Quadratic 복수 컨테이너 적재 문제에 관한 연구*

여기태** · 석상문***† · 이상욱****

A Study on the Quadratic Multiple Container Packing Problem

Gi-Tae Yeo** · Sang-Moon Soak*** · Sang-Wook Lee****

■ Abstract ■

The container packing problem is one of the traditional optimization problems, which is very related to the knapsack problem and the bin packing problem. In this paper, we deal with the quadratic multiple container packing problem (QMCPP) and it is known as a NP-hard problem. Thus, it seems to be natural to use a heuristic approach such as evolutionary algorithms for solving the QMCPP. Until now, only a few researchers have studied on this problem and some evolutionary algorithms have been proposed.

This paper introduces a new efficient evolutionary algorithm for the QMCPP. The proposed algorithm is devised by improving the original network random key method, which is employed as an encoding method in evolutionary algorithms. And we also propose local search algorithms and incorporate them with the proposed evolutionary algorithm.

Finally we compare the proposed algorithm with the previous algorithms and show the proposed algorithm finds the new best results in most of the benchmark instances.

Keyword : Quadratic Multiple Container Packing Problem, Evolutionary Algorithm, Network Random Key Encoding

논문접수일 : 2009년 05월 09일 논문게재확정일 : 2009년 08월 18일
논문수정일(1차 : 2009년 07월 25일)
* 본 논문은 인천대학교 2008년도 자체연구비 지원에 의하여 연구되었음.
** 인천대학교 동북아물류대학원
*** 특허청 정보심사과
**** 목원대학교 정보통신공학과
† 교신저자

1. 서 론

한국은 세계 5대 컨테이너 처리 물동량을 자랑하는 세계적인 컨테이너 터미널을 보유하고 있는 국가로서 매년 막대한 물량의 컨테이너 화물을 처리하고 있다. 그리고 한국해양수산개발원에 따르면[5] 항만을 통하여 수출·입되는 전체 화물 중 수입기준으로 2011년 47%, 2015년 48%, 2020년 49% 정도가 컨테이너로 운송이 될 것이며, 수출기준으로는 2011년 67%, 2015년 72%, 2020년 74% 정도가 컨테이너로 운송이 될 것으로 전망하고 있다.

또한 현재 수출의 주된 루트는 항만을 통하는 것이 95% 이상을 차지하고 있으며, 특히 항만에서 처리되는 대부분의 상품 처리 수단은 컨테이너를 통해서 이루어지고, 이러한 컨테이너 화물의 운송비용은 일반적으로 개당 처리 비용으로 결정된다. 따라서 화주 입장에서는 많은 물량을 한 컨테이너에 혼재해서 보내는 것이 이득일 뿐만 아니라 컨테이너 처리비용을 줄임으로서 상품의 가격 경쟁력을 높일 수 있는 수단이 된다.

이러한 의미에서 화물을 컨테이너에 어떻게 적재할 것인가는 우리나라와 같이 수출 의존도가 높은 경제 구조를 가진 나라에서는 아주 중요한 문제이다. 또한, 컨테이너에 제품을 적재할 경우 유사한 목적지를 가진 제품들을 동일한 컨테이너에 적재하는 것이 추가로 발생하게 되는 처리비용을 줄일 수 있음은 자명한 사실이다.

따라서 본 연구에서는 다수의 목적지를 가지는 상품들을 컨테이너에 적재할 경우 어떻게 하면 화주가 더 많은 이득을 얻을 수 있는지에 대한 문제를 다룬다.

결국, 이 문제는 기존에 Hiley and Julstrom[4]이 처음으로 제안했던 Quadratic 복수 배낭 문제(Quadratic Multiple Knapsack Problem: QMKP)로 해석될 수 있다. 하지만 복수 배낭 문제(Multiple Knapsack Problem: MKP)는 많은 연구들에서 서로 다른 문제를 지칭하는 것으로 사용되어 오면서 연구자들에게 혼돈을 초래했었다. 따라서 QMKP 또한 MKP로 인해 발생하는 혼돈에서 자유롭지 못하기 때문에 본 논문에서는 이러한 혼돈을 피하기 위해 QMKP 대신 기존의 복수 컨테이너 적재 문제(Multiple Container Packing Problem : MCPP)의 특수한 경우인 Quadratic 복수 컨테이너 적재 문제(Quadratic Multiple Container Packing Problem : QMCPP)라는 용어를 사용한다.

QMCPP는 Quadratic 배낭 문제를 확장한 형태의 문제로 아주 어려운 NP-hard 문제로 분류되기 때문에, 본 논문에서는 이를 해결하기 위해 Rothlauf et al.[7] 이 걸침나무 문제를 해결하기 위해 제안한 네트워크 랜덤 키(Network Random Key : NetKey) 방법을 개선하는 새로운 방법을 사용하는 유전 알고리즘을 제안한다.

NetKey 방법을 사용하는 유전 알고리즘의 경우 다양한 네트워크 문제에 손쉽게 적용이 가능할 뿐만 아니라 유전 연산자의 사용에 있어 제약을 받지 않는다는 장점을 지닌다. 또한 임의로 부여된 랜덤 키에 의해 정렬된 순서에 따라 해를 고려하기 때문에 기존 유전 알고리즘의 단점으로 지적된 해 표현법 내에서의 유전인자들(gene)사이의 연결문제(linkage problem)를 극복할 수 있는 장점이 있다.

하지만, 기존의 NetKey 방법의 경우는 네트워크 상에 모든 가능한 에지를 해 표현법(encoding)을 위해 사용하기 때문에 네트워크의 규모가 증가하게 되면 해의 길이가 노드의 개수만큼씩 길어진다. 또한, NetKey 해 표현법은 다수의 유전형(genotype)이 하나의 표현형(phenotype)을 나타내는 중복성을 지닌 표현법(redundant encoding)에 해당하기 때문에 네트워크의 규모가 증가하면 증가할수록 중복성(redundancy)은 더욱 증가하게 된다. 뿐만 아니라 기존의 NetKey 방법은 위에서 언급한 것처럼 디코딩 방법이 임의로 부여된 랜덤 키의 정렬에 의해 결정되기 때문에 효율적인 휴리스틱 방법을 결합하는 것에 한계를 지닌다.

결국 이러한 해 표현법 자체가 지니는 특성은 네트워크의 규모가 증가할수록 탐색의 효율성을 떨어뜨릴 뿐만 아니라 계산시간도 증가시키고, 성능

의 개선도 어렵게 만든다는 단점으로 작용한다.

