工學碩士 學位論文

집합 패킹 문제에 적용된 컬럼 서브트랙션 해법구현에 관한 연구

Implementation of Column Subtraction Approach for Set Packing Problem

指導教授　金　是　和

2000年 2月

韓國海洋大學校 大學院

海事輸送科學科

黃　喜　秀

本 論文을 黃喜秀의 工學碩士 學位論文으로 認准함.

主審　　趙 誠 哲　　印

副審　　朴 鎭 洙　　印

副審　　金 是 和　　印

2000年 2月

韓國海洋大學校 大學院

海事輸送科學科

黃　　喜　　秀

目 次

Implementation of Column Subtraction Approach for Set Packing Problem

Hee-Su, Hwang

Department of Maritime Transportation Science
Graduate School of Korea Maritime University

Abstract

Set problems to be made up of special formulation among IP optimization models have various applications. One of them, ship scheduling problem has developed into set packing problem and made a key role in success of shipping management.

A great number of efforts have been made to build not only relevant optimization models, but also decision support system for ship scheduling problem. But no means are available to estimate the efficiency of those algorithms applicable to the optimization model.

This paper aims at implementing column subtraction algorithm applied to set packing problem, especially to ship scheduling problem.

A brief experiment shows that column subtraction algorithm works well with the problem, a decision support system which written by OOP computes for the efficiency of column subtraction algorithm compared to branch-and-bound algorithm and gives chance to test various algorithms.

제 1 장 서 론

1.1 연구의 배경

정수계획 최적화 모형에는 특별한 문제구조를 가지는 다양한 최적화 모형들이 있는데, 그 중에서도 모형의 응용범위가 매우 넓고 문제 구조가 유사한 것으로 집합 패킹 문제, 집합 분할 문제, 그리고 집합 포함 문제 등, 이른 바 집합문제들(Set Problems)이 있다. 이 문제들의 대표적인 응용분야들은 Balas(1976)가 언급한대로 철도나 항공기의 승무원 스케쥴링(crew scheduling), 트럭 배송(truck deliveries), 유조선 라우팅(tanker routing), 선박 운항일정 계획(ship scheduling), 정보 추출(information retrieval), 스위칭 회로 설계(switching circuit design), 스톡 절단(stock cutting), 조립 라인 균형(assembly line balancing), 자본소요 장비의 선정(capital equipment decision), 비상 시설의 설치(location of emergency units), 행정구역의 분할(political districting) 등 광범위한 영역에 걸쳐 있다.

이처럼 다양한 문제 영역에서 발생하는 집합문제들은 다음과 같이 정의할 수 있다.

$A = (a_{ij})$를 0-1의 원소로 이루어진 $m \times n$ 행렬, $\boldsymbol{e}$를 1의 원소로 이루어진 $m \times 1$ 열벡터, $\boldsymbol{c}$를 양의 정수 가중치(weights)로 이루어진 $1 \times n$ 행벡터라고 하자.

집합 포함 문제 (SC) Min $\{ cx \mid \mathrm{A}x \geq e,\ x \in [0, 1] \}$
집합 패킹 문제 (SPK) Max $\{ cx \mid \mathrm{A}x \leq e,\ x \in [0, 1] \}$
집합 분할 문제 (SPT) Min $\{ cx \mid \mathrm{A}x = e,\ x \in [0, 1] \}$

집합문제는 A의 각 열 j가 가중치 c_j를 가질 때, A의 열의 부분집합 W 중에서 A의 각 행을 많아야(적어도, 정확히) 한번 포함하는 열들을 선택하여 가중치 합을 최대(최소)화 하는 문제라 할 수 있으며, 그 해법으로는 열거해법, LP-완화기반의 해법, 라그랑쥬완화 기반의 해법, 네트워크완화 기반의 해법, Genetic Algorithm 등이 있다.

이러한 집합 문제의 응용분야들 중, 해사수송 문제와 밀접한 관련이 있는 유조선 라우팅, 선박 운항일정 계획문제 등은 해운의 경쟁력에 큰 영향을 끼치는 요소 중 하나이다. 그 이유는, 적절한 선박을 선택하여 경쟁력있는 무역로를 기획하고 잘 갖추어진 터미널을 이용하여 해사수송 서비스를

제공하면서 최대의 수익을 얻기 위한 준비를 다 한 후에도, 선박의 운항일정계획 등과 같은 운용적 국면의 문제들은 여전히 선박의 운항경제성의 향상과 운항이익의 극대화에 큰 영향을 미치는 의사결정문제로 남아 있기 때문이다. 이것이 선박 운항일정계획 문제가 해사수송 문제들 중에 최적화 연구의 대상으로 가장 오랜 관심을 끌었던 이유 중 하나라고 할 수 있다.

그러나 선박 운항일정계획 문제에 관련된 연구는, 이와 유사한 영역에 해당하는 육상의 수송일정계획문제인 VRP(vehicle routing problem)나 VSP(vehicle scheduling problem)에 비해 상대적으로 희소하였다. Ronen(1993)은 이러한 현상의 원인으로 해운 문제의 불확실성, 문제구조의 다변성, 해운기업의 보수성, 해운시장의 국제성, 개방성, 불안정성 등을 들어 설명하고 있다.

이러한 해운문제의 특성을 반영하여 과학적이고 합리적인 의사결정을 내릴 수 있도록 지원하는, 해사수송문제에 대한 최적화기반 의사결정 지원시스템에 관한 많은 연구가 이어져왔다. 그러나 의사결정지원 시스템이 개발되기까지 선박 운항일정계획의 최적화 모형의 개발과 시스템 구현에 관심이 집중되었으며, 다양한 해법들을 시스템에 적용하여 수행도를 평가할 수 있는 장치가 마련되어 있지 않았다.

이러한 이유를 배경으로, 집합문제 유형의 하나로 표현되는 선박 운항일정계획 최적화 모형에 컬럼 서브트랙션 해법을 적용한 의사결정 지원시스템을 구현하고, 해법적용에 대한 적절성과 유효성을 검증해 보고자 한다.

1.2 연구의 목적 및 방법

본 연구의 목적은, 해사수송문제에 대한 최적화기반 의사결정 지원시스템에서 사용하고 있는 집합 분할 문제 또는 집합 패킹 문제와 같은 0-1 정수계획 모형의 해를 구하기 위하여, 최근 개발된 LP-완화기반의 해법 중 하나인 컬럼 서브트랙션 해법과 의사결정 지원시스템에서 이용되는 "branch-and-bound" 해법의 수행도를 비교 · 분석하고, 적용한 컬럼 서브트랙션 해법의 적절성과 유효성을 검증하는데 있다.

연구의 방법으로 먼저 집합문제 유형의 최적화 모형의 해법 및 선박 운항일정계획 최적화모형에 대하여 개략적으로 살펴본 뒤, 가장 최근에 개발되었으며 단순한 벡터의 연산만으로 해를 도출해주는 컬럼 서브트랙션 해

법을 세척의 선박과 여섯개의 화물로 구성된 선박 운항일정계획 최적화모형의 예제에 적용하여 상세히 살펴본다.

이어서, 컬럼 서브트랙션 해법과 "branch-and-bound" 해법의 수행도를 비교할 수 있는 최적화기반 선박 운항일정계획 의사결정 지원시스템을 객체지향언어 중 하나인 볼랜드사의 델파이 4를 이용하여 구현한다.

마지막으로, 구현된 의사결정 지원시스템을 통하여 임의로 생성되는 선박 운항일정계획 최적화 모형에 대한 반복적인 계산실험의 결과를 비교·분석하여 컬럼 서브트랙션 해법의 수행도를 평가·검증한다.

1.3 논문의 구성

본 논문의 구성은 제1장 서론에 이어서 제2장에서는 집합문제 유형의 최적화 모형에 대한 해법과 선박 운항일정계획 최적화 모형을 선행연구를 통하여 개괄한 뒤, LP-완화기반의 컬럼 서브트랙션 해법을 예제를 통하여 구체적으로 살펴본다. 제3장에서는 선박 운항일정계획 최적화 문제를 해결하기 위해 이용될 컬럼 서브트랙션 해법과 "branch-and-bound" 해법의 수행도를 비교할 수 있는 의사결정 지원시스템을 구현한다. 제4장에서는 시스템을 통하여 실험한 결과를 비교·분석하고 제5장에서는 결론을 정리하여 본 논문을 맺는다.

제 2 장 집합 문제 유형의 최적화 모형과 해법

일반적으로 최적화 모형들은 그 해법에 대한 계산상의 복잡성이 문제의 크기에 대하여 "polynomially bounded"된 쉬운 문제 즉, P-문제 유형과 그렇지 않은 NP(Nondeterministic Polynomial)-문제 유형의 두 가지로 크게 분류된다. NP-문제들은 다시 NP-complete 문제와 그렇지 않은 문제들로 구분된다. NP-complete 문제는 NP-문제들 중 가장 어려운 문제들의 집합으로, 만약 NP-complete 문제 중의 어느 하나에 대하여 계산상 "polynomially bounded"인 해법이 발견된다면, NP-complete 문제들 전부가 P-문제가 되는 성질을 갖는다고 정의한다.

대부분의 정수계획 문제는 NP-문제들이며, 이 연구에서 주안하고 있는 집합 문제들은 NP-complete문제들이다. 최적화 모형 중 다양한 문제 영역에서 발생하는 집합 패킹(Set Packing), 분할(Set Partitioning), 포함(Set Covering)문제 등, 이른바 집합문제들(Set Problems)은 서론에서 정의한 바 있다.

집합 패킹, 분할, 포함 문제들을 자세히 살펴보면 집합 패킹문제(SPK)와 집합 분할문제(SPT)의 제약식은 0-1변수들 중에서 많아야 하나, 또는 정확히 하나만 1이어야 하는 팽팽한(tight) 제약이지만, 집합 포함문제(SC)의 제약식은 적어도 하나는 1이어야 하는 느슨한(loose) 제약이다. 이러한 문제 구조의 성질 때문에 (SPK)와 (SPT)의 상호관계는 그들과 (SC)와의 관계에 비하여 훨씬 더 근족의 관계이다.

이번 장에서는 이러한 집합문제 유형의 문제에 대한 해법들에 대하여 선행연구를 통하여 대략적으로 살펴보고, LP-완화 기반의 해법 중 하나인 Harche 등(1994)의 컬럼 서브트랙션(Column Subtraction)해법을 해사수송 문제 영역의 선박 운항일정계획 문제에 적용하여 본다.

2.1 집합문제의 해법

Kim(1999)은 집합문제 유형의 문제들의 해법에 대해 접근법에 따라 구분하면서, 해사수송 문제영역의 선박 운항일정계획을 위한 집합 패킹 모형 및 집합 분할 모형의 해법으로는 LP-완화 기반의 해법이 가장 적절하다고 하였다.

여기서는 집합문제 유형의 최적화문제의 해법들에 대하여 Kim의 접근법에 따라 개괄한다.

2.1.1 열거해법(Implicit Enumeration)

열거해법은 한번에 한 변수씩 0 또는 1의 값을 논리적으로 할당하여 부분해를 생성하면서 해집합을 체계적으로 탐색하여 나가는 나무탐색(Tree search)해법의 하나이며, 나눗셈 연산이 없는 가산적인 해법절차이어서 수치적으로 안정되어 있다.

2.1.2 LP-완화(LP-Relaxation) 기반의 해법

LP-완화 기반의 해법이란 정수계획 최적화 모형들의 정수제약을 완화하여 LP 최적화 모형으로 문제를 푼 뒤 정수해로 접근해 가는 것을 말한다. 정수계획 모형 해법의 중요한 관심사 중 하나는 그 모형의 정수제약을 완화한 문제가 정수해로 될 가능성에 관한 것이다. 집합 문제들의 행렬 A가 "totally unimodular"이면 그 LP-완화 문제는 정수해를 갖는다.

해사수송 문제 영역에서 실제로 발생하는 최적화 모형의 사례들도 LP-완화 문제가 정수해를 갖는 경우가 대부분이다. Appelgren(1969)도 그의 계산 실험에서 약 98%의 높은 비율로 정수해를 보여 주었다고 하였다. 그러나 반드시 정수해를 보장하지 않기 때문에 그 대책이 필요한 것이다.

LP-완화 기반의 해법은 "primal"이든 "dual"이든 "simplex"해법에 기초를 두고 있기때문에 높은 퇴화현상으로 항상 계산상의 최대 병목현상을 일으킨다. LP-완화 기반의 해법으로는 절단평면(cutting plane)해법, 분단탐색(Branch and bound) 또는 나무탐색(Tree search)해법, 컬럼 서브트랙션(Column Subtraction)해법 등이 있다.

2.1.3 라그랑쥬 완화(Lagrangean relaxation)기반의 해법

집합 문제들에 대한 라그랑쥬 완화 기반의 해법에 관한 연구로는 크게 두 가지 유형으로 구분할 수 있는데, 그 하나는 라그랑쥬 쌍대문제의 해법을 직접 다루는 접근법이며 다른 하나는 LP-완화 문제의 쌍대문제를 다루

는 접근법이다.

라그랑쥬 쌍대문제에 대한 대표적인 해법으로는 "subgradient optimization", "multiplier adjustment" 등이 있다. 라그랑쥬 쌍대문제의 최적해를 구하면 이를 원문제의 최적 정수해의 상한으로 두고 분단 탐색법, 나무 탐색법 또는 여러가지 발견적 기법들을 사용하여 최적해 방향으로 탐색을 계속한다. 이러한 라그랑쥬 완화법은 원문제인 어떤 정수계획 모형의 최적해에 가까운 상한을 구하기 위하여 또 다른 정수계획 모형을 해결해야 한다는 모순을 안고 있는 것처럼 보인다. 그러나 라그랑쥬 완화법이 정수계획 문제의 해결에 잘 사용되는 이유로는 첫째, 많은 까다로운 제약조건들을 완화시켜 목적함수로 올려버리면 쉽게 P-문제로 변한다는 것과 둘째, 라그랑쥬 완화법에 의한 실제 문제의 해결 경험에서 적절한 계산 비용으로 대단히 좋은 상한을 얻을 수 있다는 것 등이 있다.

LP-완화 문제의 쌍대문제 접근법으로는 "dual heuristic"을 들 수 있는데 이 접근법은 라그랑쥬 완화에서 사용하는 "dual ascent"나 "multiplier adjustment"와 매우 유사하다.

2.1.4 네트워크 완화(Network Relaxation) 기반의 해법

집합문제 유형을 포함하여 어려운 "combinatorial optimization" 문제들에 대한 또 다른 완화 기반의 해법으로 네트워크 완화기반의 해법이 있다. 네트워크 완화 기반의 해법은 어려운 "combinatorial optimization" 문제 속에 감추어진 네트워크(hidden network)구조를 찾아내어 원문제를 "side constraints"를 가진 쉽게 풀 수 있는 네트워크 문제로 변형하고, 이 "side constraints"를 완화한 네트워크 문제의 해와 "bound"를 원문제의 최적해를 구하기 위한 탐색에 이용하는 해법이다.

