한국지능시스템학회 논문지 2011, Vol. 21, No. 1, pp. 138-144
DOI : 10.5391/JKIIS.2011.21.1.138

Simulated Annealing 알고리즘에 기반한 $L(2,1)$-labeling 문제 연구

Study on the $L(2,1)$-labeling problem based on simulated annealing algorithm

한근희 · 이용진
Keun-Hee Han and Yong-Jin Lee

공주대학교 응용수학과

요 약

그래프 $G = (V, E)$의 $L(2,1)$-labeling 은 무선통신에서 무선 기기에 할당되는 주파수를 효율적으로 사용하기 위한 최적화 문제로서 NP-complete 계열에 포함되는 문제이다. 본 연구에서는 $L(2,1)$-labeling 문제에 적용 가능한 Simulated Annealing 알고리즘을 제시한 후 다양한 그래프에 제시된 알고리즘을 적용하여 그 효용성을 보이고자 한다.

키워드 : 그래프, $L(2,1)$-labeling, Simulated Annealing 알고리즘

Abstract

$L(2,1)$-labeling problem of a graph $G = (V, E)$ is a problem to find an efficient way to distribute radio frequencies to various wireless equipments in wireless networks. In this work, we suggest a Simulated Annealing algorithm that can be applied to the $L(2,1)$-labeling problem. By applying the suggested algorithm to various graphs we will try to show the efficiency of our algorithm.

Key Words : Graph, $L(2,1)$-labeling, Simulated Annealing algorithm

1. 서 론

주파수 할당 문제(frequency assignment problem)는 무선 통신망에서 각 무선 기기에 할당되는 주파수를 허용 가능한 영역 내에서 효율적으로 분배하는 최적화 문제이다. 본 문제는 Hale [1]에 의해 최초로 T-coloring 이라는 그래프 이론의 정점 색칠 문제(graph coloring problem)의 일종으로 정립되었으나 이후 Roberts [2]는 근거리에 위치한 무선기기 사이에는 서로 다른 주파수를 할당하고 인접한 무선기기 사이에는 최소한 2 이상의 차이를 갖는 주파수를 할당하는 새로운 문제를 제기하였다. 본 문제는 1992년 Griggs [3]에 의하여 $L(2,1)$-labeling 문제로 정립되었으며 본 문제의 수학적 정의는 다음과 같다.

$G = (V, E)$를 정점 집합 $V = \{1, ..., n\}$ 및 간선 집합 $E \subseteq V \times V$를 갖는 단순 무방향 그래프라 하고 $d(u, v)$를 G의 두 개 정점 u, v 사이의 최단 거리라 하자. 그래프 G에 대한 $L(2,1)$-labeling 이란 다음 두 개 조건을 만족하는 함수 $f : V(G) \rightarrow \{0, 1, 2, ..., \lambda\}$를 정의하는 것이다.

(i) 만일 $d(u,v) = 1$ 이라면 $|f(u) - f(v)| \geq 2$, (A)
(ii) 만일 $d(u,v) = 2$ 라면 $|f(u) - f(v)| \geq 1$,

주어진 그래프 G에 대하여 함수 f가 상기한 조건 (A)를 만족하는 $L(2,1)$-labeling 이라면 $f \in L(2,1)(G)$이라 하고 $\|f(G)\| = \max\{f(v) \mid v \in V(G)\}$ 라 하자. 즉, $\|f(G)\|$는 G에 대한 $L(2,1)$-labeling 에서 사용된 가장 큰 정수를 나타낸다. 주어진 그래프 G 의 모든 $f \in L(2,1)(G)$에 대하여 G의 $L(2,1)$-labeling number 로 불리우는 $\lambda(G)$는 $\min \|f(G)\|$ 로 정의되며 그러한 $L(2,1)$-labeling f를 G에 대한 최적의 $L(2,1)$-labeling 이라 한다. 이를 다시 표현하면, 식 (A)를 만족하는 상태에서 그래프 내의 모든 정점들에 정수를 부여하되 부여되는 가장 큰 정수값을 최대한 낮추도록 하는 것이 $L(2,1)$-labeling 문제의 목적이며 이는 그래프의 각 정점들을 무선기기로 취급하면 전체 무선기기에 할당되는 전체 주파수의 대역폭을 최소화 시킬 수 있는 장점을 갖게 된다. 무선기기에 분배될 수 있는 주파수 대역은 물리적, 정책적 및 상업적으로 매우 한정된 자원이므로 주어진 주파수 대역 내에서 주파수를 효율적으로 분배하는 것은 매우 중요하다.

