 전형적으로 그런 연구들의 대부분은 제약식들을 쌍대화(dualize)함으로써 분지한계법(branch-and-bound algori- thm)에 사용될 상한 값을 찾는 것이 주된 목적이었다[23]. 이런 이유로 지금까지 연구된 라그랑지안 방법으로 품질 좋은 가능 해를 바로 찾기란 쉽지 않은 일이다. 저자가 아는 한 좋은 가능 해를 찾기 위한 라그랑지안 방법은 Magazine과 Oguz에 의해 제안된 연구 하나 뿐이었다[11]. 최근에 그들의 방법을 유전 알고리즘과 결합하여 성능을 향상시킨 연구도 있었다[24]. 하지만 이 유전 알고리즘 방법은 단지 기존의 라그랑지안 방법을 자신의 알고리즘에서 평가 함수로만 이용했을 뿐이지 새로운 라그랑지안 방법을 제시한 것은 아니었다. 본 논문에서는 Magazine과 Oguz의 결과를 개선한 연구결과를 제시할 것이다.

이 논문의 나머지는 다음과 같이 구성된다. 2절에서는 기존의 라그랑지안 연구에 대해 보다 자세히 검토한다. 3절에서는 논리 전개의 편의상 다중 배낭 문제를 수학적 기호를 사용하여 형식적으로 정의하고 라그랑지안 완화(Lagrangian relaxation)에 대한 기본을 제공한다. 4절에서는 3절에서 제시된 라그랑지안 완화 해법을 기반으로, 새롭게 얻은 라그랑지 승수의 특성을 이용하여 품질 좋은 가능 해를 효율적으로 찾기 위한 새로운 라그랑지안 탐색 방법을 제안한다. 5절에서는 실험 내용 및 그 결과를 분석하고 6절에서는 앞으로의 연구 방향을 제시하며 이 논문을 마무리한다.

2. 관련 연구

이 절에서는 그동안 이산 최적화 문제를 풀기 위해 제안되었던 라그랑지안 방법들에 대해 좀 더 자세히 검토한다. 이산 최적화 문제에 사용되는 라그랑지 승수는 미분이 불가능하다는 성질이 있어서 연속 공간의 최적화 문제에 비해 많은 제한이 있게 된다. 그동안의 연구는 이러한 문제를 극복하려는 시도로 주로 서브그래디언트(subgradient) 알고리즘에 초점을 두어 왔다. 서브그래디언트 알고리즘은 분지한계법을 위한 상한(upper bound)을 생성하기 위해 사용된다. 이러한 방법을 라그랑지안 완화법(Lagrangian relaxation)이라 부른다[19,25]. 라그랑지안 완화법에 대해 더 자세히 알고 싶다면 [26,27]를 참고하면 된다. 서브그래디언트 방법은 수렴할 때까지 반복적으로 수행이 되는데, 각 단계에서 라그랑지 승수의 서브그래디언트의 반대 방향으로 현재의 라그랑지 승수를 이동시킨다. 이러한 탐색의 목적은 오직 라그랑지안 쌍대(dual) 문제의 최적해를 구하는 데에 있으므로 문제의 상한을 구하는 데에는 유용하게 사용할 수 있지만 가능 해를 찾는 일과는 크게 관련이 없다고 할 수 있다. Magazine과 Oguz[11]는 서브그래디언트 알고리즘의 수행 중에 가능 해(feasible solution)가 나올 수 있으므로 이 때 나온 해를 가능 해로 저장하여 출력하는 방법을 고려해 보았으나 그 방법이 처음부터 가능 해를 찾으려고 노력하는 자신들의 방법보다 좋지 않음을 보였다.