네트워크 완화 기반의 해법은 LP-완화 기법에 비하여 다음과 같은 이점을 가진다. 첫째는 계산 과정이 모두 정수 연산으로 이루어지며 대형의 행렬에 대한 반복적인 역행렬 계산이 필요없다는 점, 둘째는 많은 대형 네트워크 문제에 대한 매우 효율적인 해법들이 개발되어 있다는 점, 그리고 셋째는 LP-완화 기반의 해법이 갖는 계산상 병목의 주요 원인인 퇴화문제가 발생하지 않는다는 점 등이다.

2.1.5 Genetic Algorithm

일반적으로 "Genetic Algorithm"(GA)이란 지능형 확률적 탐색 해법의 하나로 이해할 수 있다. GA는 1975년 Holland에 의해 이론적 기초가 확립되었고, 그 기본개념은 적자 생존의 원리에 따르는 자연의 생물학적인 조직의 진화과정에 기초를 두고 있다. 적자 생존의 원리는 적응도가 높은 유전자(gene)일수록 다음 세대에서 더 많은 수의 개체가 퍼져나가며, 적응도가 높은 조상들의 좋은 형질의 결합은 훨씬 더 적응도가 높은 후손들을 생산한다는 것을 말해준다.

GA는 개체들의 초기 집단을 취하여 다음 세대의 생산에 대하여 일반적으로 선택(selection), 교차(crossover), 돌연변이(mutation) 등 세 가지 유전자조작(genetic operations)을 행함으로 이 진화과정을 모의 실험한다. 최적화 문제에 적용할 때, 집단의 각 개체는 주어진 문제의 가능해를 나타내는 염색체(chromosome)속에 기술된다. 개체의 적응도는 주어진 목적함수에 관하여 평가한다. 적응도가 높은 선택된 해들(solutions)은 교차조작 과정에서 그들의 유전정보의 일부를 교환하면서 다음 세대를 생산할 기회를 갖는다. 돌연변이는 선택교차가 수행된 후에 염색체의 일부값을 변경시키는 조작이다. 다음 세대 집단을 만드는 세대 교체 모델에는 세대 전체를 교체하는 "generational approach"와 적응도가 낮은 일부 개체를 교체하는 "steady-state approach"가 있다. GA는 이러한 평가-선택-재생성 과정을 만족할 만한 해를 찾을 때까지 반복한다.

2.2 선박 운항일정계획 최적화 모형

20세기 초 독점 제조공업회사 및 석유회사들이 자사화물의 경제적인 수송을 위해 그 산하에 해운기업을 거느리는 화주 직접운항 해운이 등장하면서, 선박 운항의 기본적인 유형을 정기선 운항(Liner operation), 부정기선 운항(Tramp operation), 그리고 화주 직접운항(Industrial operation)의 세가지로 구분하고 있다.

본 연구에서 다루고 있는 집합문제의 성격을 지닌 최적화 모형으로 표현할 수 있는 것으로는 부정기선 운항 또는 화주 직접운항 유형의 선박 운항일정계획(ship scheduling) 최적화문제라고 할 수 있다.

이 절에서는 선행연구들을 통하여 집합문제 유형으로 변화되어 온 선박 운항일정계획 최적화 모형에 대하여 살펴보고자 한다.

2.2.1 Appelgren(1969, 1971) 이전의 모형

유조선 운항일정과 관련된 Dantzig와 Fulkerson(1954)은 동형의 유조선과 단위 선박 선적량의 정수배 크기의 화물을 가정하여, 적·양하 항 및 일정이 정해진 화물의 수송수요를 충족시키기 위해 필요한 최소 선박의 수를 구하는 문제를 다루었다. Dantzig와 Fulkerson은 임의 화물의 수송을 위해 배선된 선박을 다른 화물의 수송에 가능한 한 많이 재배선하여 전체 투입 선박의 수를 최소화할 수 있다는 사실을 토대로, β날짜에 j 양하항에서 양하작업을 마친 선박을 다시 α날짜에 i 선적항에서 화물을 선적할 수 있도록 재배선하는 선박의 수 $x_{\alpha i \beta j}$ 를 최대화하는 선형계획모형을 제안 하였다. 이 모형은 이른바 히치코크 수송모형의 형태를 가지며 Dantzig와 Fulkerson은 간단한 표를 이용하여 이 문제의 최적해를 구하였다.

Laderman 등(1966)은 부정기선 운항 또는 화주 직접운항 유형에서 발생할 수 있는 선박의 라우팅 문제(Ship routing problem)를 다루었다. Laderman 등은 오대호의 다양한 항만간의 화물들을 수송하는 해운회사가 최소의 선박으로 수송수요를 충족하도록 보유선대의 선박을 할당하는 문제를 선형계획법으로 모형화하였다. Whiton(1967)은 이 모형에 항만의 화물 취급능력 및 접안능력 등에 관한 제약조건을 추가하여 그 모형을 확장하였으며, Laderman 등은 이 모형을 이용하여 계산실험을 행하고 정수해가 아닌 경우에는 "rounding off"한 값을 취하였다.

Bellmore(1968)는 Dantzig와 Fulkerson의 유조선 운항일정계획 문제를 변형하여, 제한된 선박을 이용하여 최대의 효용을 얻도록 선박의 운항일정계획을 수립하는 문제를 다루었다. Bellmore는 어떤 화물의 수송이 취소될 수도 있다는 가정 하에, 제한된 선박으로 각 화물수송의 효용(utility)을 고려하여 그 효용을 최대화하는 선박 운항일정계획을 수립하는 문제를 최소비용 최대효용의 네트워크 모형으로 표현하였다. Bellmore는 네트워크에 대하여 가지 상의 효용을 그 가지의 길이로 보아 임의마디 e에서 종료마디 t까지의 최대효용 $\pi(e)$를 최장경로 알고리즘으로 구하고, 임의의 가지 (e, f)에 대한 비용을 $a(e, f) = \pi(e) - \pi(f) - v(e, f)$와 같이 정의하여, 네트워

크 G^*의 최소비용 최대효용해를 찾아서 제한된 선박으로 화물 수송의 효용을 최대화하는 선박 운항일정계획 문제를 해결하였다.

2.2.2 Appelgren의 모형

Appelgren(1969, 1971)은 전형적인 부정기선 운항유형의 선박 운항일정계획 문제에 대하여, 선행연구들과는 다른 새로운 모형을 제안하였다. Appelgren은 부정기선 운항선대를 보유하고 있는 스웨덴의 한 해운회사의 선박 운항일정계획 문제를 다음과 같은 집합 분할(Set Partitioning) 또는 집합 패킹(Set Packing)모형으로 정식화하였다.

[기호]

$$a_{ijk} = \begin{cases} 1, & \text{만일 화물 } k\text{가 선박 } i\text{의 } j\text{번째 운항일정에 의해 수송될 경우,} \\ 0, & \text{그 외의 경우.} \end{cases}$$

$v_{ij} =$ 선박 i의 j번째 운항일정에 대한 이익.

[의사결정변수]

$$x_{ij} = \begin{cases} 1, & \text{만일 선박 } i\text{의 } j\text{번째 운항일정이 선택될 경우,} \\ 0, & \text{그 외의 경우.} \end{cases}$$

[모형]

$$Max \ \sum_{i} \sum_{j \in N(i)} v_{ij} x_{ij}$$

s. t.

$$\sum_{j \in N(i)} x_{ij} = 1, \text{ 모든 선박 } i\text{에 대하여}$$

$$\sum_{i} \sum_{j \in N(i)} a_{ijk} x_{ij} \le 1 \ \left(\text{또는 } \sum_{i} \sum_{j \in N(i)} a_{ijk} x_{ij} = 1\right), \text{ 모든 화물 } j\text{에대하여}$$

$$x_{ij} = \{0, 1\}, \text{ 모든 운항일정에 대하여}$$

Appelgren의 연구가 중요한 분수령이 되어 이후의 부정기선운항 또는 화주 직접운항 유형의 선박 운항일정계획 최적화모형은 거의 모두가 집합 분할 또는 집합 패킹모형의 형태를 취하고 있다.

2.2.3 Appelgren 이후의 모형

McKay와 Hartley(1974)는 Bellmore 등(1971)이 분할 선적을 허용하는 유조선 운항일정계획 문제를 혼합 정수계획으로 모형화 한 연구를 더 일반화하여 다루었으며, 선박의 운항가능일정의 집합을 정의하여 사용한 Appelgren의 모형을 수용하고 있다.

Brown 등(1987)은 중동에서 유럽과 북미 지역으로 원유를 수송하는 메이져 석유회사의 유조선 운항일정계획 문제를 집합 분할모형으로 정식화하였는데, 이 모형은 본질적으로 Appelgren(1969)의 모형과 유사하다.

Fisher와 Rosenwein(1989)은 살물(bulk cargo)선박의 운항일정계획 문제를 위한 대화형 최적화 시스템에 관한 연구에서, 각 화물에 대한 용선 요율과 통제 선대 내의 선박을 계획기간동안 계선시키지 않고 임의 스케줄에 투입하여 운항할 경우의 경비를 고려하여 집합 패킹 문제로 정식화하였다. Fisher와 Resenwein의 모형 역시 Appelgren(1969)과 Brown 등(1987)의 모형과 유사함을 알 수 있다.

Kim(1999)은 목적함수의 정의가 의사결정 기준에 따라 다른 것을 제외하면, 부정기선 운항과 화주 직접운항의 선박운항일정계획 최적화 문제가 동일한 구조를 가지기 때문에 Appelgren의 모형과 본질적으로 같아짐을 지적하면서, 이러한 문제구조의 성격을 최적화 의사결정 기준에 따라 구별하여 각각의 경우에 적용할 수 있는 일반화한 선박 운항일정계획 최적화 모형 (SPK-1)과 (SPK-2)를 제안하였다.

(1) 부정기선 운항의 선박 운항일정계획 최적화 모형 (SPK-1)

부정기선 해운을 경영하는 선주는 용선자인 화주와의 용선계약의 성약과정에서 신속한 의사결정을 통해 운임수익이 높은 화물을 확보하여야 단위 기간당 이익을 최대화할 수 있으며, 이러한 부정기선 운항의 선박운항일정계획에 적용할 수 있는 Kim(1999)의 최적화 모형(SPK-1)은 다음과 같다.

[기호]

$q_{ijk} = \begin{cases} 1, & \text{만일 선박 } i\text{가 운항일정 } j\text{에 투입되어 화물 } k\text{를 수송할 경우,} \\ 0, & \text{그 외의 경우.} \end{cases}$

$p_k =$ 화물 k를 수송할 경우의 운임수익

$h_{ij} =$ 운항일정 j에 투입된 선박 i의 운항비용

$J_i =$ 선박 i의 후보 운항일정 집합

[의사결정변수]

$x_{ij} = \begin{cases} 1, & \text{만일 선박 } i\text{가 운항일정 } j\text{에 투입될 경우,} \\ 0, & \text{그 외의 경우.} \end{cases}$

[모형(SPK-1)]

$$Max \ \sum_i \sum_{j \in J_i} \left(\sum_k q_{ijk} P_k \right) x_{ij} - \sum_i \sum_{j \in J_i} h_{ij} x_{ij}$$

$s.\ t.$

$$\sum_{j \in J_i} x_{ij} \le 1, \quad \text{모든 선박 } i\text{에 대해}$$

$$\sum_j \sum_{j \in J_i} q_{ijk} x_{ij} \le 1, \quad \text{모든 화물 } k\text{에 대해}$$

$$x_{ij} = \{0,\ 1\},\ j \in J_i\ , \text{ 모든 선박 } i\text{에 대해}$$

이 최적화 모형(SPK-1)의 목적함수는 보유 선대의 선박 i를 계획 기간 동안 운항할 때, 화물의 수송에 투입된 선박의 운항비의 총합에 대하여 화물의 수송에 의해 산출되는 운임수익의 총합을 최대화하는 것이다.

(2) 화주 직접 운항의 선박 운항일정계획 최적화 모형 (SPK-2)

화주 직접운항 해운의 경영자는 우선적으로 자사화물의 수송비용을 최소화하면서 그 선대의 운영을 최적화하려 할 것이며, 이에 따른 화주 직접 운항의 선박운항일정계획에 적용할 수 있는 Kim(1999)의 최적화 모형(SPK-2)은 다음과 같다.

[기호]

$q_{ijk} = \begin{cases} 1, & \text{만일 선박 } i\text{가 운항일정 } j\text{에 투입되어 화물 } k\text{를 수송할 경우,} \\ 0, & \text{그 외의 경우.} \end{cases}$

$p_k =$ 화물 k를 위해 선박을 용선할 경우의 용선비용

$h_{ij} =$ 운항일정 j에 투입된 선박 i의 운항비용

$J_i =$ 선박 i의 후보 운항일정 집합

[의사결정변수]

$x_{ij} = \begin{cases} 1, & \text{만일 선박 } i\text{가 운항일정 } j\text{에 투입될 경우,} \\ 0, & \text{그 외의 경우.} \end{cases}$

[모형(SPK-2)]

$$Min \ \sum_i \sum_{j \in J_i} h_{ij} x_{ij} - \sum_k (1 - \sum_{j \in J_i} \sum_i q_{ijk} x_{ij}) p_k$$

$s.\ t.$

$$\sum_{j \in J_i} x_{ij} \le 1, \quad \text{모든 선박 } i\text{에 대해}$$

$$\sum_j \sum_{j \in J_i} q_{ijk} x_{ij} \le 1, \quad \text{모든 화물 } k\text{에 대해}$$

$$x_{ij} = \{0, 1\}, \ j \in J_i, \ \text{모든 선박 } i\text{에 대해}$$

이 최적화 모형(SPK-2)의 목적함수의 첫째 항은 화물 수송에 투입한 자사선대의 선박 운항비에 해당되는 비용의 항이며, 둘째 항은 자사선대로 수송하지 못한 화물에 대해 지출해야 할 운임 즉, 기회비용의 항이므로 최소화 문제로 표현된다. Kim은 이 모형(SPK-2)에 대하여 화주 직접운항 선대의 운영과 관련된 자료와 변수들을 추가하여, 다음과 같은 집합분할모형으로 변형하였다.

[기호]

$q_{ijk} = \begin{cases} 1, & \text{만일 선박 } i\text{가 운항일정 } j\text{에 투입되어 화물 } k\text{를 수송할 경우,} \\ 0, & \text{그 외의 경우.} \end{cases}$

$p_k =$ 화물 k를 위해 선박을 용선할 경우의 용선비용

$h_{ij} =$ 운항일정 j에 투입된 선박 i의 운항비용

$J_i =$ 선박 i의 후보 운항일정 집합

$L_i =$ 운항일정에 투입되지 않은 선박에 대한 체선비용

[의사결정변수]

$$x_{ij} = \begin{cases} 1, & \text{만일 운항일정 } j\text{에 선박 } i\text{가 투입될 경우,} \\ 0, & \text{그 외의 경우.} \end{cases}$$

$$y_i = \begin{cases} 1, & \text{만일 선박 } i\text{을 투입할 운항일정이 없을 경우,} \\ 0, & \text{그 외의 경우.} \end{cases}$$

$$z_k = \begin{cases} 1, & \text{만일 화물 } k\text{를 수송할 수 없는 경우,} \\ 0, & \text{그 외의 경우.} \end{cases}$$

[모형(SPT)]

$$Min \quad \sum_i \sum_{j \in J_i} h_{ij} x_{ij} + \sum_i L_i y_i + \sum_k p_k z_k$$

$s.\ t.$

$$\sum_{j \in J_i} x_{ij} + y_i = 1, \quad \text{모든 선박 } i\text{에 대해}$$

$$\sum_i \sum_{j \in J_i} q_{ijk} x_k + z_k = 1, \quad \text{모든 화물 } k\text{에 대해}$$

$$x_{ij} = \{0,\ 1\},\ j \in J_i\ ,\ \text{모든 선박 } i\text{에 대해}$$

Kim은 부정기선 운항 또는 화주 직접운항 유형의 해사수송 문제의 핵심이 되는 선박운항일정계획 문제를 일반화하여 정의하고 그 최적화 모형을 집합문제 유형인 (SPK-1), (SPK-2)와 같이 제안하면서 목적함수의 계수를 정확히 산정할 수 있다는 가정하에 최적화 모형 기반의 시스템인 이른 바 MoDiSS를 개발하였다.