그래프 $G = (V, E)$에 대하여 식 (A) 의 두가지 조건을 만족하는 labeling을 유효한 $L(2,1)$-labeling 이라하자. 그림 1 은 4 개 정점 a, b, c 및 d 로 구성된 동일한 그래프 G에 대한 3 개의 $L(2,1)$-labeling f_1, f_2 및 f_3을 보여준다.

그림 1 (a) 의 경우 $d(b,d) = 1$ 이지만 $|f_1(b) - f_1(d)| = 1$ 이므로 f_1 은 유효하지 않는 $L(2,1)$-labeling 이다. 그림 1 (b) 의 f_2는 비록 유효한 $L(2,1)$-labeling 이지만 최적의 $L(2,1)$-labeling은 아니며 그림 1 (c) 는 최적의 $L(2,1)$-labeling을 보여주고 있다.

접수일자 : 2010년 10월 29일
완료일자 : 2011년 1월 30일
이 논문은 2009년 공주대학교 학술연구지원사업의 연구비 지원에 의하여 연구 되었습니다.

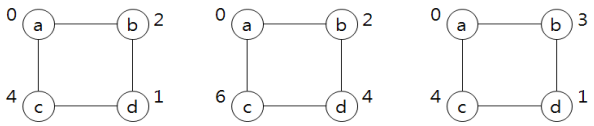

그림 1. 4 개 정점 a, b, c 및 d 로 구성된 그래프 G의 $L(2, 1)$-labeling

Fig. 1. $L(2,1)$-labeling on a graph G consisting of four vertices a, b, c and d

$G = (V, E)$를 단순 무방향 그래프라 하자. G 의 정점 v ($\in V$) 의 차수는 v에 인접한 간선의 개수를 나타내며 $deg(V)$ 로 표기된다. G 의 최대차수 $\Delta(G)$ 및 최소차수 $\delta(G)$ 는 $\Delta(G) = \max\{deg(v) \mid v \in V\}$ 및 $\delta(G) = \min\{deg(v) \mid v \in V\}$ 이라 각각 정의된다. G 의 직경(diameter) 은 $diam(G) = \max\{d(u,v) \mid u,v \in V\}$로 정의된다. 그래프의 정점 및 간선의 개수는 $|V| = n$ 및 $|E| = e$ 로 각각 표기된다. 그래프 G 의 인접행렬 (adjacency matrix) $A(G)$ 는 $n \times n$ 크기의 0-1 행렬로서 원소 $a_{ij} = 1$ 일 필요충분조건은 정점 i 와 정점 j가 서로 인접하다는 것이다.

주어진 그래프 $G = (V, E)$ 의 $\lambda(G)$ 를 계산하는 것은 NP-complete 계열에 속하는 문제로서 $diam(G) = 2$ 인 그래프의 $\lambda(G) \le |V|$를 판별하는 문제조차도 NP-complete 계열에 속하는 문제이다 [3]. 따라서, $L(2,1)$-labeling에 대한 연구는 그래프 G 의 최대차수를 나타내는 $\Delta(G)$ 를 매개 변수로 이용하여 다양한 일반 및 특수 그래프들에 대한 $\lambda(G)$ 의 하한 및 상한 값을 계산 하는데 치중되어 왔다. Griggs [3]는 일반 그래프 G에 대하여 $\lambda(G) \le \Delta(G)^2 + 2\Delta(G)$ 임을 제시하였지만 본 결과는 Chang [4] 에 의하여 $\lambda(G) \le \Delta(G)^2 + \Delta(G)$ 로 개선되었으며, 2003 년에는 Král [5] 에 의하여 $\lambda(G) \le \Delta(G)^2 + \Delta(G) - 1$ 로 개선되었다. 또한 2005 년에는 Goncalves [6] 에 의하여 본 결과는 $\lambda(G) \le \Delta(G)^2 + \Delta(G) - 2$ 로 개선되었다. 특수 그래프들에 관련된 $\lambda(G)$ 의 상한 및 하한 값들은 [11]에서 참조될 수 있다.