라그랑지안 방법을 사용하여 가능 해를 찾는 연구는 Magazine과 Oguz에 의해 이루어 졌다[11]. 저자가 아는 한 지금까지 이 연구가 유일하다. 이 방법은 결정적(deterministic)이며 건설적(constructive) 알고리즘으로 이제부터는 그들의 방법을 MO-CONS라 부르기로 한다. (MO-CONS의 보다 자세한 알고리즘 동작은 부록 A를 참고하기 바란다.) 하지만 이 MO-CONS는 다른 메타 휴리스틱과의 결합 없이는 그다지 만족할 만한 결과를 보이지 못했다. Raidl은 MO-CONS와 유전 알고리즘을 결합한 연구를 하여 많은 성능 개선을 이루어 내었다[24]. Raidl의 연구는 각 물건의 가치 값을 적절히 바꾸어 변형된 문제를 만든 다음에 이 변형된 문제에 MO-CONS를 적용하여 가능 해를 구하는 방식이다. 그의 연구는 단지 MO-CONS를 자신의 방법의 평가함수로만 이용한 방법으로서 라그랑지안 방법 자체만을 따져본다면 큰 공헌이 없다고 할 수 있다.

3. 문제 정의와 라그랑지안 완화

n과 m을 각각 물건의 개수와 제약식의 개수라고 하자. 각 물건 j는 가치 c_j 값을 갖고 각 제약조건 i에 대한 용량 소비값 a_{ij}를 갖는다. 각 제약조건 i는 용량 b_i를 갖는다. 이제 다중 배낭 문제를 수학적으로 형식화하여 표현하면 다음과 같다.

$$\text{Maximize } c^T x$$
$$\text{subject to } Ax \le b,\ x \in \{0,1\}^n.$$

여기에서 $c = (c_1, c_2, ..., c_n)^T$, $A = (a_{ij})_{m \times n}$, $b = (b_1, b_2, ..., b_m)^T$, $x = (x_1, x_2, ..., x_n)^T$이다. A, b, c는 입력으로 주어지고 이들의 각 원소는 음이 아닌 정수이다. 다시 풀어 쓰면 다중 배낭 문제의 목적은 m 개의 제약식 $Ax \le b$을 만족하면서 가중치 합 $c^T x$를 최대화하는 이진 벡터 x를 찾는 것이다.

다중 배낭 문제는 제약식을 갖는 최적화 문제지만 라그랑지 승수 벡터 $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_m)^T$를 사용하면 다음과 같이 제약식이 없는 최적화 문제로 변환이 가능하다.

$$\text{Maximize } c^T x - \lambda^T A x$$
$$\text{subject to } x \in \{0,1\}^n.$$

이 변환된 문제는 연속 공간에서 성립하는 라그랑지 쌍대 정리가 성립하지는 않지만 다른 약간의 좋은 특성을 가지게 된다. 라그랑지 승수 벡터 $\lambda \ge 0$가 주어지면 이를 사용하여 제약식이 없는 변환된 최적화 문제의 해를 쉽게 구할 수 있다. 이에 대한 해법은 그림 1에 있으며 $O(nm)$의 시간 복잡도(time complexity)를 가진다. 이렇게 변환해서 얻은 해 x^*에 대해 성립하는 중요한 성질은 원래 문제의 용량 벡터 b를 $Ax^* (= b^*)$로 바꿀 경우 x^*는 이 변환된 문제에서 최적해라는 사실이 보장된다는 것이다. 만약 적당한 λ에 대해 $b^* \le b$가 된다면 x^*는 가능 해(feasible solution)가 된다. 또한, 운이 좋게도 $b^* = b$가 된다면 x^*는 원래 주어진 문제의 최적해가 되겠지만 이는 좀처럼 발생하지 않는 경우라 기대하기는 어렵다.