2.3 Harche 등(1994)의 컬럼 서브트랙션 해법

지금까지 집합문제 유형의 최적화모형의 해법과 선박 운항일정계획 최적화 모형에 대하여 살펴보았다. 여기에서는 집합 패킹 모형 또는 집합 분할 모형으로 표현되는 선박 운항일정계획 최적화 모형의 해법으로 적절하다고 생각되어지는 Harche 등(1994)의 컬럼 서브트랙션 해법과 적용에 대

하여 구체적으로 살펴본다.

2.3.1 집합 패킹 모형에 대한 이론적 고찰

집합 패킹 모형(SPK)은 다음과 같이 정의하였다.

$$\begin{aligned} & \text{Maximize} \quad cx \\ (\mathrm{P}) \quad & \mathrm{A}x \le e \\ & x \in [0,\ 1] \end{aligned}$$

위의 집합 패킹 문제는 다음과 같은 LP-완화 기반의 문제(RSPK: Relaxed Set Packing Problem)로 바꿀 수 있다.

$$\begin{aligned} & \text{Max} \quad dv && (1) \\ (\mathrm{P}') \quad & (\mathrm{A},\ I)v = e && (2) \\ & v \ge 0 && (3) \end{aligned}$$

단, $d = (cj,\ 0)$값을 가지는 $1 \times (m+n)$인 행벡터. (for $j = 1, \cdots, n$)

$v = (x,\ y)T$ 는 $m \times 1$인 열벡터이다.

단, $v_j = xj$ (for $j = 1, \cdots, n$),

$v_{n+i} = yi$ (for $i = 1, \cdots, m$). 여기서 yi는 $v_{n+i} = yi$ 인 "slack variable"을 의미한다.

$I = m \times m$ 인 단위행렬

위의 문제들은 일반성(generality)을 잃지 않고, A는 반복되거나 모든 요소(element)가 0인 열 또는 행이 없으며, $cj > 0$ (for $j = 1, \cdots, n$)이라고 가정한다

(SPK)의 해는 (RSPK)의 해를 구하였을 때 나타나는 "simplex" 테이블의 임의 열들을 우변항의 열에서 빼줌으로서 구할 수 있으며, 그 이유는 다음과 같다.

(RSPK)를 푼 "simplex" 최종테이블을 T 라고 하자.

$T(0)$를 "dual feasible"인 초기테이블, f 를 최적해에 이르기 위한 "dual pivot", $T(k)$ (for $k = 1, \cdots, f-1$)를 중간테이블, $T(f)$를 최종테이블이라 하고, 원문제(SPK)에 다음과 같은 형태의 변수고정(variable fixing) 제약식을 추가 한다고 하자.

$$v_{r(j)} \geq 1 \qquad (4)$$

단, $v_{r(j)}$ 은 $T(k)$ (for $k = 0, \cdots, f$)의 j열의 비기저변수이다.

비록 (4)는 부등식이지만 집합 패킹 모형의 특성상 $v_{r(j)} = 1$ 로 고정시키는 결과를 가져오므로 변수고정 제약식이라고 하였다. "simplex" 테이블에서 기저변수들은 비기저변수들의 형태로 표현될 수 있으며, 하나 혹은 여러개의 (4)과 같은 변수고정 제약식들이 동시에 문제에 추가될 수 있다.

왜냐하면, 만일 j와 k열($j \neq k$)의 비기저변수에 대하여 (4)형태의 제약식 2개를 원문제(SPK)에 추가하게 될 때, 임의 순서 혹은 동시에 추가할 수 있으며 그 결과는 같기 때문이다.

[정의]

$S = \{r(j1), \cdots, r(jh)\}$를 (4)형태의 제약식이 추가된 특정 비기저변수의 집합이라 하고, $T(k)$의 우변항(마지막) 열 $t^{(k)}_{i,n+1}$를

$$t^{(k)}_{i,n+1} - \sum_{r(j)\in s} t^{(k)}_{ij} \qquad (5)$$

로 대체한 테이블을 $T^{(k)}_s$라 할 때, $T^{(k)}_s$는 (SPK)의 S-제약 테이블이라고 하자. $T^{(k)}_s$는 $T(k)$의 우변항(마지막)열로부터 비기저변수 열들을 빼준 테이블임이 명백하다.

<정리1> [Harche 등(1994)]

1) $T^{(k)}_s$의 각 비기저변수 $v_{r(j)}$ (단, $r(j) \in S$)의 값은 1이 된다.

2) $T^{(k)}_s$의 기저변수들의 해와 $v_{r(j)} = 1$ (단, $r(j) \in S$)인 비기저변수의 해는 원문제의 제약식을 만족한다.

<증명> [Harche 등(1994)]

1) (4)의 제약식을 고쳐서

$$-v_{r(j)} \leq -1 \qquad (6)$$

단, $r(j) \in S$인 (6)과 같은 제약식들을 테이블에 더한다.

이 새로운 h개의 행을, 각각의 제약식에 대해 한 행씩 $T(k)$에 더해주고 이 새로운 제약식의 -1들에 대해 h번의 독립적 "dual pivot"을 행한다. 그

결과를 $T^*(k)$라고 하자. 비기저변수 $v_{r(j)} \in S$는 이제 1의 값을 가진 기저변수로 되었고, $T^*(k)$의 마지막열은 (5)와 같이 수정된 $T(k)$의 우변항(마지막) 열과 같다. $T_s^{(k)}$는 위에서 더해진 h개의 행들을 지움으로써 $T^*(k)$로부터 간단히 구할 수 있다.

2) 위의 1)에서 행한 "dual pivot"단계는 단순히 비기저(독립)변수들 $v_{r(j)} \in S$ 을 1이 되도록 고정하는 것과 같다. 왜냐하면 기저변수들은 비기저변수들의 값에 의하여 결정되기 때문에, 1로 고정된 비기저변수와 관련있는 기저변수는 제약식을 만족할 수 있는 다른 값을 가지게 된다.

<정리2> [Harche 등(1994)]

$T_s^{(f)}$를 SPK의 S-제약 마지막 테이블이라 하자.

1) $T_s^{(f)}$은 "dual feasible"이다.

2) $T_s^{(f)}$의 목적함수 값은 원문제의 하한을 제공한다.

3) $T_s^{(f)}$의 해가 "primal infeasible"이라면, 유한번의 "dual pivot"을 수행하면 "primal feasibility"를 얻을 수 있고, 마지막 테이블은 목적함수 값에 대한 참하한(true lower bound)을 제공한다.

<증명> [Harche 등(1994)]

1) 우변항으로부터 비기저변수 열을 빼는 것은 마지막 행에 영향을 미치지 않으므로 자명한 사실이다.

2) $T_s^{(f)}$는 "dual feasible"이란 사실로부터 유추가능하다.

3) 컬럼 서브트랙션은 $T_s^{(f)}$의 우변항(마지막) 열의 값을 바꾸게 되고, 그 값들은 기저변수들의 값이다. 그러므로 $T_s^{(f)}$의 "primal"해는 양수 또는 양수가 아닐 수도 있으므로 "primal feasble" 또는 "primal feasble"이 아닐 수도 있다. 마지막 열에 음수의 값이 존재하면, "dual pivot"를 유한번 수행하기 위해 "dual simplex"방법을 적용한다. 그 결과 음수값이 사라지면 "primal feasibility"가 회복되고, 마지막 테이블의 목적함수 값은 증가하게 된다. 그러므로 $T_s^{(f)}$의 초기 목적함수의 값은 제약조건을 지닌 문제의 최

적 목적함수값의 하한이다.

<정리3> [Harche 등(1994)]
문제 (SPK)에 대한 기저해를 v^*라 하자. v^*는 마지막 테이블 T(f)의 마지막 열에서 기껏해야 n번의 열을 빼어줌으로써 구할 수 있다.

<증명> [Harche 등(1994)]
$Q = Q^{(1)} \cup Q^{(2)}$라 하자. 단, 마지막 테이블 $T(f)$로부터 원문제의 해에 대해 $Q(1)$은 기저변수들의 집합이고 $Q(2)$는 비기저변수의 집합이다. 각 비기저변수 $v_{r(j)} \in Q^{(2)}$에 대해 $T(f)$의 $v_{r(j)}$열을 $T(f)$의 마지막 열에서 빼어준다. <정리1>의 1)에 의하여 비기저변수 $v_{r(j)}$는 $v_{r(j)}$ = 1인 기저변수로 된다.

2.3.2 집합 패킹 문제를 위한 컬럼 서브트랙션 해법

우선 집합 패킹 문제의 완화된 LP문제는 Sethi 등(1984)의 "pivot and probe" 알고리즘(PAPA)을 이용하면, LP-완화 기반의 문제들에서 많이 발생하는 퇴화문제를 해결할 수 있고 "simplex"에 의한 계산량보다 훨씬 저렴한 20 40%의 계산비용으로 문제를 풀 수 있다.

만일, LP-완화문제의 해가 정수해이면 마친다. 정수해가 아니면, 깊이 우선 탐색법(DFS: Depth First Search)으로, "simplex"의 마지막 테이블에서 비기저변수를 컬럼 서브트랙션하여 각 나무마디(tree node)에서 하한을 계산한다. 먼저, LP최적해보다 작은 정수로 상한 ZUB을 얻는다. 다음으로 오름차순의 수정비용 $C_1^* \le C_2^* \le ... \le C_n^*$에 따라 후보 비기저 변수 집합 L = {1, 2, ..., n}을 구한다. 수정비용이 같은 값일 경우 임의로 순서를 결정짓는다.

LP해에 의해 제공된 수정비용에 기초하여 "greedy heuristic"방법으로 구한 해는 (SPK)의 최적해에 대한 하한 ZLB을 제공한다. 물론, $|Z_{UB} - Z_{LB}| < 1$이면 "greedy heuristic"해는 (SPK)의 최적해이다. 그렇지 않으면 다음과 같이 "bounding test"를 실시하여, $|Z_{UB} - C_j^*| \le Z_{LB}$ 인 후보열 j를 제거하여 리스트 L을 줄여 나간다. 만일 비기저변수의 열 L이

모두 제거되면, 현재 구해놓은 정수해가 최적해가 된다. 그렇지 않으면, 앞서 설명한 방법으로 리스트 L에서 "subtraction" 과정을 실시할 비기저변수 j를 후보로 선택한다. $|Z_{UB} - C_j^*| \leq Z_{LB}$라면 더 낳은 해를 구할 수 없으므로 j에 대해서는 "subtraction"를 실시하지 않는다. $|Z_{UB} - C_j^*| > Z_{LB}$이면, 현재 "simplex" 테이블의 우변항에서 비기저변수 j열을 1회 "subtraction"한다.

만일 기저변수들이 모두 양의 정수라면 우리는 원문제 (SPK)에 대한 새로운 정수해를 발견한 것이다. Z*를 최소 정수해라면, 새로운 하한을 ZLB = Z*라 두고, 리스트 L에서 다시 "subtraction" 할 수 없거나 더 나은 해를 줄 수 없는 비기저변수열을 제거한다. 만일 이 부분적 해가 (SPK)에 대해 실행불가능 이라면 즉, 소수해이고 실행가능이거나 실행불가능 또는 정수해이고 실행 불가능한 경우에 우변항의 열을 저장한다. 전 단계에 이어서 다음으로 수정비용이 큰 비기저변수의 열을 리스트 L에서 고른다. 실행가능해이거나 실행불가능해에 관계없이 목적함수 값을 계산하여 각 "tree node"에 대한 "bounding test"를 실시한다. 이것은 단순한 벡터의 가감이기 때문에 계산이 매우 용이하다. 이때의 해가 현재의 해보다 나은 경우가 아니면 이에 해당하는 노드(node)는 체크하고, 더 이상 고려하지 않는다. 가장 최근에 열었던 노드를 먼저 탐색한다. 이 탐색은 각각의 가지(tree level)에서 비기저 변수에 1 또는 0의 값을 고정하여 수행한다. 비기저변수에 1을 고정는 것은 현재 "simplex" 테이블에서 해당되는 비기저 변수의 열을 1회 감하는 것이다. 최소 정수해가 발견되면 "backtrack"한다. 비슷한 방법으로 목적함수의 해가 상한을 초과하면 현재 노드를 체크하고, "backtrack"한다. 비기저변수 열의 리스트 L이 비면 열거 탐색법을 마친다. 현재 존재하는 정수해 중에서 가장 좋은 것이 최적해이다. 만일 탐색을 끝낸 뒤 실행가능해가 존재하지 않으면 원문제는 해가 없다.

나무탐색의 어느 노드에서든지 부분적 해는 다음 중 하나이다.

1) 원문제는 실행가능하며 정수해. 연관 된 해의 값은 최적해의 참하한(true lower bound)이다.

2) 원문제는 실행가능하며 소수해. 목적함수 값은 원문제의 ZUB와 같은 상한이거나 더 나은 값이다.

3) 원문제는 실행불가능한 정수 또는 소수해. 해가 존재하지 않는다.

집합 패킹 문제를 위한 컬럼 서브트랙션 해법을 요약하면 다음과 같다.

완화된 LP문제의 최종 테이블을 T$(i,\ j)$라 하자. T$(i,\ j)$는 m+1행과 n+1열을 가지는데, n+1열은 기저변수들의 값을, m+1행은 수정비용의 값들을 나타낸다. 특별히, $T(m+1,\ n+1)$은 목적함수값을 나타낸다. ZUB를 $T(m+1,\ n+1)$보다 작은 정수값으로 둔다. 하한을 찾기위하여, "heuristic"한 방법으로 실행가능한 해를 찾고, 그 해를 ZLB로 둔다. "heuristic"의 해를 구할 수 없으면 하한을 ZLB = $-\infty$ 이라고 둔다.

탐색나무의 마디들을 기록하기 위한 행렬 $(m+1)\times n$ 행렬, 원소가 $E(i,\ j)$인 행렬 E를 사용한다. LIFO(last in, first out) 또는 깊이우선 탐색법(DFS: Depth First Search)을 이용한다.