본 논문에서는 그래프 G 의 $L(2,1)$-labeling number 인 $\lambda(G)$ 의 근사값을 계산할 수 있는 Simulated Annealing (SA) 알고리즘을 제시하고자 한다. 제시된 알고리즘은 $\lambda(G)$ 의 최적값을 쉽게 계산할 수 있는 특수 그래프인 complete k-partite graph에 적용하여 제시된 알고리즘의 효용성을 검증하고자 하며 또한 제시된 SA 알고리즘이 이미 제안된 알고리즘 [7] 에서 제시된 유전 알고리즘 (Genetic Algorithm : GA) 의 결과보다 뛰어남을 보이고자 한다. 또한 본 연구에서 제시되는 알고리즘은 [7] 에서 제시된 알고리즘보다 계산 수행의 시간적 측면에서도 뛰어남을 실험을 통하여 보이고자 한다.

본 논문의 나머지 부분은 다음과 같이 구성되어 있다. 2 장에서는 Simulated Annealing 알고리즘에 대한 간략한 소개 및 $L(2,1)$-labeling 을 위하여 본 연구에서 제안하는 SA 알고리즘의 구조를 설명하며 이어서 3장에서는 본 연구에서 제안하는 SA 알고리즘을 다양한 complete k-partite graph에 적용한 실험 결과를 분석 한다. 끝으로 4장에서는 결론을 맺는다.

2. $L(2, 1)$-labeling 을 위한 Simulated Annealing 알고리즘

2.1 Simulated Annealing 알고리즘

본 논문에서 언급되는 모든 그래프들은 단순 무방향 그래프이며 SA 알고리즘 및 그래프 이론에 관련된 더욱 자세한 내용은 [8] 및 [9]을 각각 참조하도록 한다.

Simulated Annealing (SA) 알고리즘은 철을 조련하는 과정에서 철의 온도를 적절히 조절하는 상태에서 철을 담금질 하여 더욱 높은 품질의 철을 제련하는 방법을 모방하는 알고리즘으로서 1983 년 kirkpatrick [10]에 의해 최초로 개발되었다. 이러한 SA 알고리즘은 유전 알고리즘과 마찬가지로 NP-complete 계열에 속하는 다양한 최적화 문제를 계산하는데 주로 사용된다. SA 알고리즘은 다수의 해 집단을 취급하는 유전 알고리즘과는 달리 하나의 해 (solution) 만을 다루며 크게 쇠를 두들기어 변형시키는 과정 (move operation) 과 온도 (temperature)에 따라 해를 선택적으로 수용하는 적합도 함수 (evaluation function) 등으로 구성된다. 다음 그림 2 는 본 연구에서 사용된 SA 알고리즘에 대한 전반적인 흐름을 보여주고 있다.

```
1   Begin
2     T = T0 ;
3     Tstop = Ts ;
4     Current_Solution = Generation of a initial solution ;
5     While Tstop > 0 do
6       accept = False ;
7       For i = 1 to M do
8         New_Solution = Move (Current_Solution)
9         Δ Gain = eval (Current_Solution) - eval (New_Solution) ;
10        If Accept (Δ Gain , T ) then
11          Current_Solution = New_Solution ;
12          accept = True ;
13      If accept then
14        Tstop = Ts ;
15      Else
16        Tstop = Tstop - 1 ;
17      T = T × a ;
18  End
```