따라서, 본 논문에서는 기존 NetKey 방법이 지닌 중복성(redundancy)를 줄이기 위해 새로운 인코딩 및 디코딩 방법을 사용하며, 성능을 향상시키기 위해 휴리스틱 방법을 결합하는 새로운 방법을 제안한다. 비록 제안하는 방법이 해의 개선을 위해 사용하는 휴리스틱 방법으로 인해 더 많은 계산 시간을 필요로 하지만 제안하는 방법이 기존의 NetKey 방법보다 성능면에서 훨씬 더 우수한 결과를 찾을 수 있음을 비교 실험을 통해 보인다. 이와 더불어 기존에 QMCPP를 위해 개발된 방법들과의 비교를 통해 제안하는 방법의 성능을 보이는 것을 본 연구의 주 목적으로 하고 있다.

본 논문의 구성은 다음과 같다. 제 2장에서 정형화된 QMCPP 문제를 소개하고, 제 3장에서는 기존에 QMCPP를 위해 제안된 방법론들에 대해 알아보고, 제 4장에서 제안하는 알고리즘에 대해 설명한다. 제 5장에서는 벤치마크(benchmark)문제에서 제안하는 알고리즘이 찾아낸 해와 기존에 방법들이 찾아낸 해들과의 비교 분석을 수행하고, 제 6장에서 결론을 맺는다.

2. Quadratic 복수 컨테이너 적재 문제

QMCPP를 간단하게 설명하면, 컨테이너에 적재되기 위해 대기 중인 무게(w_i)와 가치(v_i)가 미리 알려진 복수의 상품과 상품을 적재하기 위해 용량 제약(c_j)을 가지는 복수의 컨테이너가 존재하며, 여기에 동일한 목적지를 가지는 상품을 동일한 컨테이너에 적재할 경우 추가적인 상품처리 비용의 해소로 인한 잠재이익(v_{ij})이 발생하게 되는데 이를 함께 고려한다. 결국, 이 문제는 각 컨테이너에 적재되는 상품의 총 가치(v_i)와 동일 목적지의 상품을 동일 컨테이너에 적재할 경우 얻게 되는 잠재이익(v_{ij})의 합이 최대가 되는 적재계획을 세우는 문제로 해석이 가능하다.

QMCPP를 모형화하면 다음과 같다.

$$Maximize\ f=\sum_{i=1}^{n}\sum_{j=1}^{C}v_i x_{ij}+\sum_{i=1}^{n-1}\sum_{k=i+1}^{n}\sum_{j=1}^{C}x_{ij}x_{kj}v_{ik} \quad (1)$$

$$subject\ \ to \sum_{i=1}^{n} w_i x_{ij} \le c_j,\ \ j=1,\cdots,C \quad (2)$$

$$\sum_{j=1}^{C} x_{ij} \le 1\ ,\ \ j=1,\cdots,n \quad (3)$$

$$x_{ij} \in \{1,\ 0\} \quad (4)$$

여기서, $w_j>0,\ v_j>0,\ c_j>0$이다.

위 모형에서 x_{ij}는 의사결정변수(decision variable)로 상품 i가 컨테이너 j에 적재되면 1의 값을 가지며 그렇지 않을 경우 0의 값을 가진다. 따라서 제약식 (3)은 각 상품이 나누어져서 적재될 수는 없음을 나타내는 제약식이며, 제약식 (2)는 컨테이너 j에 적재되는 상품들의 총 무게의 합이 컨테이너 j의 최대 적재가능량 c_j를 초과할 수 없음을 나타내는 제약식이다.

3. 기존 연구 방법

QMCPP는 Hiley and Julstrom[4]에 의해 처음 소개되었으며, Hiley와 Justrom은 QMCPP를 해결하기 위해 상대적 가치 밀도(relative value density)라는 개념을 사용하는 3가지 방법을 제안하였다. 상대적 가치 밀도는 다음과 같이 정의된다.

$$vd(i,\ S)=(v_i+\sum_{j\in S}v_{ij})/w_i \quad (5)$$

즉, 상대적 가치 밀도는 집합 S에 포함되어 있는 상품 i와 다른 상품들과의 잠재이익(v_{ij})의 합과 상품 i의 가치를 더한 값을 상품 i의 무게(w_i)로 나눈 것으로, 이는 결국 집합 S에 포함되어 있는 상품 i가 가지는 상대적인 총 가치를 나타낸다.

Hiley와 Julstrom이 제안한 첫 번째 방법은 탐욕적 휴리스틱 방법(greedy heuristic)으로, 초기에 전

체 상품의 집합 S에 대한 각각의 상품들의 상대적 가치 밀도를 계산하여 가장 큰 상대적 가치 밀도를 가지는 상품을 첫 번째 컨테이너에 적재하고 그 다음부터는 첫 번째 컨테이너에 적재된 상품과 함께 적재될 경우 가장 상대적 가치 밀도가 높은 상품을 선택하는 방식으로 컨테이너를 채워 나간다. 그리고 한 컨테이너가 더 이상 채워질 수 없다면 다른 컨테이너를 동일한 방식으로 채워나가는 방식으로 해를 찾는다.

두 번째 방법은 유전 알고리즘(Genetic Algorithm: HJ-GA)을 이용하는 방법인데, 우선 n의 길이를 가지는 각각의 해(chromosome) $c[\cdot]$를 $\{0, 1, 2, \cdots, k\}$ 중 하나의 값으로 각 유전인자(gene)를 할당한다. 즉, $c[i] = k$는 상품 i가 컨테이너 k에 적재됨을 의미하고, $c[i] = 0$는 상품 i가 어떤 컨테이너에도 적재되지 않음을 의미한다. 특히, 각각의 해를 생성할 때 컨테이너의 최대 적재가능량 c_j를 고려하여 모든 제약식을 만족하도록 생성한다. 그리고 여기에 특별히 고안된 교차 연산자(crossover operator)와 변이 연산자(mutation operator)를 적용하였다.

마지막으로 세 번째 방법은 확률적 언덕 오르기 기법(Stochastic Hill-Climber : HJ-SHC)으로 두 번째 방법처럼 n의 길이를 가지는 해를 생성하고, 여기서 두 번째 방법에서 고안된 휴리스틱 변이 연산자(heuristic mutation operator)를 적용하여 현재 해의 이웃해를 생성하고, 이를 정해진 횟수만큼 반복하는 방식으로 탐색공간을 탐색한다.