[단계 0] (Heuristic 해) "heuristic"한 방법으로 찾은 실행가능한 해를 ZLB로 둔다. ZUB − ZLB = 0 이면 [단계 6]으로 간다. "heuristic"의 해를 구할 수 없으면 하한을 ZLB = $-\infty$ 이라 둔다. T의 열들 중 수정비용이 ZUB - ZLB 보다 큰 것을 제거한다.

[단계 1] (초기화) i = 1, ..., m+1에 대하여 $E(i,\ j) = T(i,\ n+1)$로 둔다. $scol = 1$, $index = 1$, $acol = point(1)$로 두고 [단계 2]로 간다.

[단계 2] (전진탐색) $scol = scol + 1$이라 둔다. i = 1, ..., m+1에 대하여 $E(i,\ scol) = E(i,\ scol-1) - T(i,\ scol)$ 라 둔다.
$save\_index(scol) = index$, $index = index + 1$로 두고 [단계 3]으로 간다.

[단계 3] (정수해검사) i = 1, ..., m+1에 대하여 $E(i,\ j)$가 비음정수이면, 이 해를 현재의 새로운 최적해 ZLB로 갱신한 뒤 [단계 5]로 간다. 그렇지 않으면 [단계 4]로 간다.

[단계 4] (점검단계) 만일 $index > n$이면 [단계 5]로 간다. $acol = point(index)$라 둔다. 만일 $E(m+1,\ scol) - T(m+1,\ scol) \le$ ZLB 이면 [단계 5]로 간다. 그렇지 않으면 [단계 2]로 간다.

[단계 5] (후진탐색) $index = save\_index(scol) + 1$, $scol = scol - 1$로 둔다. 만일 $scol > 0$이면 [단계 4]로 간다. 그렇지 않으면 [단계 6]으로 간다

[단계 6] (해 출력) 현재 최적해라고 증명된 해를 출력한다. 끝.

2.3.3 컬럼 서브트랙션 해법의 적용

지금까지 집합 패킹 문제에 적용된 Harche 등(1994)의 컬럼 서브트랙션 해법에 대하여 상세히 살펴보았다. 여기서는 컬럼 서브트랙션 해법을 선박 운항일정계획 최적화 문제에 직접 적용하여, 실제문제에 대한 해법의 적절성과 효용성에 대하여 살펴본다.

[예제] 선박 운항일정계획 최적화 모형

Maximize

$$\begin{aligned} & 1829\ X1 + 1729\ X2 + 1499\ X3 + 1080\ X4 + 1947\ X5 + 1800\ X6 \\ & + 1315\ X7 + 1263\ X8 + 1453\ X9 + 1842\ X10 + 1238\ X11 + 1908\ X12 \\ & + 1335\ X13 + 1988\ X14 + 1990\ X15 + 1579\ X16 + 1165\ X17 + 1335\ X18 \\ & + 1722\ X19 + 1245\ X20 + 1513\ X21 + 1035\ X22 \end{aligned}$$

Subject to

[선박1] $X1 + X2 + X3 + X4 + X5 + X6 + X7 \leq 1$

[선박2] $X8 + X9 + X10 + X11 + X12 + X13 + X14 + X15 \leq 1$

[선박3] $X16 + X17 + X18 + X19 + X20 + X21 + X22 \leq 1$

[화물1] $X3 + X7 + X17 + X20 \leq 1$

[화물2] $X3 + X6 + X9 + X10 + X15 + X19 + X21 \leq 1$

[화물3] $X1 + X3 + X12 + X13 + X15 + X16 + X18 \leq 1$

[화물4] $X5 + X12 + X13 + X15 + X17 + X18 + X22 \leq 1$

[화물5] $X2 + X4 + X8 + X13 + X14 + X15 + X17 + X18 + X19 \leq 1$

[화물6] $X4 + X10 + X11 + X14 + X16 + X21 + X22 \leq 1$

$$X_i = \begin{cases} 1, & \text{만약 운항일정 } i \text{ 가 선택되면} \\ 0, & \text{그렇지 않으면} \end{cases}$$

목적함수는 선박을 투입하여 화물을 수송할 수 있는 스케줄들을 변수로, 투입된 선박의 운항가능 스케줄을 통하여 벌어들일 수 있는 총 이익금액을 계수로 이루어져 있다. 선박은 각각의 스케줄에 대하여 한번만 투입될 수 있도록, 화물은 각각의 스케줄에 의해 한번만 운송될 수 있도록 구성되어 있다.

[예제]는 정수계획모형이며, LP-완화문제로 바꾸기 위하여 $Xi \geq 0$ 으로 마지막 제약식을 대체한다. 이 LP-완화문제를 풀어 최종 "simplex" 테이블을 구하면 [표 2-1]과 같다. [표 2-1]의 첫 번째 행은 목적함수의 변수와 Slack 변수를, 마지막 행은 수정비용과 최대 목적함수값을, 오른쪽 열은 기저변수와 기저변수의 해를 나타낸다.

문제의 해 Z = 5494.66667은 정수해가 아니므로 ZUB = 5494라 두고, 최종 "simplex" 테이블로부터 컬럼 서브트랙션 해법을 적용한다.

비기저변수의 집합을 S 라 하면, S = { X2, X3, X4, X6, X7, X8, X9, X11, X13, X15, X17, X18, X21, X22, S1, S2, S3, S5, S6, S7, S8, S9 } 이다.

"heuristic" 한 방법으로 해를 구하는 요령으로, 너비우선 탐색(BFS: Breadth First Search)을 실시한다. $T(i, n+1)$ - $S(i, j)$의 정수해 중 최대값을 구한다. 즉, 우변항에서 각 비기저변수의 열을 빼어 최대의 정수해를 구한다(단, i = 1, …, 10, j = 1, …, 22, n = 31).

$S(i, j)$ 는 최종테이블 중 비기저변수 열을 의미한다. 이 때 구한 정수해들은 하한을 제공하며, 최적해(ZOPT)에 빨리 도달하기 위해서 ZLB값 중 최대값을 찾는 것이다. 정수해를 구할 수 없으면, ZLB = 0 으로 둔다. 정수해를 제공하는 기저변수를 발견하면 ZUB — ZLB 보다 큰 수정비용을 가지는 비기저변수를 $S(i, j)$에서 제거하고 집합 S = $\varnothing$이면 멈춘다.

[표 2-1]로부터 위와 방법을 이용하면, 비기저변수 집합 S 중 정수해를 제공하는 변수는 X17 이며, X1 = X10 = X17 = 1, ZLB = 4836 인 정수해를 얻을 수 있다. 이 정수해는 X17 $\geq$ 1인 제약식을 LP-완화문제에 추가하는 것과 같은 원리이며, SPK 문제(Set Packing Problem)의 특성상 X17 $\geq$ 1 의 제약식은 X17 = 1 로 고정시키는 결과를 가져다 준다.

제약식 X17 $\geq$ 1 (즉, - X17 $\leq$- 1)을 LP-완화문제의 최종 "simplex" 테이블에 추가하여 "dual pivot"을 행한 결과는 [표 2-2]와 같다.

[표 2-1] LP-완화문제의 최종 "Simplex" 테이블

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	X_{16}	X_{17}
1	0.66667	1	0.33333	0	0.66667	0.66667	0.33333	0.33333	0	0	0	0	0	0	0	-0.66667
0	0.33333	0	0.66667	1	0.33333	0.33333	-0.33333	-0.33333	0	0	0	0	0	0	0	0.66667
0	0	-0.5	-0.5	0	0	0.5	0.5	0.5	0	0.5	0	-0.5	0	-1	0	0
0	0	1.5	0.5	0	0	0.5	-0.5	-0.5	0	-0.5	0	0.5	0	1	0	1
0	-0.33333	0.5	-0.16667	0	0.66667	0.16667	-0.16667	0.83333	1	0.5	0	-0.5	0	0	0	-0.66667
0	-0.33333	0	0.33333	0	-0.33333	-0.33333	-0.66667	-0.66667	0	0	0	0	0	0	1	0.33333
0	-0.33333	0	-0.66667	0	-0.33333	-0.33333	0.33333	0.33333	0	0	1	1	0	1	0	0.33333
0	0.66667	-0.5	0.83333	0	-0.33333	0.16667	0.83333	-0.16667	0	0.5	0	0.5	1	0	0	0.33333
0	0.33333	0.5	0.16667	0	0.33333	-0.16667	0.16667	0.16667	0	-0.5	0	0.5	0	1	0	0.66667
0	262.33333	495.5	1096.16667	0	45.33333	364.83333	540.16667	204.16667	0	438.5	0	884.5	0	395	0	658.66667

X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	RHS	Basic Var.
-0.33333	0	0	-0.33333	-1	0.66667	0.33333	0	0	0	0.33333	-0.66667	0	-0.33333	0.33333	X_1
0.33333	0	0	0.33333	1	0.33333	-0.33333	0	0	0	-0.33333	0.66667	0	0.33333	0.66667	X_5
-0.5	0	1	0	0	0.5	1	1	0	-0.5	-0.5	-0.5	-0.5	-0.5	0	X_{20}
0.5	0	0	0	0	-0.5	-1	-1	1	0.5	0.5	0.5	0.5	0.5	1	S_4
-0.83333	0	0	0.66667	0	0.16667	0.33333	0	0	0.5	-0.16667	-0.16667	-0.5	0.16667	0.33333	X_{10}
0.66667	0	0	0.66667	1	-0.33333	-0.66667	0	0	0	0.33333	0.33333	0	0.66667	0.33333	X_{16}
0.66667	0	0	-0.33333	0	-0.33333	0.33333	0	0	0	0.33333	0.33333	0	-0.33333	0.33333	X_{12}
0.16667	0	0	-0.33333	0	0.16667	0.33333	0	0	-0.5	-0.16667	-0.16667	0.5	0.16667	0.33333	X_{14}
0.83333	1	0	0.33333	0	-0.16667	-0.33333	0	0	0.5	0.16667	0.16667	0.5	-0.16667	0.66667	X_{19}
637.83333	0	0	82.33333	662	1679.83333	1491.66667	1245	0	165.5	149.16667	267.16667	311.5	184.83333	5494.66667	

[표 2-2] 제약식 $X_{17} \geq 1$를 추가하여 "Dual Pivot"을 행한 결과

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	X_{16}	X_{17}
1	0.66667	1	0.33333	0	0.66667	0.66667	0.33333	0.33333	0	0	0	0	0	0	0	0
0	0.33333	0	0.66667	1	0.33333	0.33333	-0.33333	-0.33333	0	0	0	0	0	0	0	0
0	0	-0.5	-0.5	0	0	0.5	0.5	0.5	0	0.5	0	-0.5	0	-1	0	0
0	0	1.5	0.5	0	0	0.5	-0.5	-0.5	0	-0.5	0	0.5	0	1	0	0
0	-0.33333	0.5	-0.16667	0	0.66667	0.16667	-0.16667	0.83333	1	0.5	0	-0.5	0	0	0	0
0	-0.33333	0	0.33333	0	-0.33333	-0.33333	-0.66667	-0.66667	0	0	0	0	0	0	1	0
0	-0.33333	0	-0.66667	0	-0.33333	-0.33333	0.33333	0.33333	0	0	1	1	0	1	0	0
0	0.66667	-0.5	0.83333	0	-0.33333	0.16667	0.83333	-0.16667	0	0.5	0	0.5	1	0	0	0
0	0.33333	0.5	0.16667	0	0.33333	-0.16667	0.16667	0.16667	0	-0.5	0	0.5	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	262.33333	495.5	1096.16667	0	45.33333	364.83333	540.16667	204.16667	0	438.5	0	884.5	0	395	0	658.66667

X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	RHS	Basic Var.
-0.33333	0	0	-0.33333	-1	0.66667	0.33333	0	0	0	0.33333	-0.66667	0	-0.33333	1	X_1
0.33333	0	0	0.33333	1	0.33333	-0.33333	0	0	0	-0.33333	0.66667	0	0.33333	0	X_5
-0.5	0	1	0	0	0.5	1	1	0	-0.5	-0.5	-0.5	-0.5	-0.5	0	X_{20}
0.5	0	0	0	0	-0.5	-1	-1	1	0.5	0.5	0.5	0.5	0.5	0	S_4
-0.83333	0	0	0.66667	0	0.16667	0.33333	0	0	0.5	-0.16667	-0.16667	-0.5	0.16667	1	X_{10}
0.66667	0	0	0.66667	1	-0.33333	-0.66667	0	0	0	0.33333	0.33333	0	0.66667	0	X_{16}
0.66667	0	0	-0.33333	0	-0.33333	0.33333	0	0	0	0.33333	0.33333	0	-0.33333	0	X_{12}
0.16667	0	0	-0.33333	0	0.16667	0.33333	0	0	-0.5	-0.16667	-0.16667	0.5	0.16667	0	X_{14}
0.83333	1	0	0.33333	0	-0.16667	-0.33333	0	0	0.5	0.16667	0.16667	0.5	-0.16667	0	X_{19}
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	X_{17}
637.83333	0	0	82.33333	662	1679.83333	1491.66667	1245	0	165.5	149.16667	267.16667	311.5	184.83333	4836	

ZLB를 결정하였으므로, 컬럼 서브트랙션 해법에 따라 더 낳은 해를 찾는다. 이 때, 오름차순으로 정렬된 비기저변수의 열들중 "slack variable"에 대하여 먼저 해를 탐색하는 "slack variable heuristic"을 이용하면 아주 적은 계산량으로 탐색단계에서 일찍 해를 발견할 수 있다. 또한 정수해의 여부를 판별할 때, m+1번째 행의 차이값이 정수인지 아닌지를 먼저 검사해 보면 계산량을 줄일 수 있다.

컬럼 서브트랙션 해법을 적용한 결과는 다음과 같다.

[단계 0] ZUB = 5494, ZLB = 4836.

ZUB ≠ ZLB 이므로, 집합 T 에서 ZUB−ZLB 보다 작은 수정비용을 가지는 비기저변수 열들을 제거한 후, "slack variable"에 대해 오름차순으로 정렬한다. 그 집합을 S'라 하면 S'(i, j)는 [표 2-3]과 같다. 단, i = 1, …, 10, j = 1, …, 6, n = 5, $S'(i, 6) = T(i, 32)$.

[표 2-3] 수정비용에 따라 오름차순으로 정렬한 "slack variable"

point(1)	point(2)	point(3)	point(4)	point(5)		
s6	s5	s9	s7	s8	RHS	Basic Var.
0.33333	0	-0.33333	-0.66667	0	0.33333	x1
-0.33333	0	0.33333	0.66667	0	0.66667	x5
-0.5	-0.5	-0.5	-0.5	-0.5	0	x20
0.5	0.5	0.5	0.5	0.5	1	s4
-0.16667	0.5	0.16667	-0.16667	-0.5	0.33333	x10
0.33333	0	0.66667	0.33333	0	0.33333	x16
0.33333	0	-0.33333	0.33333	0	0.33333	x12
-0.16667	-0.5	0.16667	-0.16667	0.5	0.33333	x14
0.16667	0.5	-0.16667	0.16667	0.5	0.66667	x19
149.16667	165.5	184.83333	267.16667	311.5	5494.66667	

[단계 1] 최종 테이블의 마지막 열을 $E(i, 1)$에 저장한다.