그림 2. Simulated Annealing 알고리즘

Fig. 2. Simulated Annealing Algorithm

그림 2 에서 초기 해의 구성은 난수를 이용하여 임의적으로 구성된다. T_0 는 초기온도, T는 현재온도이며 T_s 는 일정온도에서의 담금질 횟수, T_{stop} 는 실행해야 할 남은 담금질 수를 나타낸다. 그림 2 의 7 번째 행의 M 은 주어진 해를 변형시키는 $Move()$ 의 반복횟수를 나타내며 9 번째 행의 Δ Gain 의 값은 주어진 해의 적합도를 계산하는 함수 $eval()$ 에 따라 계산된다. Δ Gain 값이 계산되면 새로 생성된 해의 수용여부는 현재온도 T 를 고려하여 $Accept()$에 의하여 결정된다. 17 번째 행의 a 는 [0, 1) 사이의 실수 값으로 온도를 낮추는 역할을 하며 담금질이 반복 될수록 온도가 내려가면서 반복적인 최적화가 이루어진다. $Accept()$는 다음 식 (1) 과 같이 정의된다.

$$Accept() = \begin{cases} true, & \Delta Gain > 0 \text{ or } R() = 1 \\ false, & otherwise \end{cases} \quad (1)$$

새로 생성된 해가 현재의 해보다 우월한 경우 새로 생성된 해는 (1) 식에 의해 항상 수용된다. 만일 새로 생성된 해가 현재의 해보다 열등한 경우에는 새로 생성된 해는 선택적으로 수용된다. 예를 들어, 본 문제와 같이 최소값을 구하는 최적화 문제에서 현재의 해를 C라 하고 새로 생성된 해를 N 이라 할 때 만일 $eval(C) < eval(N)$ 이라면 N의 수용여부는 $R()$에 의하여 결정된다. $R()$은 다음 식 (2)과 같이 정의되며 여기서 R은 [0, 1) 사이의 실수 난수이다.

$$R() = \begin{cases} 1, & R \le e^{\frac{-\Delta Gain}{T}} \\ 0, & otherwise \end{cases} \quad (2)$$

따라서, $Accept()$는 비록 열등한 해 일지라도 해의 다양성 및 지역 최적해(local optimum) 로 수렴하는 것을 방지하기 위하여 이러한 열등한 해를 수용할 수도 있게 된다. (2) 식으로부터 온도가 높을 때에는 열등한 해를 수용할 확률이 크지만 온도가 감소하면서 열등한 해를 수용할 확률은 점차 감소하게 됨을 알 수 있다.

2.2 $L(2, 1)$-labeling 문제 분석

그래프 $G = (V, E)$ ($|V| = n$)에서 정점집합 V 의 배열(ordering) 이란 전단사 함수 $\pi : \{1, 2, ..., n\} \leftrightarrow V$를 의미한다. $G(\pi)$를 G의 정점들이 특정 배열 π에 의하여 배열된 그래프(ordered graph)라 하자. G의 $L(2,1)$ -labeling을 계산하는 기본적인 알고리즘은 Griggs [3]가 제안한 First Fit Algorithm (FFA) 이다. 본 알고리즘은 주어진 그래프의 정점들을 특정배열 π에 의하여 순서를 정한 후 π의 순서에 따라 각 정점 v를 차례대로 방문하면서 {0, 1, 2, ...} 의 정수들 중 조건 (A)를 만족시키면서 v에 부여될 수 있는 가장 작은 정수를 v에 부여하는 알고리즘이며 그림 3에 표현되어 있다.

Algorithm First-fit
Input : A graph $G(\pi)$ with an ordering $\pi = (v_1, v_2, ..., v_n)$ of vertices, $S = \{x \mid x \text{ is an integer} \ge 0\}$.
Output : $L(2, 1)$-labeling of G

begin
  for $i = 1$ **to** n do
    label v_i by the lowest possible integers in S;
end