위 세 가지 방법들에 대한 비교 실험에서 확률적 언덕 오르기 기법이 평균적으로 가장 우수한 성능을 보였으며, 다음으로 유전 알고리즘, 탐욕적 휴리스틱 방법 순으로 우수한 성능을 보여 주었다. 특히나 유전 알고리즘의 경우는 컨테이너의 개수가 작은 문제들에서 더 좋은 성능을 보였으며, 컨테이너의 개수가 증가할수록 성능이 급격하게 떨어지는 특성을 보여 주었다.

QMCPP를 위한 또 다른 접근법은 Singh and Baghel[9]에 의해 제안된 안정상태 해집단 대체 전략(steady-state population replacement method)을 가지는 그룹핑 유전 알고리즘(Grouping Genetic Algorithm : SB-GGA)이다. SB-GGA는 우선 각각의 해(chromosome)를 Hiley와 Julstrom이 제안한 탐욕적 휴리스틱 방법(greedy heuristic)을 변형한 방법을 이용하여 해집단을 생성한다. 구체적으로, 각 컨테이너의 첫 번째 상품을 가장 상대적 가치 밀도 값이 큰 상품을 선택하는 것이 아니라 랜덤하게 선택하고 아직 적재되지 않은 상품들 중에서 상대적 가치 밀도 값이 가장 큰 상품들을 선택함으로써 컨테이너를 채워나간다. 그리고 이렇게 생성된 해가 해집단 내에 동일한 해가 있는지를 확인하고 없을 경우에만 해집단에 포함시킨다. 마지막으로 이렇게 생성된 해집단에 특별하게 고안된 교차연산자(crossover operator)와 두 가지 다른 변이 연산자(mutation operator)를 적용하여 탐색공간을 탐색한다.

Singh와 Baghel는 그들이 개발한 SB-GGA와 Hiley와 Julstrom이 제안한 HJ-SHC와 HJ-GA와 동일한 문제들에서 비교실험을 수행하여, SB-GGA가 가장 좋은 해를 찾을 뿐만 아니라 평균적인 해의 성능에서도 HJ-GA 및 HJ-SHC를 능가함을 보여주었다. 특히, SB-GGA를 통해 얻은 평균 해의 성능의 경우는 HJ-GA 및 HJ-SHC의 평균 해의 성능보다 항상 더 우수한 결과를 보여주었다.

또 다른 QMCPP를 해결하기 위한 시도는 Sarac and Sipahioglu[10]에 의해서 이루어졌다. Sarac와 Sipahioglu 또한 유전 알고리즘(SS-GA)을 사용하였다. SS-GA에서는 Hiley와 Julstrom이 제안한 HJ- GA를 사용한 해 표현법을 사용하였으며, 성능을 개선하기 위해 특별히 고안된 교차 연산자 및 변이 연산자를 사용하여 비교실험을 수행하였다.

이들 또한 Hiley와 Julstrom의 실험결과와 비교실험을 수행하였는데, SS-GA가 몇몇 문제들에서 알고리즘들 간의 가장 우수한 해들 사이의 차이가 20% 이상 좋은 해를 찾으며, 또한 SS-GA가 컨테이너의 개수가 3 또는 5개인 d = 0.75인 문제보다는 d = 0.25인 문제들에서 더욱 좋은 성능을 보인다는 것을 보여주었다. 여기서, d = 0.25와 d = 0.75는 각

〈표 1〉 랜덤 키 해 표현법의 예

Edge Number	1	2	3	4	5	6
Item-Container	1-1	1-2	2-1	2-2	3-1	3-2
Random Keys	0.36	0.46	0.62	0.89	0.28	0.21
Descending Order	4	3	2	1	5	6

상품의 가치 v_i와 잠재이익 v_{ij}가 0이 아닌 비율에 따라 벤치마크 문제들을 분류한 것으로, d = 0.25라는 것은 해당 벤치마크 문제에서 v_i와 v_{ij}의 값 중 25%는 0이 아님을 의미한다. 따라서 이 수치가 낮을수록 상품들 사이의 잠재이익 v_{ij}의 중요도가 낮아진다.

4. 개선된 네트워크 랜덤 키 방법을 이용한 유전 알고리즘

4.1 네트워크 랜덤 키 방법

유전 알고리즘을 설계할 때 가장 중요한 부분은 다루는 문제에 적합한 해 표현(encoding)을 찾는 것과 이를 이용해서 우수한 새로운 해를 찾도록 하는 합당한 연산자를 선택하는 것이다[2]. 왜냐하면, 사용되는 해 표현법과 이에 적합한 연산자들에 따라 다루는 문제의 탐색공간(search space)을 효율적으로 탐색할 수 있을 뿐만 아니라 결국 이를 통해 알고리즘의 성능이 좌우되기 때문이다.

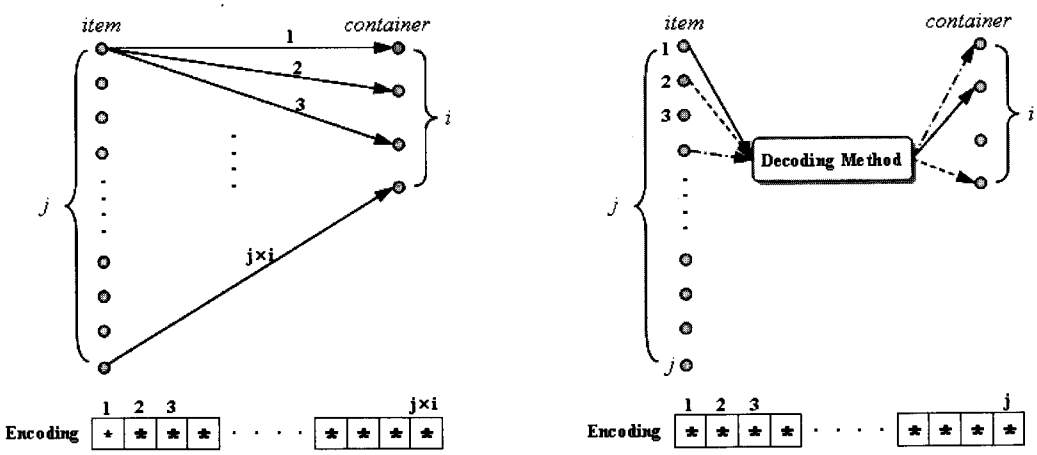
본 논문에서 제안하는 알고리즘은 Rothlauf et al.[7]이 제안한 네트워크 랜덤 키 표현법(Network Random Key Representation : NetKey)을 기반으로 하고 있다. NetKey 표현법의 경우 기존에 스케쥴링 문제(scheduling problem)나 트리 네트워크 최적화 문제 등 다양한 문제들에 적용되어 그 유효성이 입증되어왔다[2, 7].