4. 제안한 라그랑지안 휴리스틱

이 절에서는 라그랑지 승수 공간의 탐색을 통해 좋은 가능 해를 찾는 새로운 방법을 제안한다. 제안한 방법의 주된 틀은 적당한 라그랑지 승수 벡터를 찾아 그림 1을 적용하여 얻은 새로운 용량 b^*가 원래 주어진 문제 인스턴스의 용량 b에 최대한 가까우면서 어느 하나라도 주어진 용량을 초과

하지 않도록 하는 데에 있다. 이 새로운 방법을 설계하기 위한 중요한 성질로서 저자들은 다음의 정리를 도출하였다.

```
// λ = (λ_1, λ_2, ..., λ_m)^T ≥ 0: input non-negative Lagrange multiplier vector
F(λ) {
    for i = 1 to n
        if c_i > Σ_{j=1}^{m} λ_j a_ji then x_i^* = 1;
        else x_i^* = 0;
    end for
    b^* = Ax^*;
    μ^* = c^T x^*;
    return (μ^*, x^*, b^*);
}
```

그림 1. 라그랑지안 완화의 풀이 알고리즘
Fig. 1. Algorithm for solving Lagrangian relaxation

정리 1. 두 개의 라그랑지 승수 벡터 λ, λ'가 주어져 있는데 여기에서 λ'은 λ와 k 번째 원소만 다른 값이고 ($\lambda_k \neq \lambda_k'$) 나머지 원소는 모두 같다고 가정하자. $(\mu, x, b) = F(\lambda)$이고 $(\mu', x', b') = F(\lambda')$이라고 하자. 그러면 만약 $\lambda_k < \lambda_k'$이면 $b_k \geq b_k'$이고 만약 $\lambda_k > \lambda_k'$이면 $b_k \leq b_k'$이 된다.

증명: $F(\lambda)$와 $F(\lambda')$의 최적성(optimility)에 의해 $c^T x - \lambda^T A x \geq c^T x' - \lambda^T A x'$,
$c^T x' - \lambda'^T A x' \geq c^T x - \lambda'^T A x$이 성립한다. 이 두 부등식을 더하면
$c^T x - \lambda^T A x + c^T x' - \lambda'^T A x' \geq c^T x' - \lambda^T A x' + c^T x - \lambda'^T A x$
이 된다. 이제 양변의 공통적인 원소를 소거하고 -1을 양변에 곱해 부호를 바꾸어주면
$\lambda^T A x + \lambda'^T A x' \leq \lambda^T A x' + \lambda'^T A x$이 된다. $b = Ax$이고 $b' = Ax'$이므로 식을 더 간소화할 수 있다.

즉, $\lambda^T b + \lambda'^T b' \leq \lambda^T b' + \lambda'^T b$이 된다. 가정에 의해 λ와 λ'은 k 번째만 다른 값이므로 전개하여 공통 원소를 소거하면 $\lambda_k b_k + \lambda_k' b_k' \leq \lambda_k b_k' + \lambda_k' b_k$가 되고 이 부등식의 우변을 좌변으로 이항시켜 인수분해하면
$(\lambda_k - \lambda_k')(b_k - b_k') \leq 0$이 되어 증명이 끝난다. □

위 정리에 의해 만약 $b_k^* > b_k$라면 k 번째 라그랑지 승수로 λ_k보다 큰 λ_k'를 선택함으로써 k 번째 제약식에서 b_k^*보다 더 작은 용량을 얻을 수 있다. 이러한 변화는 k 번째 제약식을 만족하게 하거나 초과된 용량을 줄일 수 있게 만든다. 물론 이 연산에 의해 다른 제약조건이 위배될 수도 있다. 이 때 생각해야 할 중요한 문제는 여러 제약조건이 만족되지 않을 때 어느 제약조건에 해당하는 라그랑지 승수를 바꿔야 하느냐 하는 것이다. 또한 바꾼다면 얼마만큼 바꾸어야 하는지도 중요한 문제가 된다.