$E(i, 1) = T(i, 32)$. $scol$ = 1, $index$ = 1, $acol = point(1)$. $point(j)$는 S'의 j번째 열을 의미한다. $save\_index(1) = 0$. [단계 2]로 간다.

[단계 2] $scol = scol + 1 = 2$. $E(i, 2) = E(i, 1) - S'(i, point(1))$.
$save\_index(2) = 1$, $index = index + 1 = 2$. [단계 3]으로 간다.

[단계 3] $E(i,\ 2)$는 정수해가 아니므로 [단계 4]로 간다.

[단계 4] $index = 2 < 5$. $acol = point(index) = point(2)$.
$E(10,\ 2) - S'(10,\ point(2)) = 5180.00000 > ZLB$ 이므로 [단계 2]로 간다.

[단계 2] $scol = 3$. $E(i,\ 3) = E(i,\ 2) - S'(i,\ point(2))$.
$save\_index(3) = 2$, $index = 3$. [단계 3]으로 간다.

[단계 3] $E(i,\ 3)$은 정수해이므로 ZLB = 5180.00000 으로 갱신한 뒤, [단계 5]로 간다. ZUB - ZLB = 5494 - 5180 = 314 이며, S'의 모든 수정비용은 314보다 작다. 그러므로, S'을 계속해서 후보집합으로 사용한다.

[단계 5] $index = save\_index(3) + 1 = 3$. $scol = scol - 1 = 2$.
$scol > 0$ 이므로 [단계 4]로 간다.

[단계 4] $index = 3 < 5$. $acol = point(3)$.
$E(10,\ 2) - S'(10,\ point(3)) = 5160.66667 < ZLB$ 이므로 [단계 5]로 간다.

[단계 5] $index = save\_index(2) + 1 = 2$. $scol = 1$.
$scol > 0$ 이므로 [단계 4]로 간다.

[단계 4] $index = 2 < 5$. $acol = point(2)$.
$E(10,\ 1) - S'(10,\ point(2)) = 5329.16667 > ZLB$ 이므로 [단계 2]로 간다.

[단계 2] $scol = 2$. $E(i,\ 2) = E(i,\ 1) - S'(\mathrm{i},\ point(2))$.
$save\_index(2) = 2$, $index = 3$. [단계 3]으로 간다.

[단계 3] $E(i,\ 2)$는 정수해가 아니므로 [단계 4]로 간다.

[단계 4] $index = 3 < 5$. $acol = point(3)$.
$E(10,\ 2) - S'(10,\ point(3)) = 5144.33334 < ZLB$ 이므로 [단계 5]로 간다

[단계 5] $index$ = $save\_index(2)$ + 1 = 3. $scol$ = 1.
$scol$ > 0 이므로 [단계 4]로 간다.

[단계 4] $index$ = 3 < 5. $acol$ = $point(3)$.
$E(10, 1)$ - $S'(10, point(3))$ = 5309.83334 > ZLB 이므로 [단계 2]로 간다.

[단계 2] $scol$ = 2. $E(i, 2) = E(i, 1) - S'(\mathrm{i}, point(3))$.
$save\_index(2)$ = 3, $index$ = 4. [단계 3]으로 간다.

[단계 3] $E(i, 2)$는 정수해가 아니므로 [단계 4]로 간다.

[단계 4] $index$ = 4 < 5. $acol$ = $point(4)$.
$E(10, 2)$ - $S'(10, point(4))$ = 5042.66667 < ZLB 이므로 [단계 5]로 간다.

[단계 5] $index$ = $save\_index(2)$ + 1 = 4. $scol$ = 1.
$scol$ > 0 이므로 [단계 4]로 간다.

[단계 4] $index$ = 4 < 5. $acol$ = $point(4)$.
$E(10, 1)$ - $S'(10, point(4))$ = 5227.50000 > ZLB 이므로 [단계 2]로 간다.

[단계 2] $scol$ = 2. $E(i, 2) = E(i, 1) - S'(\mathrm{i}, point(4))$.
$save\_index(2)$ = 4, $index$ = 5. [단계 3]으로 간다.

[단계 3] $E(i, 2)$는 정수해가 아니므로 [단계 4]로 간다.

[단계 4] $index$ = 5 $\leq$ 5. $acol$ = $point(5)$.
$E(10, 2)$ - $S'(10, point(5))$ = 4916.00000 < ZLB 이므로 [단계 5]로 간다.

[단계 5] $index$ = $save\_index(2)$ + 1 = 5. $scol$ = 1.
$scol$ > 0 이므로 [단계 4]로 간다.

[단계 4] $index$ = 5 $\leq$ 5. $acol$ = $point$(5).
E(10, 1) - S'(10, $point$(5)) = 5183.16667 > ZLB 이므로 [단계 2]로 간다.

[단계 2] $scol$ = 2. E(i, 2) = E(i, 1) - S'(i, $point$(5)).
$save\_index$(2) = 5, $index$ = 6. [단계 3]으로 간다.

[단계 3] E(i, 2)는 정수해가 아니므로 [단계 4]로 간다.

[단계 4] $index$ = 6 > 5. [단계 5]로 간다.

[단계 5] $index$ = $save\_index$(2) + 1 = 6. $scol$ = 1.
$scol$ > 0 이므로 [단계 4]로 간다.

[단계 4] $index$ = 6 > 5. [단계 5]로 간다.

[단계 5] $index$ = $save\_index$(1) + 1 = 1. $scol$ = 0.
$scol$ = 0 이므로 [단계 6]로 간다.

[단계 6] 현재까지의 해 중 가장 최적한 해를 출력한다.
X5 = X14 = X20 = 1, ZOPT = ZLB = 5180.00000.

컬럼 서브트랙션 해법을 이용한 선박 운항일정계획 모형의 최적해는 ZOPT = 5180.0 이며, 이 해는 LINDO Optimizer를 이용하여 구한 최적해와 동일하였다. 그리고, 해를 구하는 과정에서 최적해는 아니지만 정수해인 4836.0과 4916.0를 발견할 수 있었다. 이 정수해들은 해당 운항일정계획을 수립할 수 없을 때, 즉 ZOPT인 해를 이용할 수 없을 때 차선책으로 이용될 수 있음을 의미한다.

제 3 장 컬럼 서브트랙션 해법의 전산구현

집합문제 유형의 좋은 예가 되는 선박 운항일정계획 최적화 모형의 해법으로는 컬럼 서브트랙션 해법이 적절함을 제2장에서 예제를 통하여 살펴보았다. 이번 장에서는 선박 운항일정계획 최적화모형의 의사결정변수가 되는 운항가능일정 생성 그래프와 알고리즘 및 컬럼 서브트랙션 해법을 전산구현하여, 해법의 유효성을 검증할 수 있는 의사결정지원 시스템을 구축한다. 그러나, 본 논문의 성격상 의사결정지원 시스템에 대한 시스템 분석 및 설계, 개발 등에 관한 상세한 내용은 언급하지 않는다.

3.1 시스템 구현

본 연구는 해법의 수행도 비교에 초점을 맞추고 있기 때문에 특별한 GUI(Graphic User Interface)가 필요 없었지만, 임의로 생성되는 자료들 이외에도 실험자가 원하는 자료들을 가지고도 실험할 수 있도록 완전한 의사결정 지원시스템의 형태로 구현 하였다. 시스템 구현에 사용된 프로그래밍 언어는 객체지향 프로그래밍언어(OOPL)라고 할 수 있는 볼랜드사의 델파이 4이며, 구현환경은 Windows 98, Pentium Ⅱ CPU가 탑재된 PC 상에서 시스템을 개발하였다.

3.1.1 시스템 흐름도

집합문제 유형의 최적화 모형을 기반으로 하는 선박 운항일정 계획 의사결정 지원시스템에서 사용되는 주요 입력자료는 선박, 화물, 항만의 제 특성과 해사수송 문제에서 발생하는 다양한 제약조건들이다.

이러한 입력자료들을 초기입력자료로 하여 데이터베이스로 구축하고 난수를 발생시켜 임의의 데이터 형태로 무작위 변경되도록 하였다. 이렇게 형성된 자료들을 바탕으로 모든 운항가능일정을 생성하는 모듈을 구현하였으며, 생성된 운항가능일정을 변수로 최적화 모형을 구축하여 LINDO 및 컬럼 서브트랙션 해법에 의하여 문제를 해결할 수 있도록 하였다. 또한 적절한 GUI를 이용하여 입력자료, 운항가능 일정, 최적해, LP완화 문제의 최종 테이블, LINDO 및 컬럼 서브트랙션 해법에 의해 해를 구하는데 소요되는

시간 등을 출력할 수 있도록 하였다.

개괄적인 시스템 흐름도는 Fig. 3.1 과 같다.

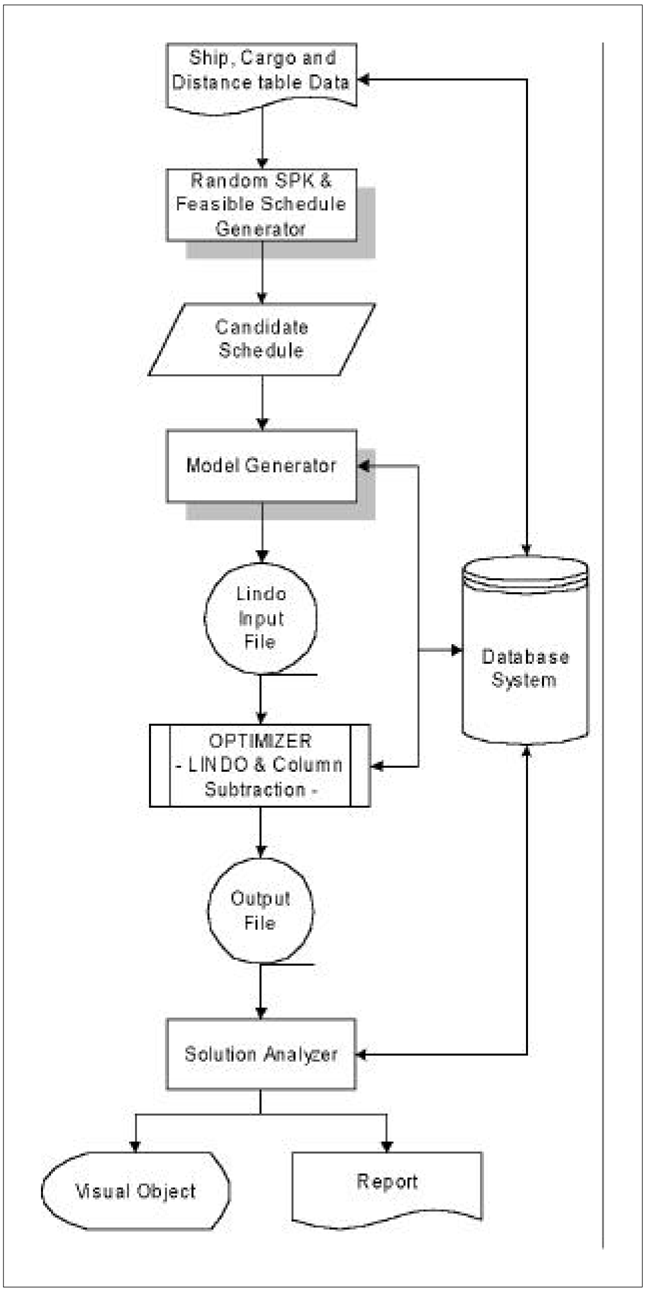
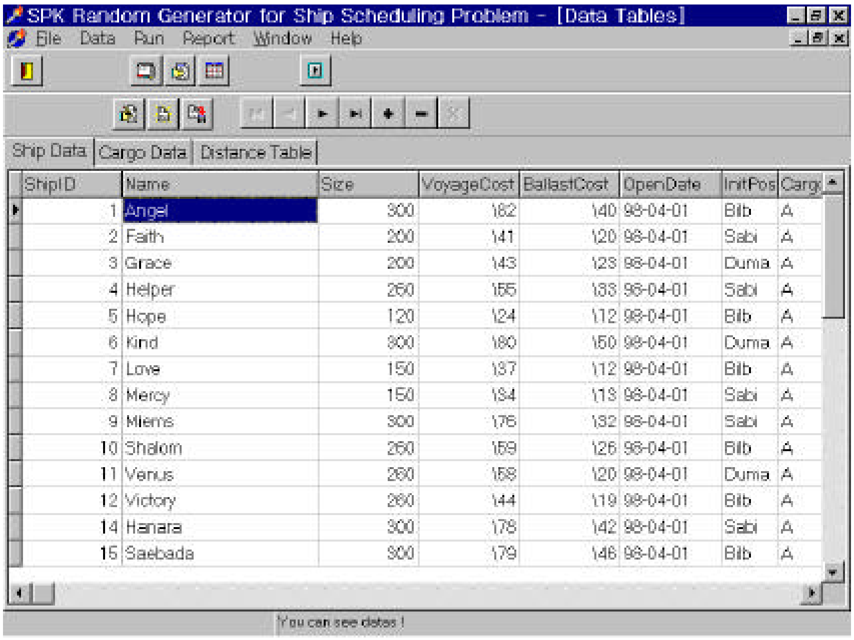


Fig. 3.1 System Flowchart

3.1.2 기본 입출력 사양

시스템의 입력화면은 GUI를 통하여 주요 선박, 화물 및 항만간 거리에 관한 자료들을 준비된 파일을 불러들이거나, 마우스와 키보드를 이용하여 편집할 수 있도록 하였다. Fig. 3.2의 화면은 델파이4의 "DB Engine"인 파라독스 데이터베이스를 이용한 초기자료이다.



ShipID	Name	Size	VoyageCost	BallastCost	OpenDate	InitPos	Carg
1	Angel	300	\182	\140	98-04-01	Bilb	A
2	Faith	200	\141	\120	98-04-01	Sabi	A
3	Grace	200	\143	\123	98-04-01	Duma	A
4	Helper	260	\155	\133	98-04-01	Sabi	A
5	Hope	120	\124	\112	98-04-01	Bilb	A
6	Kind	300	\180	\150	98-04-01	Duma	A
7	Love	150	\137	\112	98-04-01	Bilb	A
8	Mercy	150	\134	\113	98-04-01	Sabi	A
9	Miems	300	\176	\132	98-04-01	Sabi	A
10	Shalom	260	\159	\126	98-04-01	Bilb	A
11	Venus	260	\158	\120	98-04-01	Duma	A
12	Victory	260	\144	\119	98-04-01	Bilb	A
14	Hanara	300	\178	\142	98-04-01	Sabi	A
15	Saebada	300	\179	\146	98-04-01	Bilb	A

Fig. 3.2 초기 입력자료 정보에 관한 인터페이스

이렇게 입력된 초기자료들은 난수발생기를 거쳐 무작위로 변경된다. 변경된 초기 입력자료들을 바탕으로 후보 운항가능일정을 만들어 LP-완화된 최적화모형으로 구축한 뒤, LINDO Optimizer를 이용하여 모형의 해가 정수인지의 여부를 판별한다. 그 해가 정수인 경우에는 그대로 결과를 출력하지만, 소수인 경우에는 LP-완화제약식을 정수제약식으로 수정하여 정수계획 최적화모형으로 재구성하여 다시 원문제의 해를 구한다.