NetKey 표현법의 가장 큰 장점은 우선 유전 연산자의 사용에 유연하다는 것이다. 이는 일반적으로 TSP 문제를 다룰 경우 주로 사용되는 치환 표현법(permutation encoding)처럼 해 표현법 자체가 가지는 제약이 없기 때문이다. 또한, 기존에 일반적인 유전 알고리즘이 가지고 있는 가장 큰 단점으로 지적된 해 표현법 내에서 우수한 유전 인자들(schema) 끼리의 연결문제(linkage problem)를 해소할 수 있다는 것이다. 즉, 이는 랜덤 키 표현법이 임의로 부여된 키 값의 정렬에 의해서 해를 생성하기 때문에 유전 인자들끼리의 이웃관계는 해 표현 내에서 아무 의미가 없어지기 때문이다.

NetKey 표현법은 초기에 모든 가능한 에지에 번호를 부여하고 그 번호를 이용해서 해당되는 에지를 가려낸다. 그리고 각각의 에지에[0, 1] 사이의 임의의 실수 값(random key)을 할당하고, 이 임의 값에 따라 전체 에지를 정렬하고 정렬된 순서에 따라 해를 형성해 나간다. 또한, 이러한 랜덤 키 값들의 교환을 위해서 유전 연산자들을 이용한다.

<표 1>은 랜덤 키를 사용해서 형성한 간단한 적재계획의 예이다. 3개의 상품과 2개의 컨테이너가 존재하는 경우를 고려해 보면 총 적재 가능한 경우의 수는 3 × 2로 총 6개의 가능한 에지가 존재한다. 결국, 총 해의 길이는 이 에지들의 개수인 6이 되고 각 상품에 랜덤 넘버를 발생시킨다. 그 다음 이 랜덤 넘버를 내림차순으로 정렬한 순서에 따라 각 에지들을 적재 계획을 위해 차례대로 고려해 나간다. 즉, 여기서는 4번 에지가 가장 먼저 컨테이너에 적재를 위해 고려되고 다음으로 3, 2, 1, 5, 6번 에지 순으로 고려된다.

결국, 기존의 랜덤 키 방법의 경우 어떤 상품이 어떤 컨테이너에 적재되는지에 대한 정보를 담고 있는 에지를 기반으로 하기 때문에 특별한 디코딩 방법을 필요로 하지 않는다. 즉, 제일 먼저 고려되는 에지 4는 2번 상품을 2번 컨테이너에 적재시킨다는



[그림 1] NetKey 방법과 개선된 NetKey 방법의 비교

정보를 담고 있으므로, 이 적재계획이 제약식을 만족한다면 선택을 하게 되고 그렇지 않을 경우에는 다음 에지 3을 고려하게 된다. 이렇게 모든 에지를 고려하고 나면 모든 상품에 대한 하나의 적재계획이 생성되게 된다.

4.2 개선된 Network Random Key 방법

서론에서도 언급한 바와 같이 NetKey 방법의 경우 네트워크상에 존재하는 모든 가능한 에지를 해 표현으로 사용하기 때문에 네트워크 내에 노드가 하나씩 증가할 때마다 노드의 개수만큼 해의 길이가 길어진다. 결국, 이는 탐색공간의 범위를 과도하게 확장시킴으로서 탐색 성능을 급격하게 떨어뜨린다. 따라서 본 논문에서는 이러한 단점을 개선할 수 있는 방법을 제안한다.

제 4.1절에서 일반적인 NetKey 방법을 설명하기 위해 간단한 예를 소개하였다. 여기서 NetKey 방법을 위해 사용한 것이 각 상품의 적재 가능한 총 경우의 수 즉, 총 가능한 에지를 기초로 해 표현을 하였다. 하지만, 이럴 경우 상품이 한 개 증가할 때마다 에지의 개수는 컨테이너의 개수인 C개 증가하고 컨테이너가 한 개 증가할 때마다 에지의 개수는 상품의 개수인 n개 증가하게 된다.

하지만, 일반적으로 QMCPP에서는 다음의 2가지 경우를 가정한다.

(1) 상품의 중량 w_i는 컨테이너의 최대 적재 가능량 c_j를 초과하지 않는다 : $w_i \leq c_j$, $i = \{1, \cdots, n\}$, $j = \{1, \cdots, C\}$

(2) 컨테이너의 개수 C는 상품의 개수 n보다 작다 : $1 \leq C < n$

(1)의 경우는 적재가 불가능한 상품은 고려할 필요가 없기 때문이며, (2)의 경우는 각 상품이 각 컨테이너에 개별적으로 적재되는 경우를 배제하기 위한 것이다. 결국, 일반적인 경우에서는 상품의 개수가 컨테이너의 개수보다 훨씬많은 경우가 대부분이며, 컨테이너 개수와 상품의 개수가 비슷해질수록 문제는 점점 단순해진다.

따라서, 본 논문에서 제안하는 방법은 해 표현을 위해 상품만을 고려하는 방법을 제안한다. [그림 1]은 기존의 NetKey 방법과 제안하는 개선된 NetKey 방법을 보여준다. 여기서 i는 컨테이너의 개수, j는 상품의 개수를 각각 나타낸다. 왼쪽의 그림은 기존의 방법이고, 오른쪽의 그림은 개선된 방법이다. 그림에서 확인할 수 있는 바와 같이 해 표현법 자체의 길이가 기존의 $j \times i$에서 j로 줄어듦을 알 수 있다. 하지만, 기존의 방법의 경우는 각각의 에지가 어떤 상품을 어떤 컨테이너에 적재할 것인가에 대한 정보를 담고 있었지만, 개선된 방법에서는 단순

히 어떤 상품을 먼저 고려할 것인가에 대한 정보만을 담고 있어 선택된 상품을 어떤 컨테이너에 적재할 것인가에 대한 적절한 디코딩 방법이 필요하다.

본 논문에서는 디코딩 방법으로 Raidl[6]이 복수 컨테이너 적재 문제(the multiple container packing problem : MCPP)를 위해 제안했던 방법을 사용한다.

즉, 상품이 적재되었을 경우 컨테이너의 최대 적재 가능량을 초과하지 않으면서 가장 작은 여유 공간이 남는 컨테이너를 찾아 이 컨테이너에 상품을 적재하는 방식으로 모든 상품들을 컨테이너에 채워 나간다.