그림 2는 이 논문에서 제안한 새로운 라그랑지안 방법(NLS)을 보여준다. 제안한 방법은 반복적으로 임의의 하나의 라그랑지 승수를 변화시키는 방법이다. 기존의 MO-CONS는 결정적 알고리즘인데 이는 변화시킬 라그랑지 승수가 매 단계마다 유일하게 결정되기 때문이다. 일단 본 논문에서는 매 단계마다 변화시킬 라그랑지 승수를 임의로 정하게 하는 변형된 MO-CONS(구체적인 내용은 부록 A와 그림 4 참고)를 이용하여 라그랑지 승수 벡터를 초기화하였다. 여기에서 시작하여 정해진 회수 N만큼 다음 단계를 반복 수행한다. 각 단계에서 $b^* \leq b$이면 얻어진 해 x^*가 모든 제약조건을 만족하는 가능 해이므로 지금까지 얻어진 가장 좋은 가능 해와 비교하여 그 값을 갱신한다. 좋은 가능 해를 찾았다 하더라도 더 나은 라그랑지 승수를 찾을 수 있는 가능성은 여전히 존재하므로 여기서 알고리즘을 멈추지는 않는다. 임의의 k 번째의 라그랑지 승수를 조금 감소시킴으로써 b_k^*를 b_k에 조금 더 가깝게 맞추어 더 나은 가능 해를 찾을 수 있는지 살펴보게 된다. 만약 $b^* \leq b$가 아니라면 적어도 어느 하나의 제약조건은 위배되었으므로 그 위배된 제약조건을 맞추는 데 집중한다. 만족하지 않는 제약조건 중 임의의 k 번째의 라그랑지 승수를 조금 증가시켜 위배되었던 해당 제약조건이 만족되게 하거나 위배된 용량의 크기를 줄여 준다. 라그랑지 승수의 변화량은 이러한 반복이 지속됨에 따라 점차 감소하게 하였다. 단순히 겉으로 드러난 알고리즘 절차의 차원에서 서브그래디언트 알고리즘과 비교를 한다면, 본 논문에서 제안된 방법은 각 단계마다 오직 하나의 라그랑지 승수만을 변화시키는 반면 서브그래디언트 알고리즘은 모든 라그랑지 승수들을 어떤 벡터에 비례하는 양 만큼을 바꾸어 준다는 차이점을 가지게 된다. 기존의 연구인 MO-CONS를 변형한 알고리즘의 시간 복잡도는 $O(n^2 m)$이고 (부록 A 참고) 그림 1에 주어진 함수 $F(\lambda)$의 계산 복잡도는 $O(nm)$이므로 제안한 방법의 시간 복잡도는 $O(nm(n+N))$이 된다. 여기에서 N은 반복 횟수이다.

```
// N: the number of iterations
λ ← result of a variant MO-CONS; // see Figure 4
for i = 0 to N−1
    (μ^*, x^*, b^*) = F(λ); // see Figure 1
    I = {i : b_i^* ≤ b_i} and J = {j : b_j^* > b_j};
    if b^* ≤ b then // if x^* is a feasible solution
        Update the best feasible solution;
        Choose a random number k among I;
        λ_k = λ_k − δ_i;
    else
        Choose a random number k among J;
        λ_k = λ_k + δ_i;
    end if
end for
return the best feasible solution;
```

그림 2. 새로운 라그랑지안 탐색 알고리즘 (NLS)
(본 논문의 실험에서는 $N = 30{,}000$, $\delta_i = 1/(i+60)$으로 정했다.)
Fig. 2. The proposed Lagrangian search algorithm

5. 실 험

실험은 OR-Library에 있는 제약식을 30개 가지고 있는 잘 알려진 벤치마크(benchmark) 데이터를 가지고 수행하였다[28]. 이들은 총 90개의 문제 인스턴스(instance)로 구성된다. 30개는 100개의 물건, 30개는 250개의 물건, 나머지 30개는 500개의 물건으로 구성된 문제 인스턴스로 구성되어 있다. 각 문제 인스턴스의 군을 아래처럼 이름 짓기로 한다.