그림 3. First Fit 알고리즘
Fig. 3. First Fit Algorithm

$LG(\pi)$를 주어진 그래프 $G = (V, E)$ 및 V에 대한 배열 π에 대하여 FFA를 실행 시킨 결과로 나타나는 $L(2, 1)$-labeling 이라 하고 $L(v)$를 $LG(\pi)$에서 정점 v에 부여된 정수(label) 라 하자. 또한, $maxL(LG(\pi))$를 $LG(\pi)$에서 나타나는 가장 큰 정수(label) 라고 하자. $V = \{1, 2, 3, 4, 5, 6, 7\}$ 이라 하고 $\pi_1 = [1, 2, 3, 4, 5, 6, 7]$ 및 $\pi_2 = [4, 5, 6, 2, 1, 7, 3]$ 이라 하였을 때 다음 그림 4는 동일한 그래프에 대하여 π_1 및 π_2에 따르는 FFA 의 결과인 $LG(\pi_1)$ 및 $LG(\pi_2)$를 각각 보여준다.

그림 4의 (a) 와 (b) 로부터 $maxL(LG(\pi_1)) = 6$ 인 반면에 $maxL(LG(\pi_2)) = 5$ 임을 알 수 있으며 이로부터 FFA에 기반하는 $L(2, 1)$-labeling에 있어서 정점들의 배열 π에 따라서 $maxL(LG(\pi))$가 변할 수 있음을 알 수 있다. 또한, FFA는 Greedy Algorithm으로서 모든 배열 π에 대하여 $LG(\pi)$는 Minimal Ordering 임을 알 수 있다. 즉, $LG(\pi)$로부터 임의의 정점 v의 $L(v)$를 독립적으로 $L(v)$보다 작은 값으로 대체하면 $LG(\pi)$는 더 이상 유효한 $L(2, 1)$-labeling 이 아니다.

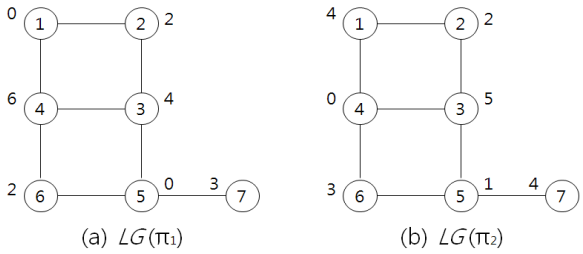

그림 4. (a) $\pi_1 = [1, 2, 3, 4, 5, 6, 7]$에 따른 FFA 결과, (b) $\pi_2 = [4, 5, 6, 2, 1, 7, 3]$에 따른 FFA 결과
Fig. 4. (a) FFA's result based on $\pi_1 = [1, 2, 3, 4, 5, 6, 7]$, (b) FFA's result based on $\pi_2 = [4, 5, 6, 2, 1, 7, 3]$

본 연구에서 제시되는 SA 알고리즘은 FFA를 위한 최적의 정점들의 배열 π를 생성시키는 것에 주안점을 두고 있다. 제안되어진 유전 알고리즘 [7] 에서는 FFA를 기반으로 정점들의 배열 π를 최적의 상태로 변화 시키며 π를 재배열시키는 교배 연산(crossover) 으로 순회방문 판매원 문제(Traveling Salesman Problem)를 위하여 개발된 Operation을 적용하였다. 따라서 제안된 유전 알고리즘 [7]에는 Current_Solution이 저장하고 있는 $L(2, 1)$-labeling에 관련된 정보가 New_Solution에 전달되지 못한다는 단점을 갖고 있으며, 본 연구에서는 이러한 단점을 완화 시키는 방법으로 Move Operation을 적용 하고자 한다.

그래프 $G = (V, E)$ 로부터 $n = |V|$ 라 하자. M을 길이 n을 갖는 벡터라 하면 $M[i..j]$ $(1 \le i \le j \le n)$를 M으로부터 $M[i] \sim M[j]$ 원소들로 구성되는 부분 벡터라 하자. 해(solution)를 표현하기 위하여 길이 n의 두 개 정수 벡터 π 및 L을 이용한다. $L[i]$는 FFA에 의하여 정점 i에 부여된 $L(2, 1)$-label을 저장하며, $\pi[i]$는 π에서 정점 i의 순서를 저장한다. 벡터 π 및 L의 관계는 π가 정해지면 π를 기반으로 FFA를 실행시킨 결과가 L에 저장된다. 따라서, $L[i]$에는 π에서 정의된 정점 순서에 따라 정점 i에 대하여 FFA가 부여한 $L(2, 1)$-label이 저장된다. 이 때 초기해(initial solution) π는 무작위로 생성된다. 그림 2의 8 행에 나타나는 $Move()$ Operation 은 다음과 같이 실행된다.