그리고, 제안하는 알고리즘 역시 기존의 NetKey 방법을 이용하는 유전 알고리즘들처럼 연산자 사용에 어떤 제약도 없다. 따라서 기존의 랜덤 키 표현법에서 가장 우수한 결과를 주는 것으로 알려진 균등 교차 연산자(uniform crossover)와 상호 교환 변이연산자(reciprocal exchange mutation)를 사용하였다[11].

균등 교차 연산자는 임으로 해의 길이만큼 0 또는 1의 랜덤 넘버를 발생시키고 각 유전자좌(locus)의 랜덤 넘버 값이 0이면 부모 1의 동일 유전자좌의 유전인자 값을 자손의 유전인자 값으로 취하고, 1이면 부모 2의 동일 유전자좌의 유전인자 값을 자손의 유전인자 값으로 취하는 방식으로 자손을 생성한다. 상호 교환 변이 연산자는 임으로 선택된 두 유전 인자의 값을 서로 교환하는 방식으로 변이를 일으킨다.

그리고 선택 전략으로는 실세계 토너먼트 선택전략[1]을 사용하였는데, 이는 스포츠 게임에서 이루어지는 토너먼트 방식을 모사한 방법으로 해 집단 내에 있는 모든 해들은 서로 이웃하는 해들과 쌍을 형성해서 경쟁을 하게 되고, 경쟁에서 이긴 해들은 다시 이웃하는 승자와 쌍을 형성하여 경쟁을 해나가는 방식으로 반복해서 경쟁을 해 나간다. 결국 각 경쟁에서 한번이라도 승리한 해들을 이용하여 새롭게 해집단을 형성해 나가는 방식이다.

4.3 지역 탐색법(local search algorithm)

본 논문에서는 제안하는 방법을 통해 구해진 해를 보다 더 개선시키기 위한 두 가지 지역 탐색법을 제안한다.

4.3.1 적재된 상품들의 교환을 통한 해 개선법 (Heuristic 1)

첫 번째 휴리스틱 방법은 각각 서로 다른 컨테이너에 적재되어 있는 상품들끼리 교환을 통해 제약조건을 만족하면서 목적함수를 개선할 수 있는지를 확인하여, 만약 목적함수를 개선할 수 있다면 선택된 두 상품을 서로 교환하는 방법으로 해를 개선시키는 것이다. 이는 결국 적재된 상품들 간의 이동을 통해 더 우수한 해를 찾는 방법으로, 상품들 간에 더 많은 상호 관련성을 가지는 즉, v_{ij}가 v_i보다 상대적으로 중요도가 높은 문제들에서 더 효과적인 방법이다.

4.3.2 적재된 상품과 적재되지 않은 상품과의 교환을 통한 해 개선법(Heuristic 2)

두 번째 휴리스틱 방법은 컨테이너에 적재되어 있는 상품과 적재되지 않은 상품끼리 교환을 통해 목적함수를 개선할 수 있는지를 확인하여, 만약 목적함수를 개선할 수 있다면 선택된 두 상품을 서로 교환하는 방법으로 해를 개선시키는 방법이다.

[그림 2]는 제안하는 방법의 전체 절차를 나타낸다.

5. 실험 결과 비교

본 논문에서는 기존의 연구들이 벤치마크(benchmark)문제[12]로 사용한 문제를 보다 정확한 비교실험을 위해 사용하였다. 그리고 벤치마크문제는 v_i와 v_{ij}값이 0이 아닌 비율에 따라 d = 0.25와 d = 0.75로 크게 2가지 그룹으로 나누어지며, 각 컨테이너의 최대 허용가능 용량은 기존의 연구들처럼 모든 상품의 용량(w_j)의 합의 80%를 컨테이너의 개수 C로 나눈 값으로 설정하였다.

$$Capa = (0.8\sum_{i=1}^{n} w_i)/C$$

Step 1 : 초기화 단계
모든 유전인자 $gene[i][j] \leftarrow rand()$ 및 각 컨테이너 t에 적재되는 상품을 나타내는 집합 $C_t \leftarrow \{\phi\}$ 초기화.

Step 2 : 적재계획 생성 및 평가 단계
$A \leftarrow E$ (E는 모든 가능한 상품집합);
for i = 1 to *pop_size* do
Step 2.1 적재계획 생성
while $A \neq \{\phi\}$ do
집합 A에서 가장 낮은 $gene[i][j]$ 값을 가진 상품 j를 선택;
$A \leftarrow A - \{j\}$;
$k \leftarrow BestFit(j)$;
$C_k \leftarrow C_k \cup \{j\}$;
Step 2.2 : 휴리스틱 1 및 휴리스틱 2 적용
Step 2.3 : 식 (1)을 이용해서 적합도 평가

Step 3 : 유전 연산자 적용 단계
Step 3.1 : 균등교차 연산자(Uniform Crossover)
Step 3.2 : 상호 교환 변이 연산자(Reciprocal exchange mutation)
Step 3.3 : 실세계 토너먼트 선택 전략(Real World Tournament Selection)

Step 4 : 종료 조건을 만족하면 종료, 그렇지 않으면 Step 2로 이동.

i : 해집단의 크기, j : 상품의 총 개수, t : 컨테이너의 총 개수
$rand()$: 램덤 넘버 발생 함수
$BestFit(j)$: 상품 j가 적재될 경우 가장 작은 여유 공간을 가진 컨테이너 k를 반환하는 함수

[그림 2] 제안하는 알고리즘의 절차

〈표 2〉 실험 조건

Population Size	100
Initialization	Random
Selection	Real World Tournament Selection
Crossover	Uniform(100%)
Mutation	Swap(100%)
Termination Condition	No improvement for 1,000 generation

<표 2>는 본 논문에서 제안하는 개선된 NetKey 방법을 사용하는 유전 알고리즘을 위한 실험 조건을 보여준다. 기존의 연구들에서는 종료 조건으로 200번의 반복회수(generation limit)를 사용하였지만, 일반적으로 이 조건은 유전 알고리즘이 수렴하기에는 충분하지 않을 뿐만 아니라 제안하는 방법 역시 이 조건에서 충분히 수렴하지 못하였다. 그리고 본 논문의 목적이 기존의 방법들이 찾아낸 해

보다 우수한 해를 찾는 것이기 때문에 기존의 방법과 달리 제안하는 방법이 충분히 수렴할 수 있도록 1,000세대(generation) 동안 해의 개선이 없을 경우 종료시키도록 종료조건을 설정하였다. 하지만, 기존의 방법들과의 직접적인 비교를 위해 200세대에서의 결과도 함께 보인다.