> P$m.n$: m개의 제약식과 n개의 물건을 갖는 문제 인스턴스 군.
> 각 인스턴스 군은 30개의 인스턴스로 구성됨.
> P30.100, P30.250, P30.500의 세 가지 군이 있음.

각 알고리즘의 성능 측정의 척도로 다른 연구에서도 많이 사용되고 있는 다음 식을 사용하였다[12.24.28].

$$100\times(LPoptimum-output)/LPoptimum.$$

여기에서 $LPoptimum$은 정의역을 실수 공간 위로 선형계획(linear programming) 완화된 문제의 최적해 값을 의미한다. 이 $LPoptimum$ 값은 당연히 실제 최적해 보다는 같거나 큰 상한 값이 된다. 이 척도 값은 0 이상 100 이하의 수치이며 작을수록 최적해에 보다 가깝다는 것을 의미한다. 상한 값과 비교하였으므로 실제 최적해를 찾았다 하더라도 0의 값을 갖지 않을 수 있다.

표 1. 벤치마크 데이터에 대한 결과 비교
Table 1. Comparison of Results for Benchmark Data

Instance class	MO-CONS [1] [11] (%)	Our variant of MO-CONS [2] (%)		NLS [2] (%) (our method)	
		Best	Avg	Best	Avg
P30.100	11.93	7.82	14.85	**2.69**	**5.01**
P30.250	8.89	7.21	11.25	**1.89**	**3.65**
P30.500	6.89	6.30	9.37	**1.68**	**3.17**
Average	9.24	7.11	11.82	**2.09**	**3.94**

1. 결정적 알고리즘이므로 여러 번 수행해도 같은 결과가 나옴.
2. 비결정적 알고리즘이며 각각 1,000번씩을 수행하여 결과를 얻었음.

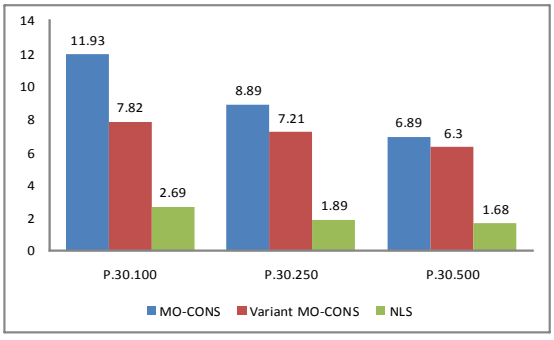

그림 3. 벤치마크 데이터에 대한 결과(표 1)의 막대그래프 (막대 높이가 낮을수록 더 좋다.)
Fig. 3. Bar Chart of results (Table 1) for benchmark data

이 논문에서 제안한 방법 NLS와 가능 해를 찾는데 이용된 기존 라그랑지안 방법의 성능을 비교한다. 비교 대상의 하나는 실행할 때마다 늘 같은 결과를 내는 결정적(deterministic) 알고리즘인 MO-CONS[11]이고 다른 하나는 본 논문에서 변형시킨 비결정적 MO-CONS이다. NLS의 파라미터 N은 예비 실험을 통해 개선이 더 이상 잘 일어나지 않는 시점이 되도록 정하였다(그림 2 하단에 본 논문의 실험에서 설정한 값을 표시하였음). 표 1과 그림 3은 수행한 실험 결과를 보여준다. 각 수치는 30개 인스턴스 결과의 평균치를 의미한다. 변형된 MO-CONS와 NLS는 실행할 때마다 결과가 달라지는 비결정적(non-deterministic) 알고리즘이므로 1,000번씩 수행하여 얻은 결과들의 평균 값(Avg)과 제일 좋은 값(Best)을 구해 통계 내었다. 변형된 MO-CONS는 단일 수행으로는 MO-CONS를 능가하지 못했지만 여러 번 수행하면 MO-CONS보다 더 좋은 결과를 얻을 수 있음을 확인할 수 있었다. 변형된 MO-CONS의 결과를 초기해로 시작하는 NLS는 MO-CONS와 변형된 MO-CONS보다 월등히 좋은 결과를 보여주었다. 절대적 수치만 본다면 라그랑지안 방법에서는 물건의 개수가 많아질수록 성능이 더 좋은 것을 볼 수 있다.