Fig. 3.3는 완화된 최적화 모형이 소수해로 판명된 결과를 출력한 화면이며, Fig. 3.4은 정수계획 최적화 모형으로 재 구성하여 컬럼 서브트랙션 해법을 이용하여 해를 구한 결과를 출력한 화면이다.

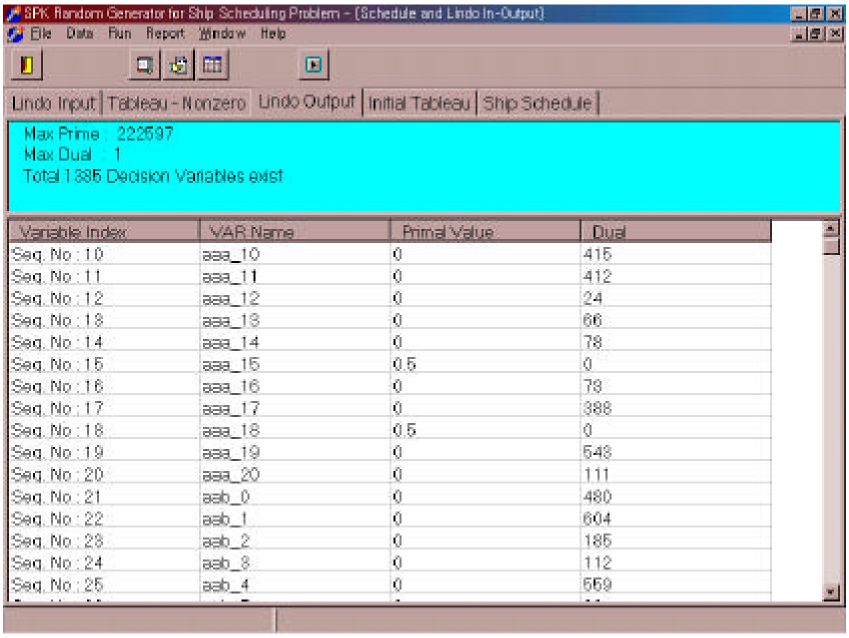


Fig. 3.3 LP-완화된 모형이 소수해인 결과

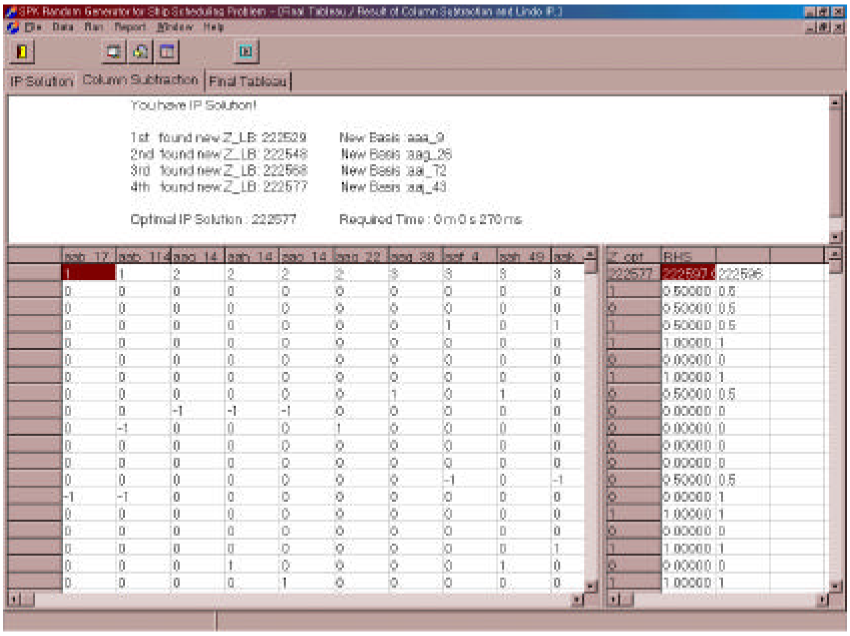


Fig. 3.4 정수계획 최적화 모형에 컬럼 서브트랙션 해법을 적용하여 해를 구한 결과

3.2 운항가능일정 생성 그래프와 알고리즘

선박 운항일정계획 문제의 최적화모형을 구축하려면 의사결정변수인 임의 선박의 모든 운항가능일정을 찾는 것이 선결과제가 된다.

이 운항가능일정을 생성하기 위해 Appelgren(1969)은 준비한 자료들로부터 모든 운항가능일정을 열거하였고, McKay와 Hartley(1974)는 각 선적항에서 제품유를 공급할 수 있는 도착항들의 표를 만들어 운항가능항로를 구축하였다. Brown 등(1987)은 인접한 시드일정(seed column)들 중 연결이 가능한 것은 서로 합하여 새로운 일정에 삽입하는 과정을 반복하여 운항가능일정 생성하는 프로그램을 사용하였으며, Fisher와 Rosenwein(1989)은 운항일정 상의 공선항해 거리의 최대허용치를 고려하면서 임의 선박의 후보 운항일정을 단계적추가 과정을 통해 생성하였는데, 이는 Brown 등(1987)의 기법과 유사한 것이다. Kim & Lee(1997)는 최적화모형을 기반으로 하는 선박 운항일정계획 의사결정지원시스템을 개발하면서 임의 선박에 대하여 운항가능일정 생성 그래프를 만들고, 이 그래프를 이용하여 임의 선박의 모든 운항가능일정을 생성하는 새로운 알고리즘을 고안하였다.

3.2.1 운항가능일정 생성 그래프

운항가능일정 생성 그래프란, 임의의 선박에 대하여 그 선박의 모든 운항가능한 일정을 생성하는 데 사용할 수 있도록 정의된 그래프이다.

만약 임의의 선박 i가 계획기간 내에 수송할 수 있는 모든 화물의 집합 $C(i)$를 구할 수 있다면, 선박 i에 대해 고유하게 정의되는 운항가능일정 생성그래프 $Gi(Ni, Ai)$는 Fig. 3.5와 같은 방향성 비순환 그래프가 된다. 이 때, 마디들의 집합 Ni는 i선박의 초기 투입을 위한 대기상태를 의미하는 시작마디 s, i선박에 의해 수송이 가능한 화물을 의미하는 화물마디 집합 C, 그리고 i선박의 계획기간 중 임무완료를 의미하는 종료마디 t로 이루어진다. 임의의 화물마디 $ck \in C$는 각 화물의 선적일에 관하여 오름차순으로 정렬한 것이다.

가지들의 집합 Ai 중 시작마디 s와 모든 화물마디 집합 C와 연결되어 있는 방향성 가지 (s, C)는 i선박이 임의의 화물을 그 운항일정의 첫 번째 화물로 수송할 경우, 초기위치에서 그 선적항까지의 공선항해를 의미한다.

그리고, 모든 화물 마디 집합 C와 종료마디 t가 연결되어 있는 방향성 가지 (C, t)는 i선박이 해당화물을 그 운항일정의 마지막 화물로 수송한 뒤 다른 화물에 재배정되지 않음을 의미한다.

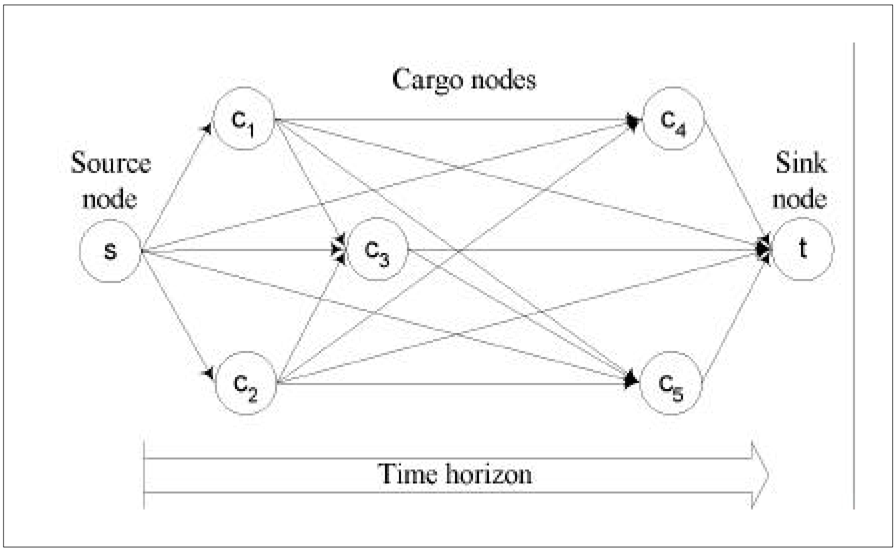


Fig. 3.5 A Feasible Scheduel Generating Graph $Gi(Ni, Ai)$ for Ship i

화물마디들 간에 연결되어 있는 가지들은 i선박이 임의의 화물을 수송한 후에 또다른 화물의 수송을 위해 재배정될 수 있을 때 정의된다. 이 때 화물마디들 간의 가지의 방향은 화물의 양하일과 선적일에 의해 결정된다. 운항가능일정 생성그래프 Gi의 시작마디에서 종료마디에 이르는 모든 가능한 경로는 선박 i의 운항가능일정이 된다.

Kim & Lee(1997)는 그래프 Gi를 컴퓨터가 인식할 수 있는 적절한 자료구조로 표현하기 위하여 연속 인접 리스트(Contiguous adjacency lists)방식을 사용하였으며, 본 시스템에서도 그래프 Gi를 표현하기 위하여 Kim & Lee(1997)가 고안한 알고리즘을 수용하였다.

3.2.2 운항가능일정 생성 알고리즘

운항가능일정 생성그래프 Gi에서 임의 선박 i의 모든 운항가능일정을 구

하는 것은 시작마디에서 종료 마디에 이르는 모든 가능한 경로를 구하는 것과 같다.

그래프 *Gi*의 시작마디에서 종료 마디에 이르는 모든 가능한 경로는 깊이우선탐색 그래프 순회 알고리즘을 다음과 같이 변형하여 적용함으로서 구할 수 있다.

(1) 그래프 *Gi*의 순회를 반드시 시작마디에서 출발한다.

(2) 깊이우선탐색 알고리즘으로 그래프 *Gi*의 마디들을 방문하되 방문한 마디들을 차례대로 스택에 저장(push)한다.

(3) 순회 중 종료마디를 방문하면 그 때까지 스택에 저장된 마디들을 순서대로 나열하여 하나의 운항가능일정으로 기록한다.

(4) 순회가 끝날 때까지 이 과정을 반복하되 순회 중 "backtrack"후진탐색이 일어날 때마다 스택에 저장된 마디들을 위에서부터 하나씩 삭제(pop)한다.

Kim & Lee(1997)는 모든 운항 가능일정을 구하기 위한 알고리즘 AFSG(*G*)를 고안하였으며, 본 시스템에서도 운항 가능일정을 구하기 위하여 알고리즘 AFSG(*G*)를 수용하였다.

Kim & Lee(1997)는 알고리즘 AFSG(*G*)의 주된 실행시간은 운항가능일정을 저장하는 반복횟수에 달려있다고 하였으며, 어떤 운항가능일정 생성 그래프 $G(N,\ A)$에 대하여 알고리즘의 실행시간을 다음과 같은 "Θ-notation"으로 나타내었다.

$$\sum_{n \in N} |\ Adj[n]\ | = \Theta(A)$$

또한, 집합유형 최적화 모형의 의사결정변수를 이러한 알고리즘으로 생성하면서 그 집합 유형 최적화 모형의 제약식의 계수 행렬의 "density"도 계산하였다.

즉, m척의 선박으로 n개의 선택적 화물을 수송하고자 하는 선박운항일정계획 최적화 모형을 가정할 때, 선박 운항일정계획 최적화 모형에 대한 제약식의 "density" ρ와 화물 제약식만의 "density" ρc를 각각,

$$\rho = \frac{\sum_i |J_i| + \sum_k q_{ijk}}{(m+n) \times \sum_i |J_i|}, \quad \rho_c = \frac{\sum_k q_{ijk}}{n \times \sum_i |J_i|}$$

와 같이 구하였다.

Fig. 3.6은 알고리즘 AFSG(G)를 그림으로 표현한 것이다.

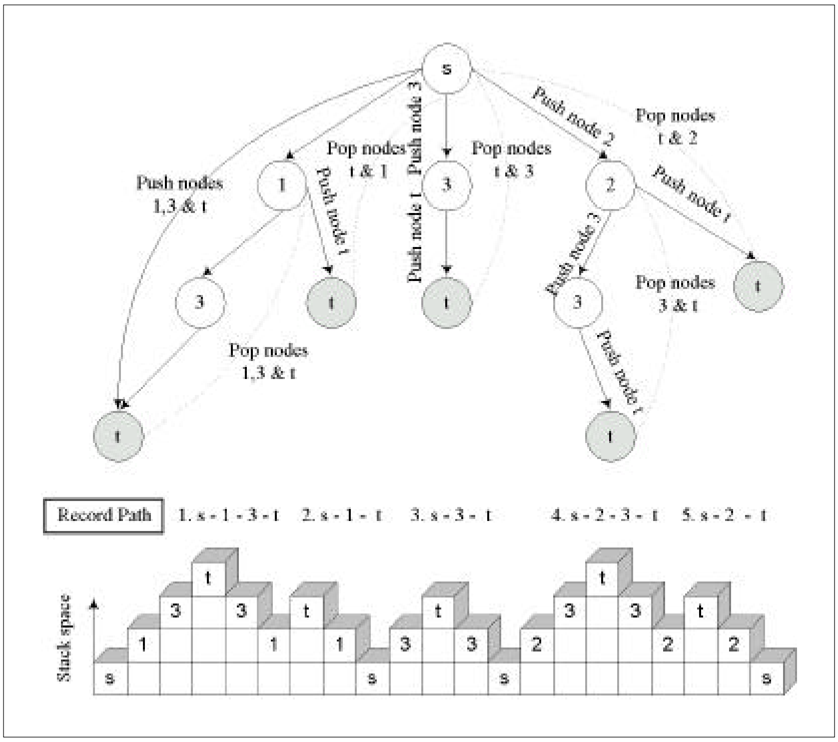


Fig. 3.8 Schematic Depth-first Search Tree in Algorithm AFSG(G)

3.2.3 운항가능일정의 자료구조

시스템에서 사용되는 초기 입력자료는 선박 자료, 화물 자료, 그리고 항만간 거리 자료이다. 이러한 초기 입력 자료 구조의 정의는 최적화 모형에 알맞게 정의하면 된다. 그리고 최적화 모형의 의사결정변수 즉, 선박 운항가능일정은 3.2.1과 3.2.2에서 설명한 바대로 연속 인접리스트(Contiguous adjacency lists)방식을 사용하여 구현하였다. 그러므로, 이 시스템을 이용하여 얻어진 LP-완화문제 또는 정수계획 최적화모형의 최적해는 후보 운항가능일정 중 제약조건을 만족하며 선택된 운항일정을 의미한다.

Fig. 3.7은 각 선박의 후보 운항일정계획이 연속 인접리스트(Contiguous adjacency lists)방식으로 표현된 것과 선박 운항일정계획 최적화 모형의 해 즉, 임의 선박이 투입될 운항일정을 트리구조로 나타낸 화면이다.