그리고 각 알고리즘은 C를 이용해서 프로그램 되었으며 Pentium IV 3.4 Ghz, 1GByte Ram 컴퓨터 상에서 실험을 하였다.

<표 3>과 <표 4>는 실험 결과를 보여준다. 각 표에서 HJ-SHC와 HJ-GA는 Hiley and Julstrom [4]의 실험 결과이고, SB-GGA는 Singh and Baghel [9]의 실험 결과이며, SS-GA는 Sarac and Sipahioglu [10]의 실험 결과이다. SS-GA의 경우는 일부의 문제들에서만 실험을 수행하여 결과가 없는 부분은 "-"로 표시하였다.

또한, 여기서 밑줄로 강조된 부분은 200세대에서 가장 우수했던 결과를 나타내며, 굵은 글씨로 강조된 부분은 최종 결과(Final)에서 가장 우수했던 결과를 나타내고 있다.

우선 NetKey 방법을 사용하는 유전 알고리즘과 개선된 NetKey 방법을 사용하는 유전 알고리즘 사이의 성능비교를 보면, d = 0.25인 경우뿐만 아니라 d = 0.75인 경우 모두에서 제안하는 개선된 NetKey 방법을 사용하는 유전 알고리즘이 훨씬 우수한 성능을 보이는 것을 확인할 수 있다. 특히, 기존의 NetKey 방법을 사용하는 유전 알고리즘의 최종 결과(Final)와 제안하는 개선된 NetKey 방법을 사용하는 유전 알고리즘의 200세대(200 generation)의 결과를 비교해 보면, 세 경우(d = 0.25의 200-5-5, 200-10-1, 200-10-3 문제)를 제외하고 모든 경우에서 개선된 유전 알고리즘이 더 우수한 성능을 보였다. 또한 최종 결과들끼리 비교한 결과 d = 0.25인 경우 최소 5.71%, 최대 21.99%, 평균 11.79%, d = 0.75인 경우 최소 4.8%, 최대 21.57%, 평균 10.52%의 해의 개선을 달성하였다. 이는 제안하는 방법이 사용하는 인코딩 및 디코딩 방법을 통해 기존의 NetKey 방법에 비해 해의 중복성(redundancy)을 줄이고, 기존의 NetKey 방법에서는 적용하기 어려웠던 효과적인 지역탐색 휴리스틱 기법을 결합한 결과로 판단된다. 하지만, 이렇게 결합된 휴리스틱 기법 때문에 제안하는 방법이 기존의 NetKey 방법에 비해 더 많은 계산시간을 필요로 하였다.

다음은 기존의 연구 결과들과 제안하는 방법의 결과를 비교해 보면, d = 0.25의 경우는 30개의 문제 중에서 11개의 문제에서 제안하는 방법이 200세대에서 가장 우수했던 결과를 보여주었으며, d = 0.75의 경우에는 19개의 문제에서 제안하는 방법이 200 세대에서 가장 우수했던 결과를 보여주었다. 또한 최종 결과를 비교해 보면, d = 0.25의 경우는 25개의 문제에서 d = 0.75의 경우는 모든 문제에서 제안하는 방법이 기존에 찾았던 해 보다도 더 우수한 해를 찾아내었다.

그리고 d = 0.25에 비해 d = 0.75에서 기존의 해를 훨씬 더 개선시켰으며 이는 제안하는 방법이 상품들 상호간의 의존도가 높은 문제들에서 더 우수한 성능을 보여준다는 것을 의미한다. 이는 해의 개선을 위해 사용한 지역탐색 휴리스틱 기법의 특성 때문이다. 또한, 표준편차(STD) 역시 대부분의 경우에서 최종 평균 결과 대비 2%를 넘지 않았는데, 이는 제안하는 방법이 언제나 안정된 결과 값을 줄 수 있음을 의미한다.

이에 반해서, 제안하는 방법은 기존의 방법들에 비해서도 계산시간 면에서는 훨씬 더 많은 계산시간을 소모하였는데, 이는 위에서도 언급한 바와 같이 사용하는 디코딩 방법과 지역탐색 휴리스틱 기법 때문이다. 하지만, 컨테이너 적재문제는 아주 급박하게 처리를 해야 하는 문제가 아닐 뿐만 아니라 컨테이너에 적재되는 상품이 많을수록 화주입장에서는 이득이기 때문에 기존의 방법과 비교해 제안하는 방법이 더 매력적일 수 있을 것으로 판단된다.