6. 결 론

라그랑지안 방법은 제약식이 있는 최적화 문제에 널리 이용되어 왔다. 다중 배낭 문제와 같은 조합 최적화 문제에서는 보통 좋은 상한 값을 얻는 목적으로만 주로 사용되어 왔는데 이 논문에서는 좋은 가능 해를 찾는 새로운 라그랑지안 방법을 제안하였다. 벤치마크 데이터를 사용한 실험에서는 기존의 라그랑지안 방법들보다 개선된 결과를 얻을 수 있었다. 현재는 제안한 방법의 결과가 다중 배낭 문제를 풀이하기 위해 제안된 최고 성능(state-of-the-art)의 알고리즘[12,28]보다는 성능이 개선되었다고 말할 수 있는 수준이 아니지만 제안된 방법을 다른 방법과 결합함으로써 더 개선할 수 있을 것이다. 예를 들어, 제안된 방법으로 얻어진 가능 해들을 다른 메타 휴리스틱의 초기해로 사용할 수도 있다. 또한 제안된 방법은 MO-CONS를 월등히 개선한 방법이므로 Raidl의 유전 알고리즘[24]에서 MO-CONS 대신에 본 논문에서 제안된 NLS를 사용한다면 더 월등한 성능 향상을 보일 것으로 기대한다. 이런 다른 휴리스틱과의 결합을 통해 더 좋은 결과를 얻을 수 있는 가능성이 많으므로 앞으로 연구할 가치가 충분히 있다.

제안된 라그랑지안 방법은 제약식이 더 많을수록, 물건의 개수가 더 많을수록 더 좋은 성능을 보이는 경향을 가지고 있는데 문제 크기가 커질수록 문제가 더 어려워진다는 일반적인 사실을 깨는 직관적이지 않은 행동 패턴을 보이고 있다. 이러한 특성에 관한 더 자세한 실험적인 관찰과 이에 대한 이론적인 분석을 앞으로의 연구 과제로 남긴다.

본 논문의 연구 결과는 또한 제약조건이 있는 다른 조합 최적화 문제에도 적용될 수 있다. 물론 제약식이 있는 모든 조합 최적화 문제에 적용 가능한 것은 아니다. 라그랑지 승수를 사용하여 완화된 문제의 최적해를 쉽게 구할 수 있는 문제들, 예를 들면 일반화된 할당 문제(generalized assignment problem)나 다중 컨테이너 패킹 문제(multiple container packing problem) 등과 같은 문제는 이미 라그랑지안 방법의 적용이 용이한 문제로 알려져 있으며 이들 문제에 본 논문의 접근 방식과 유사한 라그랑지안 탐색 방법을

사용하면 좋은 가능 해를 얻을 수 있을 것으로 기대한다.

부록 A. MO-CONS의 의사 코드 (pseudo-code)

이 절에서는 기존의 연구인 MO-CONS 알고리즘[11]의 보다 자세한 절차를 알고 싶어 하는 독자를 위해 조금 더 구체적인 정보를 제공한다. 그림 4는 MO-CONS의 방법을 의사 코드로 표현한 내용이다. 알고리즘을 간단히 기술하면 처음에 $\lambda=0$에서 시작하여 배낭에 모든 물건이 담아지도록 한다. 그런 다음 특정 하나의 라그랑지 승수 값(λ_k)[1]을 정확히 하나의 물건이 빠질 때까지 증가시켜 물건 하나를 배낭에서 제거한다. 이러한 물건 제거 작업을 모든 제약식이 만족될 때까지 반복한다. 모든 제약식이 만족된 후에는 배낭에서 빼내었던 물건 중에서 다시 담을 수 있는 물건이 있는지 조사하여 그런 물건이 있으면 다시 배낭에 담는 작업을 수행하게 된다. 이 알고리즘은 결정적(deterministic) 알고리즘으로 수행할 때마다 언제나 같은 결과를 도출하는 방식이며 효율적으로 구현할 경우 시간 복잡도가 $O(n^2m)$이 된다.