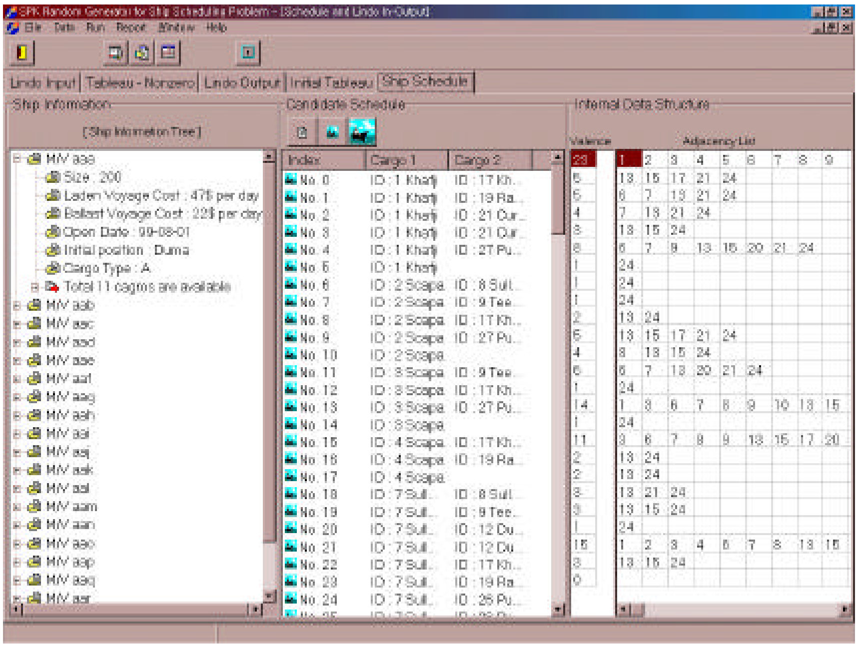


Fig. 3.7 각 선박이 투입되는 운항일정과 후보 운항일정.

3.3 컬럼 서브트랙션 알고리즘

이번 절에서는 운항 가능일정 생성알고리즘을 이용하여 구축된 선박 운항일정계획 최적화모형의 해를 구하기 위한 컬럼 서브트랙션 해법을 전산 구현한다.

다음의 Algorithm CSA_SPK(S)는 컬럼 서브트랙션 알고리즘(Column Subtraction Algorithm for Set Packing Problem)을 의사코드(pseudocode)의 형태로 표현한 것이다.

Algorithm CSA_SPK(S)

```
S_RSPK(S)          ▷ Solve the Relaxed SPK(S)
T[][] ← NIL        ▷ Initialize the entry of final tableau for RSPK
E[][] ← NIL        ▷ Initialize the entry of Cumulous Column
Slack[] ← NIL      ▷ Initialize the List of Slack Variables
```

```
ZUB, ZLB ← 0          ▷ Initialize the Upper and Lower Bound
point[] ← NIL         ▷ Initialize the Column Index of Slack List
scol ← 0              ▷ Initialize the Cumulous Coumn Index
acol ← 0              ▷ Initialize the Column Index of final Tableau
i, j, index ← 0       ▷ Initialize the Index for Row and Column
n ← 0                 ▷ Initialize the Counter
save_index[] ← NIL       ▷ Initialize the List of Saved Index
ZUB ← a Rounded Off Value of Relaxed SPK Solution
                      ▷ Set the Upper Bound
ZLB ← a Heuristic Solution, If any
                      ▷ Set the Lower Bound
for Final Tableau of Relaxed SPK(S)
  {
          n ← n + 1
          i ← i + 1
          j ← j + 1
          T[i][j] ← the [i, j]th entry of final tableau
                      ▷ Set the entry of final tableau for RSPK
          Slack[n] ← Slack Variable's Value
                      ▷ Set the List of Slack Variables for Subtracting
          if(Check Slack Variable's Value)
                      point[n] ← Index for the Slack List
                      ▷ Set the Index
  }
for each row
          E[i][1] ← T[i][j]
                      ▷ Set the first entry of Cumulous Column with
                        the last Column of Final Tableau
scol, index ← 1       ▷ Set the Index
acol ← point(1)       ▷ Set the Column Index for Slack List
ForwardStep(S)
```

```
FowardStep(S)

scol ← scol + 1
for each row I
  E[i][scol] ← E[i][scol-1] - T[i][acol]
save_index[scol] ← index
index ← index + 1
IntegralityCheck(S)
```

```
IntegralityCheck(S)

for each row
  if(Check Nonnegative Integer Solution)
  {
          ZLB ← the Nonnegative Solution
          BackTrack(S)
  }
  else
          CheckStep(S)
```

```
CheckStep(S)

if(Check the Number of Slack List)
  BackTrack(S)
acol ← point[index]
for Last row
  if(Check New Lower Bound Condition)
          BackTrack(S)
  else
          ForwardStep(S)
```

```
BackTrack(S)

index ← save_index[scol] + 1
scol ← scol - 1
if(scol > 0)
  CheckStep(S)
else
  Print ZLB
```

Algorithm CSA_SPK(S)의 1 11행은 초기화단계를, 12 13행은 해집합의 상한과 하한을 구하는 과정을, 14 23행은 최종테이블과 실행 가능한 해집합을 도출해 주는 "slack variable" 리스트를 작성하는 단계를 나타내며, 24 28행은 우변열을 저장한 뒤 "slack variable"을 이용하여 새로운 해를 구하기 위한 준비단계를 의미한다.

제 4 장　전산실험 결과 및 검토

이번 장에서는 제3장에서 구현된 시스템을 이용하여 컬럼 서브트랙션 해법과 LINDO Optimizer에서 사용하는 "branch-and-bound" 해법의 수행도를 비교하여 그 결과를 검토하고자 한다.

4.1 전산실험 방법

먼저 초기 입력자료는 데이타베이스 툴을 이용하여 구성하고, 시스템을 구동시킬 때마다 난수를 발생시켜 그 초기 입력자료들의 내용을 임의로 재구성하게 된다. 자료의 수는 데이터베이스 툴을 사용하여 얼마든지 변경이 가능하도록 하였다. 초기 입력자료인 선박, 화물, 항만간 거리에 관한 데이타의 예들은 각각 [표 4-1] [표 4-3]에 나타나 있다.

[표 4-1] 선박 자료

Ship ID	Name	Size	Laden Cost	Empty Cost	Initial Date	Initial Port	Type
1	Angel	300	$82	$46	99-04-01	Bilb	A
2	Faith	200	$41	$20	99-04-01	Sabi	A
3	Grace	200	$43	$22	99-04-01	Duma	A
4	Helper	260	$55	$30	99-04-01	Sabi	A
5	Hope	120	$24	$16	99-04-01	Bilb	A
6	Kind	300	$80	$52	99-04-01	Duma	A
7	Love	150	$37	$24	99-04-01	Bilb	A
8	Mercy	150	$34	$22	99-04-01	Sabi	A
9	Miems	300	$76	$38	99-04-01	Sabi	A
10	Shalom	260	$59	$28	99-04-01	Bilb	A

선박 및 화물자료의 일련번호(ID)는 자료의 순서에 따른 번호이고, 크기(Size)는 임의규격(1000 DWT)으로 통일해 놓은 것이다. 운임(Profit), 만선항해 시 운항비(Laden Cost), 공선항해 시 운항비(Empty Cost)는 US Dollar($/일) 단위로 작성한 것이며, 항만간 거리자료는 실제 거리를 1일 단위로 작성한 것이다. 선적항(Load. Port), 양하항(Disch. Port), 초기 투입항(Initial Port)은 항구명의 처음 넉자로 축약하였다. 선박유형(Ship's Type)

은 임의 선박이 수송가능한 화물의 유형(Cargo Type)에 따라 정해지지만 실험을 용이하게 하기 위하여 같은 유형으로 통일하였다.

[표 4-2] 화물 자료

Cargo ID	Name	Size	Profit	Load Port	Load.Date	Disch Port	Disch.Date	Type
1	Khafji	110	$96	Khaf	99-04-18	Bilb	99-05-09	A
2	Scapa	200	$69	Scap	99-05-26	Bilb	99-06-01	A
3	Sullo Voe	100	$35	Sull	99-04-28	Bilb	99-04-30	A
4	Teesport	100	$32	Tees	99-04-16	Bilb	99-04-21	A
5	Dubai	97	$45	Duba	99-04-11	Duma	99-04-24	A
6	Khafji	300	$157	Khaf	99-05-14	Duma	99-05-28	A
7	Ras Tanura	300	$153	Rast	99-05-01	Duma	99-05-15	A
8	Curacao	253	$80	Cura	99-04-13	Sabi	99-04-21	A
9	Puerto lacruz	150	$69	Puer	99-05-12	Sabi	99-05-22	A
10	Ras Tanura	101	$129	Rast	99-05-10	Sabi	99-06-13	A
11	Trinidad	250	$90	Trin	99-06-12	Sabi	99-06-22	A
12	Khafji	150	$85	Khaf	99-04-16	Bilb	99-05-01	A
13	Scapa	150	$120	Scap	99-04-17	Bilb	99-04-29	A
14	Trinidad	200	$105	Trin	99-04-19	Sabi	99-05-06	A
15	Teesport	250	$100	Tees	99-04-23	Sabi	99-01-09	A

[표 4-3] 항구간 거리 자료 단위 : 일(Day)

Port Name	Bilbao	Dumai	Sabinda
Curacao	14.5	35.6	7.8
Dubai	19.8	12.7	32.2
Khafji	21	13.9	33.4
Puerto la curz	14.2	35.1	9.6
Rastanura	20.8	13.6	33.1
Scapa	5.5	27.5	17.3
Sullomvoe	5.7	29.3	17.6
Teesport	4.9	28.5	17.8
Trinidad	13.7	34.6	9.1

먼저 제3장에서 설명한 원리에 따라 위의 자료들을 바탕으로 각 선박에 대한 운항 가능일정 그래프를 작성하고 가능한 운항일정을 목적함수의 변수로 만든다. 이 때, 화물과 선박의 크기(Size), 양하일(Disch. Date)과 적하

일(Load. Date), 항만간 거리를 각각 비교하여 화물의 선적가능 여부를 판별하며, 화물의 운임에 공선항해 시와 만선항해 시의 운항비를 감하여 목적함수의 변수에 대한 계수로 사용한다. 이렇게 구축된 최적화 모형은 "simplex 초기테이블"의 형태로 출력이 가능하다.

다음으로 운항가능일정 그래프를 바탕으로 각 선박과 화물에 대한 제약식을 구성한다. 이렇게 만들어진 정수계획 최적화 모형의 LP-완화 문제의 해를 휴리스틱한 방법을 이용하여 구한다.

정수해가 나올 경우에는 그 해를 그대로 화면에 나타내고, 소수해가 나올 경우에는 원문제에 대하여 LINDO의 "branch-and-bound" 해법과 컬럼 서브트랙션 해법을 각각 적용하여 해를 구한다. 컬럼 서브트랙션 해법 이용되는 최종 테이블은 LP-완화 문제를 풀 때 LINDO에 의하여 생성된 결과를 이용하였다.

해법의 유효성을 검증하기 위하여 다양한 방법을 사용할 수 있지만, 본 논문에서는 해를 구하는데 소요되는 계산시간을 측정하여 유효성을 검증하는 방법을 사용하였다. 그러나, 선박 운항일정계획 모형의 해를 구하기 위한 시간을 측정하기 위하여 시스템을 구동시켜 그 결과를 확인하기까지 수분에서 수십분에 이르기까지 많은 시간이 소요되었다. 그 이유는 "branch-and-bound" 해법 또는 컬럼 서브트랙션 해법을 이용하여 해를 구하는데 요구된 시간보다는 사용자에게 친숙한 환경을 제공하기 위한 GUI의 처리와 운항 가능일정 그래프등을 생성하는데 더 많은 시간이 요구되었기 때문이다. 그 결과 필요한 메모리의 양의 요구도 급증하였다. 이러한 이유들로 인하여 순수하게 해를 구하는데 소요된 시간만을 측정하여 평가기준으로 정하였다.

본 전산실험에 사용된 선박 자료들은 10척 30척, 화물 자료들은 10개 50개로 구성하였으며, 각 자료들에 대한 수치들은 난수에 의하여 변경되지만, 가능한한 현실문제와 유사하도록 구성하였다. 이러한 자료들을 다양한 방법으로 조합하여 500회 이상으로 반복실험을 행하였다.

4.2 결과 및 검토

계산실험의 결과 소수해는 Appelgren(1969)의 1 2%와 비슷하게 본 실험에서는 약 1 3%의 낮은 비율로 존재하였으며, 선박 운항일정계획 최적

화 모형이 구축되었을 때 모형내의 제약식, 변수, "non-zero", "density"등에 대한 자료들은 [표 4-4] [표 4-6]와 같다.

[표 4-4] 10척의 선박을 조합하여 구축된 모형의 자료

Ship & Cargo		Constraint	Variables	Non-Zero	Density
10 × 10	Min	16	33	72	0.11447
	Max	20	245	710	0.15317
	Ave	18.65	110.36	278.05	0.13068
10 × 20	Min	23	120	279	0.07616
	Max	30	1031	3267	0.12043
	Ave	27.71	389.87	1092.97	0.09961
10 × 30	Min	32	253	617	0.06612
	Max	40	1932	6656	0.09796
	Ave	36.57	852.51	2543.19	0.08011
10 × 40	Min	39	607	611	0.00682
	Max	50	4405	15412	0.08181
	Ave	45.42	1545.15	4788.61	0.06734
10 × 50	Min	45	786	2228	0.05056
	Max	59	5038	17546	0.07722
	Ave	54.31	2524.76	8139.98	0.05865

[표 4-5] 20척의 선박을 조합하여 구축된 모형의 자료

Ship & Cargo		Constraint	Variables	Non-Zero	Density
20 × 10	Min	25	67	129	0.06876
	Max	30	624	1798	0.10851
	Ave	28.85	234.31	589.03	0.08548
20 × 20	Min	34	262	628	0.06135
	Max	40	2028	6538	0.08616
	Ave	37.96	792.89	2229.48	0.07268
20 × 30	Min	40	560	1506	0.05332
	Max	50	4319	14747	0.08164
	Ave	46.44	1645.45	4934.54	0.06348
20 × 40	Min	47	939	2473	0.04589
	Max	60	7170	25761	0.06601
	Ave	55.56	3106.54	9687.33	0.05529
20 × 50	Min	57	1666	4634	0.04266
	Max	70	15146	56058	0.06101
	Ave	64.39	5029.32	16252.30	0.04934

제약식의 수는 투입된 선박수와 수송할 수 있는 화물수의 합으로 표현되며, 변수의 수는 운항가능일정을 의미한다.

"non-zero"의 수는 구축된 최적화 모형의 행렬 A에 나타나는 1의 개수를, "density"는 3.2.2에서 설명한 방법으로 모든 제약식에 대해 구한 것이다.