6. 결 론

본 논문에서는 Quadratic 복수 컨테이너 적재문제(QMCPP)를 해결하기 위한 효율적인 유전 알고

〈표 3〉 $d = 0.25$인 실험문제들에서의 성능비교

Instance				HJ-SHC		HJ-GA		SB-GGA		SS-GA	Original NetKey			Improved NetKey						
											Best		Final	Best		Avg.		Final		
N	C	Num	Capa	Best	Time	Best	Time	Best	Time	Best	200	Final	Time	200	Final	200	Final	STD	Gen.	Time
100	3	1	688	28144	1.49	28665	0.97	28798	1.78	28807	26261	28192	24.8	30153	31412	29659.0	30963	480.2	2159.0	240.0
		2	738	26915	1.61	28059	1.00	28036	1.75	28456	25244	27027	20.4	27919	29357	27604.0	28984.4	438.3	1976.2	262.6
		3	663	25945	1.45	26780	0.95	26936	1.75	26754	24193	26647	32.6	27762	28927	27259.6	28658.4	198.3	2653.8	310.6
		4	804	27109	1.67	28199	1.02	28418	1.77	28383	26291	27798	28.4	28884	29387	28314.8	29113.8	188.8	2728.0	342.6
		5	723	26288	1.58	27550	1.00	27617	1.72	27582	24808	27527	21.2	28618	29925	28343.6	29603.0	196.9	2702.2	305.6
100	5	1	413	21584	2.00	21914	1.20	22038	1.27	22039	17559	20797	58.8	22809	23616	21944.2	23411.2	192.2	2409.8	201.6
		2	442	20394	2.11	21216	1.25	21459	1.30	21249	17234	20797	54.2	21300	22331	20751.2	21922.6	283.8	2113.4	176.6
		3	398	19454	1.99	20243	1.17	21012	1.24	20862	17084	19773	65.4	20306	21233	19863.6	20982.8	150.0	2120.2	171.2
		4	482	20173	2.30	21698	1.31	21987	1.26	21601	17867	20731	59.0	21399	21946	20804.8	21839.6	80.5	1967.4	164.8
		5	434	19392	2.12	20808	1.25	21057	1.25	20928	16981	19874	60.0	21043	21919	20549.4	21679.2	357.4	2228.8	172.6
100	10	1	206	15232	3.86	13521	2.19	15663	1.49	15778	11084	14112	134.4	15361	16672	15166.4	16153.8	342.8	2322.6	137.0
		2	221	14210	4.11	12859	2.31	15002	1.44	14835	10324	13903	185.6	13934	14941	13625.8	14688.2	217.2	2123.8	113.8
		3	199	13334	3.88	11790	2.15	14231	1.41	14348	10669	13071	155.0	14001	14506	13455.4	14319.0	119.3	2321.2	124.6
		4	241	14321	4.36	13316	2.44	15979	1.43	15495	11376	14167	128.0	14658	15703	14033.4	15217.6	326.9	2476.6	126.6
		5	217	13405	4.14	11909	2.25	14510	1.45	14770	10281	12920	143.0	13629	14806	13261.0	14505.8	258.4	1924.2	102.6
200	3	1	1381	99232	4.54	97469	3.24	99753	5.29	99853	88156	98045	124.2	100844	107363	99516.8	105324.2	1647.4	2817.6	2230.4
		2	1246	106730	4.38	106162	3.13	107475	5.46	104277	88211	103344	131.2	109617	116202	105525.0	115374.4	1028.3	3502.4	2689.4
		3	1335	103529	4.56	101291	3.21	103607	5.36	-	87977	100979	139.2	104074	110178	101477.2	108474.8	2010.5	2834.2	2356.6
		4	1413	97407	4.55	95649	3.29	98276	5.14	97700	85371	94460	150.2	98961	104745	95325.2	102875.2	1853.7	2988.6	2447.8
		5	1358	100827	4.46	99458	3.25	101463	5.39	98326	87231	97214	104.2	102124	107586	98383.4	105692.4	1938.7	3131.4	2615.4
200	5	1	828	72277	5.77	70731	3.70	73040	3.41	73619	55212	67712	199.8	71609	76660	67701.0	74666.0	2695.4	2599.6	1387.8
		2	747	77551	5.38	76297	3.51	78428	3.66	74883	57089	73621	275.0	80124	83771	75598.8	83156.8	746.2	3630.2	1985.4
		3	801	75409	5.60	74377	3.69	76321	3.55	-	55499	70064	278.2	74635	79131	69155.4	76051.6	2645.3	2370.6	1230.0
		4	848	71307	5.87	70264	3.83	71964	3.40	71936	55937	67135	244.4	68655	75174	66598.4	74266.2	1276.1	2512.4	1466.0
		5	815	74287	5.79	72745	3.72	74936	3.54	73825	54532	69209	289.0	67795	76770	67042.2	74911.6	1671.7	3001.6	1351.6
200	10	1	414	48006	10.16	42016	5.77	49212	3.03	48119	30464	41772	465.0	41605	47800	40714.8	46945.0	1472.2	3557.0	1225.8
		2	373	51438	9.09	45483	5.38	52153	3.09	51666	31574	44145	630.2	46631	53856	44474.2	52959.0	805.7	3336.4	1263.4
		3	400	50717	9.89	45698	5.75	51205	3.10	-	31340	44643	611.4	43214	52468	41880.2	51864.0	517.3	4655.0	1787.2
		4	424	47296	10.15	41623	5.94	47853	3.06	48792	30630	41629	615.4	43330	47117	40970.4	46693.8	464.0	4267.0	1475.2
		5	407	50402	9.84	46811	5.83	51000	3.04	49504	32099	41637	482.6	45006	50554	42509.0	49597.2	1132.3	3522.2	1331.8

주) Num : problem number, Capa. : Capacity of container, Time : Sec.