π_{cur} 및 L_{cur}를 Current_Solution 이라 하자. 먼저 $1 \le p_1 \le n$, $1 \le p_2 \le n$ 및 $p_1 < p_2$를 만족하는 두 개의 정수 형태의 난수(random integer number) p_1 및 p_2를 생성한다. T를 길이 $k = p_2 - p_1 + 1$를 갖는 벡터라 하자. 벡터 T는 다음과 같이 정의 된다.

$$T[i] = L_{cur}[\pi_{cur}[p_j]], \quad 1 \le i \le k, \; j = p_1 + i - 1.$$

즉, T에는 $\pi_{cur}[p_1..p_2]$에 포함된 각 정점 v의 $L(v)$가 저장된다. ST를 T를 오름차순으로 정렬한 결과 벡터라 하자. 마지막 단계로 $\pi_{cur}[p_1..p_2]$를 ST에 따라 재배열 시킨 후 $\pi_{new} = \pi_{cur}$로 정의 한다. 즉, $ST^{-1}[i]$를 $ST[i]$의 label을 부여받은 정점이라 하면,

$$i < p_1 \text{ 또는 } i > p_2 \text{ 라면 } \pi_{new}[i] = \pi_{cur}[i],$$

$$p_1 \le i \le p_2 \text{ 라면 } \pi_{new}[i] = ST^{-1}[i+1-p_1]$$

이다. 예를 들어

$\pi_{cur} = [4, 3, 7, 2, 5, 6, 1]$
$L_{cur} = [3, 0, 5, 9, 4, 8, 6]$ 및 $p_1 = 2$, $p_2 = 5$

라 하면

$T = [5, 6, 0, 4]$
$ST = [0, 4, 5, 6]$

이므로

$\pi_{new} = [4, 2, 5, 3, 7, 6, 1]$

이다. 이 때 L_{new}는 π_{new}를 기반으로 주어진 그래프 G에 대하여 FFA를 실행시킴으로써 계산된다. 본 Move Operation은 $\pi_{cur}[p_1 .. p_2]$에 포함된 정점들을 자신들에게 부여된 label의 크기가 작은 정점들 순서로 재배열 시키는 효과를 가져온다. 그림 5는 Move Oparation의 과정을 자세히 보여주며, 그림 2의 9행의 *eval*(solution)는 벡터 L 내 가장 큰 수로 정의 된다. 따라서 *eval*(solution) = $maxL(LG(\pi))$ 이며 Cost function은 $maxL(LG(\pi))$로 적용하였다.

```
1   Procedure Move (π_cur, L_cur)
2   begin
3     Randomly select p1 and p2 such that 1 ≤ p1, p2 ≤ n
      and p1 < p2 ;
4     k = p2 − p1 ;
5     Let T be a vector of length k ;
6     Set T[i] = L_cur[π_cur[p_j]], 1 ≤ i ≤ k, j = p1 + i − 1;
7     Sort T in nondecreasing order and let ST be the sorted
      vector ;
8     Set π_new[i] = π_cur[i], i < p1 or i > p2 ;
9     Set π_new[i] = ST^-1[i+1−p1], p1 ≤ i ≤ p2 ;
10    L_new = First-Fit (π_new) ;
11    return π_new and L_new ;
12  end
```