[표 4-6] 30척의 선박을 조합하여 구축된 모형의 자료

Ship & Cargo		Constraint	Variables	Non-Zero	Density
30 × 10	Min	36	115	248	0.05000
	Max	40	829	2526	0.07912
	Ave	38.95	366.45	927.53	0.06371
30 × 20	Min	44	529	1319	0.04896
	Max	50	2800	9044	0.06973
	Ave	47.73	1175.92	3312.66	0.05799
30 × 30	Min	49	933	2375	0.04450
	Max	60	6655	23766	0.06458
	Ave	56.55	2574.10	7722.46	0.05211
30 × 40	Min	58	591	1257	0.03272
	Max	70	10290	36628	0.05692
	Ave	65.76	4178.05	12912.43	0.04592
30 × 50	Min	65	3131	8715	0.03650
	Max	79	15401	56257	0.05349
	Ave	73.08	7072.19	22768.82	0.04347

이 시스템에 의해 구축되는 모형들의 "density"의 평균값은 6.6%, 최소값은 4.3%, 최대값은 13.1%이었다. 변수의 최대 수는 15,401개였다.

다음의 [표 4-7]은 대표적으로 두 세가지의 예를 선정하여 최적화 모형의 자료들, 컬럼 서브트랙션 해법 및 LINDO에서 사용하는 Branch-and-bound 해법의 수행시간을, [표 4-8]은 "branch-and-bound" 해법과 컬럼 서브트랙션 해법의 수행도를 비교한 것이다.

[표 4-7]에서 알 수 있듯이 컬럼 서브트랙션 해법을 이용하였을 때 계산에 요구되는 시간이 "branch-and-bound" 해법을 이용하였을 때 보다도 확연히 줄어든 것을 알 수 있다. 평균 수행도 비율을 계산한 결과 컬럼 서브트랙션 해법을 이용하여 문제를 해결할 경우 "branch-and-bound" 해법에 비하여 9.4% 34.6%계산시간이 요구됨을 확인할 수 있다.

[표 4-7] 최적화 모형과 수행시간 비교 예

선박 × 화물	변 수	제약식	Density	LINDO(초)	Column(초)	시간비율
10 × 10	All Problems have Integer Solution					
10 × 20	338	25	0.11219	1.2	0.05	0.041
	549	28	0.02687	1.1	0.01	0.001
	367	26	0.11171	0.88	0.06	0.069
10 × 30	636	32	0.08820	0.99	0.05	0.051
	1086	37	0.08071	1.76	0.22	0.125
	989	33	0.09232	1.59	0.17	0.107
10 × 40	2253	47	0.06949	3.73	0.50	0.134
	2116	45	0.07310	4.23	0.55	0.130
	1605	50	0.06004	2.64	0.27	0.102
10 × 50	3279	53	0.06257	6.43	3.57	0.555
	2321	58	0.05486	5.77	1.64	0.284
	2069	57	0.05452	5.33	0.44	0.083
20 × 10	246	28	0.09858	0.44	0.05	0.114
20 × 20	All Problems have Integer Solution					
20 × 30	1308	49	0.05604	3.95	0.50	0.127
	3050	47	0.07053	9.12	1.53	0.168
	1835	43	0.07119	2.25	0.27	0.120
20 × 40	2877	55	0.05506	8.84	2.36	0.267
	5794	57	0.05850	27.91	20.21	0.724
	2877	55	0.05506	8.84	2.36	0.267
20 × 50	3022	61	0.04907	10.77	3.46	0.043
	4855	64	0.04863	14.55	4.89	0.336
	5966	67	0.04765	27.85	3.57	0.128
30 × 10	All Problems have Integer Solution					
30 × 20	790	48	0.05367	1.32	0.22	0.167
	1054	47	0.05664	1.75	0.17	0.097
30 × 30	2529	59	0.04912	4.60	0.50	0.102
	2761	54	0.05702	4.39	0.61	0.139
	3168	60	0.05125	5.16	0.72	0.140
30 × 40	3797	66	0.04478	11.15	3.07	0.275
	3555	66	0.04479	14.28	3.13	0.219
	3094	66	0.04362	8.96	1.75	0.195
30 × 50	9112	71	0.04824	15.16	6.43	0.424
	6149	75	0.04122	10.11	2.80	0.277
	6896	76	0.04256	11.7	3.13	0.268

[표 4-8] "branch-and-bound"와 컬럼 서브트랙션 해법의 수행시간 비교

선박 × 화물	branch-and-bound(초)			Column Subtraction(초)			수행도 비율 (%)		
	min	max	ave	min	max	ave	min	max	ave
10x20	0.210	1.650	0.925	0.001	0.280	0.076	0.1	31.8	9.4
10x30	0.940	2.030	1.429	0.050	0.380	0.187	5.1	18.7	12.7
10x40	1.370	4.230	2.861	0.110	0.880	0.405	8.0	51.8	15.1
10x50	3.020	6.430	5.061	0.380	3.570	1.176	8.3	55.5	21.4
20x10	0.440	0.440	0.440	0.050	0.050	0.050	11.4	11.4	11.4
20x30	2.250	11.870	5.932	0.270	2.360	0.901	9.1	59.7	15.9
20x40	5.880	27.910	11.675	0.770	20.210	4.978	13.1	72.4	34.6
20x50	9.280	160.870	28.461	1.260	25.320	6.600	2.3	89.4	33.8
30x20	1.320	1.750	1.535	0.170	0.220	0.195	9.7	16.7	13.2
30x30	1.980	5.390	4.196	0.160	1.480	0.694	8.1	27.5	15.1
30x40	7.850	14.280	10.954	1.430	3.130	2.501	18.2	27.6	22.6
30x50	7.910	15.160	10.840	1.600	6.430	3.203	18.4	42.4	27.6

본 실험의 결과 집합 문제 유형의 선박 운항일정계획 최적화 문제에 대한 해법으로는 컬럼 서브트랙션 해법이 "branch-and-bound" 해법에 비하여 유용함을 알 수 있다. 계산시간의 절감 이외에도 컬럼 서브트랙션 해법을 이용할 경우 기대되는 효과는 다음과 같은 것들이 있다.

첫째, 동일한 최적해를 제공해 주는 복수의 운항가능일정 해집합을 구할 수 있다. 동일한 최적해를 갖는 해집합들에 대하여 Lindo의 "branch-and-bound" 해법은 수행될 때 마다 다른 해들을 구해준다. 그 이유는 "pivoting" 과정 중 동일한 "ratio"를 갖는 피봇 엘리먼트(pivot element)를 무작위로 선택하기 때문이다. 그러나 컬럼 서브트랙션 해법은 제2장의 원리에서 알 수 있드시, 최적해를 제공하는 최종 테이블상에서 수정비용의 값이 같거나 “0”인 비기저 변수들 중 정수해를 제공하는 변수들을 선택하면 복수의 해를 구할 수 있게 된다.

둘째, 다양한 의사결정에 대한 옵션(Option)을 제공해 준다. LINDO Optimizer의 경우에는 최적해만을 도출해 줄 수 있을 뿐, 여러가지 상황의 변동으로 인하여 어느 운항 가능일정이 취소되었을 경우에 대안을 제공해

주지 못한다. 그러나 본 논문 제2장에서 컬럼 서브트랙션 해법을 예제에 적용해 보면서, 하한을 갱신함으로써 최적해에 도달해 갈 수 있음을 알 수 있었다. 그러므로, 최적해와 그 차선책들을 차례로 열거할 수 있게 된 것이다.

제 5 장 결 론

해운문제의 특성인 불확실성, 문제구조의 다변성, 해운기업의 보수성, 해운시장의 국제성, 개방성, 불안정성 등을 잘 반영하며, 과학적이고 합리적인 의사결정을 지원할 수 있는 선박 운항일정관리 지원시스템은 사용자들에게 필요한 정보들을 가능한한 많이 지원할 수 있어야 한다.

그러나, 지금까지 개발 된 의사결정 지원시스템은 최적화 모형의 문제를 해결하기 위해 LINDO Optimizer등과 같은 상용프로그램에 의존하고 있는 형편이며, 그러한 상용프로그램은 사용자들에게 최적해에 관한 다양한 정보를 제공하기 어렵다. 그러므로, 해운문제의 특성을 잘 반영하며 사용자들의 요구에 부합되는 정보들을 제공할 수 있는 적절한 해법에 대한 연구가 필요한 실정이다.

이 연구에서는 해사수송 문제와 밀접한 관련이 있는 선박 운항일정 계획문제에 대하여 집합 문제의 해법중 하나인 컬럼 서브트랙션 해법을 적용하여 해결하였고, 의사결정 지원시스템을 구현하여 그 해법에 대한 수행도를 비교 · 분석하였다.

이 연구에서 집합 패킹 문제 적용한 컬럼 서브트랙션 해법의 전산구현의 결과로 얻어진 성과는 다음과 같이 요약할 수 있다.

첫째, 집합 문제 유형의 선박 운항일정계획 최적화 모형의 해법으로 컬럼 서브트랙션 해법이 "branch-and-bound" 해법에 비하여 우수함을 전산실험을 통하여 확인하였다.

둘째, 컬럼 서브트랙션 해법을 이용하여 사용자들에게 복수의 해와 다양한 의사결정 옵션을 제공할 수 있음을 확인하였다.

셋째, 사용자들에게 친숙한 형태의 GUI환경의 의사결정 지원시스템을 구현하였으며, 객체 지향언어의 특징들을 이용하여 다른 해법들을 적용할 수 있는 시스템으로 손쉽게 개선할 수 있다.

이상에서 살펴본 연구 결과를 통해서 집합 패킹 문제에 적용한 컬럼 서브트랙션 해법의 유효성을 확인하였다. 그러나, 집합 문제에 대하여 접근법에 따른 다양한 모든 해법들에 대하여 비교 · 분석하지 못한 이유들로 인하여 컬럼 서브트랙션 해법만이 최선이라고 말할 수 없다. 그러므로, 다양한 해법들을 비교하여 해운문제의 특성을 잘 반영할 수 있는 최선의 해법을 찾아 의사결정지원 시스템에 탑재하는 일 등이 향후 연구해야 할 과제이다.

참고문헌

(1) 강맹규, 네트워크와 알고리즘, 박영사, 1991.

(2) 김세헌, 현대 경영과학, 무역경영사, 1994.

(3) 엄기현 · 홍영식, 자료구조 개론, 정익사, 1993

(4) Goldberg David E., Genetic algorithms in search, optimizatio, and machine learning, Addison-Wesley Pub. Co 1989

(5) Garfinkel Robert., Integer Programming, Wiley, 1972

(6) Murty Katta G., Operations Research: Deterministic optimization model, Prentice Hall Inc., 1995.

(7) Gass Saul I., Linear Programming methods and applications, fifth edition, McGraw Hill Inc., 1985.

(8) Winston Wayne L., Operations Research Applications and Algorithms, *3rd Edition, International Thomson Pub.*, 1994.

(9) Appelgren L. H., "A Column Generation algorithm for a ship scheduling problem", *Transportation Science*, 3, pp. 53-68 (1969)

(10) Appelgren L. H., "A column generation algorithm for a ship scheduling problem", *Transportation Science*, 3, pp. 53-68 (1969)

(11) Appelgren L. H., "Integer programming methods for a vessel scheduling problem", *Transportation Science*, 5, pp. 64-78 (1971)

(12) Balas E. and Padberg M. W., "Set partitioning: A survey", *SIAM Review*, Vol 18, No. 4, pp. 710-760, Oct. (1976)

(13) Bellmore M., "A maximum utility solution to a vehicle constrained tanker scheduling problem", *Naval Research Logistics Quarterly*, 15, No. 3, pp. 403-411 (1968)

(14) Bellmore M., Bennington G. and Lubore S., "A multi-vehicle tanker scheduling problem", *Transportation Science*, 5, pp. 36-47 (1971)

(15) Brown G. G., Graves G. W. and Ronen D., "Scheduling ocean transportation of crude oil", *Management Science*, Vo. 33, No. 3, pp. 335-346 (1987)

(16) Dantzig G. B. and Fulkerson D. R., "Minimizing the number of tankers to meet a fixed schedule", *Naval Research Logistics*

Quarterly, Vol. 1, pp. 217-222 (1954)

(17) Rosenwein Fisher, M. L. and M. B., "An interactive optimization system for bulk-cargo Ship Scheduling", *Naval Research Logistics*, Vol. 36, pp. 27-42 (1989)

(18) Harche F. and Thompson G. L., "The column subtraction Algorithm: An exact method for solving weighted set covering, packing and partitioning problems", *Computers & Operations Research*, Vol. 21, No. 6, pp. 689-705 (1994)

(19) Si-Hwa Kim and Kyung-Keun Lee, "An Optimization-based decision support system for ship scheduling", *Computers & I.E., An Intl. Journal*, Vol. 33, pp. 689-692 (1997)

(20) Si-Hwa Kim, "Optimization-based Decision Support System for Some Maritime Transportation Problems", *Ph. D. Thesis, Dept. of Industrial Engineering*, Pusan National University (1999)

(21) Laderman J. and Gleiberman L., Egan J. F., "Vessel Allocation By Linear Programming", *Naval Research Logistics Quarterly*, Vol. 13, No. 3, pp. 315-320, Sep. (1966)

(22) McKay M. D., "Computerized scheduling of seagoing tankers", *Naval Research Logistics Quarterly*, 21, pp. 255-264 (1974)

(23) Ronen D., "Cargo Ships routing and scheduling: Survey of models and problems", *European Journal of Operational Research*, 12, pp. 119-126 (1983)

(24) Ronen D., "Ship Scheduling: The Last Decade", *European Journal of Operational Research*, Vol. 71, pp. 325-333 (1993)

(25) Sethi A. P. and Thompson G. L., "The pivot and probe algorithm for sloving a linear program", *Mathmatical Programming*, 29, pp. 219-233 (1984)

(26) Whiton J. C., "Some constraints on shipping in linear programming models", *Naval Research Logistics Quarterly*, 14, pp. 257-260 (1967)

감사의 글

먼저, 내 삶의 주인되시는 하나님께 감사드립니다.

삶을 살아나가면서 지식보다는 한 영혼을 사랑하는 것이 얼마나 고귀한 것인지 몸소 실천하며 가르쳐 주시고, 이 논문이 완성되기까지 정성을 다하여 부족한 저를 지도해 주신 김 시화 교수님께 감사드립니다.

여러 가지 일들로 바쁘신 중에도 불구하고, 보잘 것 없는 글을 꼼꼼하게 살펴주신 조 성철, 박 진수 교수님께도 감사드립니다.

또한, 변함없는 미소로 응원하신 위 승민, 장 일동 선배님과 불평 한마디 없이 프로그래밍에 대해 꼼꼼히 조언해 주신 이 희용 선배님, 그리고 늘 친구같은 곽 민석, 김 대상, 허 강이, 전 진우 후배님들께 진심으로 감사 드립니다.

저를 아끼시며 사랑과 관심을 보여주신 여러분들에게 이 자리를 빌어 감사의 뜻을 전하고 싶습니다. 그분들의 사랑과 관심이 없었더라면 배움과 가르침의 진정한 의미들을 깨우치지 못하였을 것입니다.

오늘도 인생의 뒤안길을 묵묵히 걷고 계신,
아버지께 이 논문을 바칩니다.