〈표 4〉 $d = 0.75$인 실험문제들에서의 성능 비교

Instance				HJ-SHC		HJ-GA		SB-GGA		SS-GA	Original NetKey			Improved NetKey						
											Best		Final	Best		Avg.		Final		
N	C	Num	Capa	Best	Time	Best	Time	Best	Time	Best	200	Final	Time	200	Final	200	Final	STD	Gen.	Time
100	3	1	669	69786	1.36	69769	1.01	69935	1.89	64335	63949	69271	27.8	74881	75780	74134.6	75466.2	308.2	1898.2	185.8
		2	714	69056	1.41	69146	1.03	69344	1.75	68164	64204	68321	23.0	72604	74229	72226.6	73850.6	280.3	1837.2	204.0
		3	686	68547	1.46	68763	1.02	68776	1.76	67643	64179	67394	43.2	72308	73368	71927.6	73058.6	238.4	2689.0	292.2
		4	666	69646	1.46	69907	1.04	69696	2.05	68626	66217	68922	31.2	72729	74321	72375.2	73880.6	443.1	2769.4	301.6
		5	668	69480	1.44	69410	1.02	69520	1.70	-	65079	68468	29.2	73368	75084	73045.2	74919.6	196.5	2136.2	214.0
100	5	1	401	48888	1.97	48663	1.26	48675	1.35	47449	40063	48104	79.4	52277	52981	51487.2	52742.2	297.3	2478.8	222.4
		2	428	48686	1.95	48990	1.32	48916	1.27	47766	40253	48060	75.0	50784	51807	50490.6	51673.2	107.8	2364.6	211.8
		3	411	47396	1.89	47512	1.29	48126	1.32	48008	39232	46180	104.2	49945	50652	48197.6	50354.2	384.5	2329.6	199.4
		4	400	49468	1.94	49845	1.30	49724	1.34	46921	42827	48734	59.0	51853	53491	51371.6	52991.8	391.6	2543.6	221.2
		5	400	47982	1.90	47925	1.29	48746	1.31	-	40900	46760	62.4	50796	52587	49978.0	51893.8	447.4	1970.4	151.6
100	10	1	200	29136	3.56	26603	2.23	29179	1.39	28767	22240	27895	191.6	30817	31450	29958.8	31114.6	371.3	2293.8	134.2
		2	214	30367	3.86	28663	2.43	30640	1.42	29824	22615	28141	141.4	31365	31969	30961.8	31826.0	108.4	2138.2	133.2
		3	205	28838	3.76	26176	2.31	28857	1.34	27960	22645	26588	140.0	29657	30598	28766.0	30361.0	177.7	2289.6	126.2
		4	200	30624	3.89	29701	2.40	31039	1.40	30712	23966	29265	152.2	32652	33155	31663.8	32862.8	208.0	2620.2	166.2
		5	200	29375	3.70	27130	2.28	29641	1.40	-	22838	26646	120.0	31554	32394	31013.6	32138.8	175.7	1753.0	94.0
200	3	1	1311	269447	4.33	268919	3.40	269351	5.40	261106	229948	264473	178.0	267032	282747	259959.2	277532.6	5463.4	4607.8	3034
		2	1414	255340	4.71	252977	3.47	255759	5.21	-	227861	252617	176.4	255412	266784	252669.8	263137.4	3572.9	3435.0	2454.2
		3	1342	268682	4.30	267731	3.39	269393	5.83	-	239483	266867	169.8	269982	279686	265761.0	277053.0	3448.6	2832.4	1918.8
		4	1565	245229	4.92	243192	3.64	245751	5.24	-	215422	242043	192.2	247899	256548	244946.8	255070.8	2019.5	3189.0	2268.2
		5	1336	277221	4.42	277762	3.36	277842	5.40	-	239629	274779	151.4	281473	288804	274295.0	286074.6	2310.4	2976.2	2271.2
200	5	1	786	182374	5.19	179525	3.83	183318	3.50	173905	137986	177809	343.4	179760	191488	169758.0	187262.2	3583.5	3170.6	1278.4
		2	848	172119	5.62	168021	3.98	172158	3.34	-	136266	167312	352.8	171439	177509	165067.2	175602.8	2428.3	2784.4	1427.2
		3	805	184362	5.13	181412	3.83	184727	3.50	-	141141	178949	427.8	184320	191830	173877.0	187803.4	4179.3	2665.8	1294.2
		4	939	163832	6.13	160146	4.22	164066	3.39	-	132458	159065	352.4	165661	171880	160296.8	169729.4	2263.6	2665.8	1329.4
		5	801	189756	5.23	187333	3.82	190069	3.54	-	142497	186356	351.0	189430	196878	178454.8	195147.8	2229.8	3008.2	1470.2
200	10	1	393	110238	8.87	102002	5.91	110528	3.00	106008	76366	99081	596.0	99725	113240	96073.6	110632.2	3519.8	3689.6	1231.6
		2	424	102734	9.59	92359	6.20	103363	2.89	-	75503	88704	556.0	97186	103469	92068.6	100955.4	3201.4	3110.0	1028.8
		3	402	111770	8.82	103848	5.95	112273	2.99	-	74534	101007	648.0	104150	115234	97542.6	112034.0	4787.1	3910.2	1178.6
		4	469	95453	10.71	85801	6.64	95839	2.97	-	70884	83517	496.6	86456	97042	85294.2	95172.6	3101.5	4524.2	1398.6
		5	400	114260	8.82	105078	5.91	114585	2.93	-	79390	97994	615.0	105001	118339	97885.0	114595.4	5927.7	3638.4	1267.6

주) Num : problem number, Capa. : Capacity of container, Time : Sec.

리즘을 제안하였다. 제안하는 방법은 해 표현법으로 개선된 랜덤 키 표현법(random key representation)을 사용하고 균등 교차 연산자와 상호교환 변이 연산자를 유전 연산자로 사용하였으며, 해의 개선을 위해 두 가지 휴리스틱 기법을 사용하였다.

제안하는 방법의 성능을 평가하기 위해 기존의 연구들이 사용했던 벤치마크문제에서 실험을 수행하여 기존 연구들과 성능을 비교하였는데, 제안하는 방법이 대부분의 문제들에서 비록 계산시간은 알고리즘의 구조적인 문제로 더 많이 필요로 하였지만 기존의 알려진 가장 우수한 결과보다 더 우수한 결과를 찾을 수 있음을 확인하였다.

결론적으로 제안하는 알고리즘이 현재 Quadratic 복수 컨테이너 문제를 해결하는 알고리즘들 중에서 가장 우수한 것으로 판단되며, 따라서 앞으로 이 문제를 다룰 경우 가장 우선적으로 고려되어야 할 것으로 판단된다.

참 고 문 헌

[1] 석상문, 장석철, 이상욱, 안병하, "고정비용수송문제를 위한 효율적인 진화알고리듬", 「*대한산업공학회지*」, 제31권, 제1호(2005), pp.79-86.

[2] Bäck, T., Fogel, D.B., and Michalewicz, Z., *Handbook of Evolutionary Computation*, Oxford University Press, 1997.

[3] Bean, J., "Genetic Algorithms and Random Keys for Sequencing and Optimization," *ORSA Journal on Computing*, Vol.6, No.2(1994), pp.154-160.

[4] Hiley, A. and Julstrom, B.A., "The quadratic multiple knapsack problem and three heuristic approaches to it," *Procs. of the genetic and evolutionary computation conference*, Vol.1 (2006), pp.547-552.

[5] Korea Maritime Institute, Dynamic Changes on Transshipment Cargoes among Northeast Asian Ports, Seoul, 2004.

[6] Raidl, G.R., "A Weight-Coded Genetic Algorithm for the Multiple Container Packing Problem," *Proc. of the 1999 ACM sympo. on Applied Computing*, (1999), pp.291-296.

[7] Rothlauf, F., D. Goldberg, and A. Heinzl, "Network Random Keys-A Tree Network Representation Scheme for Genetic and Evolutionary Algorithms," *Evolutionary Computation*, Vol.10, No.1 (2002), pp.75-97.

[8] Singh, A. and A.K. Gupta, "A hybrid heuristic for the maximum clique problem," *Journal of Heuristic*, Vol.12(2006), pp.5-22.

[9] Singh, A. and A.S. Baghel, "A New Grouping Algorithm for the Quadratic Multiple Knapsack Problem," in *EvoCOP 2007, LNCS*, Vol.4446(2007), pp.210-218.

[10] Sarac, T. and A. Sipahioglu, "A Genetic Algorithm for the Quadratic Multiple Knapsack Problem," in *BVAI2007, LNCS*, Vol.4729(2007), pp.490-498.

[11] Soak, S.M., D. Corne, and B.H. Ahn, "A New Encoding for the Degree Constrained Minimum Spanning Tree Problem," in *KES 2004, LNAI*, Vol.3213(2004), pp.952-958.

[12] http://cermsem.univ-paris1.fr/soutif/QKP.