그림 5. Move Operation
Fig. 5. Move Operation

제안된 알고리즘의 계산 복잡도(time complexity)는 Simulated Annealing 알고리즘의 특성상 전체적인 계산 복잡도를 계산하는 것은 불가능하지만 각 모듈들의 계산 복잡도는 다음과 같이 계산 될 수 있다. 그림 3의 First-Fit()을 실행하기 위해서는 각 정점들로부터 최단 거리가 2인 정점들을 계산하여야 하며 이는 인접 행렬의 제곱 즉, $A^2(G) = A(G) \times A(G)$로부터 계산 될 수 있다. 일반적인 행렬 곱셈을 이용하면 본 계산은 $O(n^3)$을 요구하지만 Coppersmith [12]의 행렬 곱셈을 이용하면 본 계산은 $O(n^{2.376})$을 요구한다. 또한 $A^2(G)$는 정확히 한번만 계산되면 충분하다. Move Operation의 계산 복잡도는 그림 5의 7행에서 T를 오름차순으로 정렬하는 과정 및 10행에서 First-Fit()를 실행하는 과정에 의하여 결정된다. 최악의 경우 T의 길이는 n이 가능하며 이 경우 Merge Sort를 이용하면 T를 정렬하는 과정은 $O(n \log_2 n)$이 요구된다. First-Fit()에서 임의의 정점 v가 label을 부여 받는다고 하자. 최악의 경우 v의 차수는 $\Delta(G)$가 될 수 있으며 이들 $\Delta(G)$개 정점들의 차수 또한 모두 $\Delta(G)$가 가능하다. 따라서 정점 v로부터 최단거리 2이내에 위치한 정점들의 총 수는 최악의 경우 $\Delta(G) + \Delta(G)(\Delta(G)-1) = \Delta^2(G)$개가 존재 할 수 있다.

$G = (V, E)$에서 $V = \{v_1, v_2, ..., v_n\}$ 이라 하자. $1 \le i \le n$에 대하여 $L(v_i) = 2(i-1)$로 정의하면 결과는 명백히 $L(2,1)$-labeling 이며 따라서 First-Fit()이 검사하여야 할 총 label(정수)의 개수는 최악의 경우에 $2n-2$개가 된다. 따라서 First-Fit()은 최악의 경우 $\Delta^2(G)(2n-2) \in O(n\Delta^2(G))$의 계산 복잡도를 요구하며 따라서 그림 2에서 5행 및 17행 사이를 계산하는 계산 복잡도는 $O(n\Delta^2(G))$이다.

3. 실험 및 결과분석

다른 그래프 문제들과는 달리 아직 $L(2,1)$-labeling 문제를 위한 benchmark graph 들은 아직 존재하지 않고 있다. 따라서 본 연구에서는 제안된 SA 알고리즘의 효율성을 입증하기 위하여 $\lambda(G)$가 알려진 특수 그래프 (special graph)인 complete k-partite graph를 대상으로 실험 하였다.

k-partite graph는 동일한 부분 집합 내에서 간선이 존재하지 않도록 정점 집합을 k개 부분 집합으로 분할(partition) 시킨 그래프이다. 예를 들어, 이분 그래프(bipartite graph)는 k-partite graph 로서 $k=2$ 인 경우이다. Complete k-partite graph는 k-partite graph 로서 서로 다른 부분 집합의 모든 정점 쌍 사이에 간선이 존재하는 그래프로서 k개 부분 집합에서 각각 $p, q, ..., r$개 정점들이 존재한다면 complete k-partite graph는 $K_{p,q,...,r}$ 로 나타낸다. 여기서 $p, q, ..., r$을 complete k-partite graph 의 partite set이라 한다. Complete k-partite graph의 $\lambda(G)$는 Griggs [3]에 의하여 최초로 계산되었으며 다음 정리 1에 나타낸다. 또한 그림 6은 $K_{2,3,4}$를 보여준다.

정리 1. [3] $G = (V, E)$가 $|V| = n$인 complete k-partite graph 라면 $\lambda(G) = n + k - 2$이다.