$\lambda=(0,0,...,0)$;

$I=\{1,2,...,m\}$; $J=\{1,2,...,n\}$;

repeat

$k=argmax_{i\in I}\Sigma_{j\in J}a_{ij}/b_i$; // in our variant of MO-CONS, we choose a random k.

for each $j\in J$,

$\alpha_j=(c_j-\Sigma_{i=1}^{m}\lambda_i a_{ij})/a_{kj}$;

if $\alpha_j\le 0$, $\alpha_j=\infty$;

$\lambda_k=\lambda_k+\min_{j\in I}\alpha_j$;

$J=J-\{argmin_{j\in J}\alpha_j\}$;

(μ^*,x^*,b^*) = **F**(λ);

until $b^*\le b$;

for all i with $x_i=0$ sorted according to decreasing c_i

if $\Sigma_{k\in I}a_{jk}+a_{ji}\le b_j$ for all $j=1,2,...,m$

$x_i=1$;

end if

end for

return (μ^*,x^*,b^*);

그림 4. MO-CONS의 의사 코드(pseudo-code)
Fig. 4. Pseudo-code of MO-CONS

1) 원래의 MO-CONS에서는 정해진 수식에 따라 k 값이 유일하게 결정되며, 본 논문에서 새롭게 변형한 MO-CONS에서는 임의로 결정된 k 값을 사용한다.

참 고 문 헌

[1] M. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[2] A. Caprara, H. Kellerer, U. Pferschy, and D. Pisinger, "Approximation algorithms for knapsack problems with cardinality constraints," *European Journal of Operational Research*, Vol. 123, pp. 333-345, 2000.

[3] H. Kellerer and U. Pferschy, "A new fully polynomial approximation scheme for the knapsack problem," In *Proceedings of the 1st International Workshop on Approximation Algorithms for Combinatorial Optimization*, pp. 123-134, 1998.

[4] E. L. Lawler, "Fast approximation algorithms for knapsack problems," In *Proceedings of the 17th Annual Symposium on Foundations of Computer Science*, pp. 206-213, 1997.

[5] S. Sahni, "Approximate algorithms for the 0/1 knapsack problem," *Journal of the ACM*, Vol. 22, pp. 115-124, 1975.

[6] C. Chekuri and S. Khanna, "A PTAS for the multiple knapsack problem," *In Proceedings of the Symposium on Discrete Algorithms*, pp. 213-222, 2000.

[7] P. C. Chu, *A genetic algorithm approach for combinatorial optimization problems*. PhD thesis, The management school, imperial college of science, London, 1997.

[8] A. Freville and G. Plateau, "An efficient preprocessing procedure for the multidimensional 0-1 knapsack problem," *Discrete Applied Mathematics*, Vol. 49, pp. 189-212, 1994.

[9] A. M. Frieze and M. R. B. Clarke, "Approximation algorithms for the m-dimensional 0-1 knapsack problem: Worst-case and probabilistic analyses," *European Journal of Operational Research*, Vol. 15, pp. 100-109, 1984.

[10] B. Gavish and H. Pirkul, "Efficient algorithms for solving multiconstraint zero-one knapsack problems to optimality," *Mathematical Programming*, Vol. 31, pp. 78-105, 1985.

[11] M. J. Magazine and O. Oguz, "A heuristic algorithm for the multidimensional zero-one knapsack problem," *European Journal of Operational Research*, Vol. 16, pp. 319-326, 1984.

[12] G. R. Raidl, "An improved genetic algorithm for the multiconstrained 0-1 knapsack problem," In *Proceedings of the IEEE Conference on Evolutionary Computation*, pp. 207-211, 1998.

[13] M. Vasquez and J.-K. Hao, "A hybrid approach for the 0-1 multidimensional knapsack problem," In *Proceedings of the 17th International Joint Conference on Articial Intelligence*, pp. 328-333, 2001.

[14] D. G. Luenberger, *Optimization by Vector Space Methods*. John Wiley & Sons, Inc., 1969.
[15] G. L. Nemhauser and L. A. Wolsey, *Inter and Combinatorial Optimization*. John Wiley & Sons, Inc., 1988.
[16] L. A. Wolsey, *Integer Programming,* John Wiley & Sons, Inc., 1998.
[17] Y.-J. Chang and B. W. Wah, "Lagrangian techniques for solving a class of zero-one integer linear programs," In *Proceedings of the Computer Software and Applications Conference*, pp. 156-161, 1995.
[18] B. Gavish, "On obtaining the 'best' multipliers for a Lagrangean relaxation for integer programming," *Computers & Operations Research*, Vol. 5, pp. 55-71, 1978.
[19] A. M. Geoffrion, "Lagrangian relaxation for integer programming," *Mathematical Programming Study*," Vol. 2, pp. 82-114, 1974.
[20] D. Schuurmans, F. Southey, and R. C. Holte, "The exponentiated subgradient algorithm for heuristic boolean programming," In *Proceedings of the International Joint Conferences on Artificial Intelligence*, pp. 334-341, 2001.
[21] B. W. Wah and Y. Shang, "A discrete Lagrangian-based global-search method for solving satisability problems," In D.-Z. Du, J. Gu, and P. Pardalos, editors, *Satisability Problem: Theory and Applications*, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pp. 365-392. 1997.
[22] B. W. Wah and Z. Wu, "The theory of discrete Lagrange multipliers for nonlinear discrete optimization," *In Principles and Practice of Constraint Programming*, pp. 28-42, 1999.
[23] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Inc., 1990.
[24] G. R. Raidl, "Weight-codings in a genetic algorithm for the multiconstraint knapsack problem," In *Proceedings of the Congress on Evolutionary Computation*, Vol. 1, pp. 596-603, 1999.
[25] R. K. Martin, *Large Scale Linear and Integer Optimization: A Unified Approach*. Kluwer Academic Publishers, 1998.
[26] M. L. Fisher, "The Lagrangian relaxation method for solving integer programming problems," *Management Science*, Vol. 27, No. 1, pp. 1-18, 1981.
[27] J. F. Shapiro, "A survey of Lagrangian techniques for discrete optimization," *Annals of Discrete Mathematics*, Vol. 5, pp. 113-138, 1979.
[28] P. C. Chu and J. E. Beasley, "A genetic algorithm for the multidimensional knapsack problem," *Journal of Heuristics*, Vol. 4, pp. 63-86, 1998.

저 자 소 개

윤유림 (Yourim Yoon)
2003년 : 서울대학교 컴퓨터공학부(학사)
2006년 : 서울대학교 컴퓨터공학부 (박사과정수료)

관심분야 : 최적화 이론, 진화연산, 이산수학, 조합최적화
Phone : 02) 880-1851
E-mail : yryoon@soar.snu.ac.kr

김용혁 (Yong-Hyuk Kim)
1999년 : 서울대학교 전산과학 전공(학사)
2001년 : 서울대학교 컴퓨터공학부(석사)
2005년 : 서울대학교 컴퓨터공학부(박사)
2005년～2007년 : 서울대학교 반도체 공동연구소 연구원
2007년～현재 : 광운대학교 컴퓨터소프트웨어학과 조교수

관심분야 : 최적화, 진화연산, 지식공학
Phone : 02) 940-5212
E-mail : yhdfly@kw.ac.kr