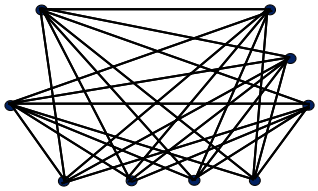

그림 6. $K_{2,3,4}$ 그래프
Fig. 6. $K_{2,3,4}$ graph

실험의 정확성을 위하여 다양한 크기를 갖는 15개 complete k-partite graph 들을 생성하였으며 이들에 대한 정보는 표 1에 나타난다. 이들은 주어진 정점 개수 n에 대하여 무작위로 생성된 partite set들을 기반으로 생성되었으며 정점 개수 n에 따라 Gn 계열의 그래프로 구분되고 있다. 또한 동일한 n 값에 대하여 서로 다른 partite set을 갖는 5개씩의 그래프들을 생성하였다. 표 1에서 $\Delta(G)$ 및 $\delta(G)$는 각각 그래프 G의 최대 차수 및 최소 차수를 나타내며 $e = |E|$를 나타낸다. 또한 마지막 열의 $\lambda(G)$는 정리 1로부터 계산된 개개 실험 그래프들의 $\lambda(G)$ 값을 보여준다.

이전 연구 [7]에서도 마찬가지로 실험 대상 그래프로서 complete k-partite graph를 사용하였지만 선정된 그래프들의 정점개수가 최대 50 개로서 크기가 다소 제한적이므로 본 연구에서는 정점개수를 최대 300 개로 확장하였다. 비교

표 1. 실험에 사용된 complete k-partite graph
Table 1. 15 complete k-partite graphs used for the experiments

Graph	Partite sets	n	e	$\Delta(G)$	$\delta(G)$	$\lambda(G)$
G100-1	15, 13, 20, 12, 17, 17, 6	100	4239	99	80	105
G100-2	10, 20, 14, 10, 12, 15, 15, 4	100	4303	99	80	106
G100-3	15, 17, 16, 18, 13, 20, 1	100	4168	99	80	105
G100-4	16, 18, 17, 11, 10, 19, 9	100	4270	99	81	105
G100-5	18, 14, 11, 16, 17, 11, 13	100	4340	99	82	105
G200-1	26, 22, 19, 21, 15, 25, 18, 20, 28, 6	200	17837	199	172	208
G200-2	26, 24, 18, 30, 27, 24, 17, 24, 10	200	17672	199	170	207
G200-3	26, 18, 18, 27, 18, 30, 28, 28, 7	200	17574	199	170	207
G200-4	26, 24, 15, 25, 22, 21, 28, 27, 12	200	17724	199	172	207
G200-5	26, 24, 24, 28, 24, 22, 23, 29	200	17885	199	172	206
G300-1	29, 35, 29, 32, 27, 33, 33, 30, 29, 23	300	40699	299	265	308
G300-2	37, 36, 35, 23, 37, 38, 21, 26, 39, 8	300	40061	299	261	308
G300-3	34, 35, 40, 33, 33, 37, 33, 21, 34	300	40454	299	260	307
G300-4	21, 33, 40, 34, 31, 23, 33, 29, 20, 36	300	40929	299	260	308
G300-5	39, 35, 28, 39, 24, 30, 27, 38, 40	300	40630	299	261	307

분석을 위하여 이미 제안된 $L(2,1)$-labeling 문제 [7]에 대한 유전 알고리즘을 GA_$L(2,1)$ 으로 나타내며 본 연구에서 제안된 SA 알고리즘은 SA_$L(2,1)$ 로 나타내기로 한다.

표 2 는 크게 3 개 열로 구분되며 각 열은 FFA, GA_$L(2,1)$ 및 SA_$L(2,1)$ 의 실험 결과를 보여준다. FFA 열의 결과는 표 1 의 그래프들에 대하여 서로 다른 정점순서를 이용하여 First-Fit 알고리즘을 100,000 회를 반복 수행한 결과를 보여준다. GA_$L(2,1)$ 열의 결과는 [7] 에서 제안한 유전 알고리즘의 결과를 그리고 SA_$L(2,1)$ 열의 결과는 본 연구에서 제안된 SA 알고리즘의 결과를 보여준다. 각 그래프에 대하여 실험은 10 회를 반복하여 이들 중 최선 (best), 최악 (worst) 및 평균 (ave) 값들을 추출하였으며 "time" 열은 10 회를 반복한 실험의 평균 실행 속도를 분 단위로 측정한 것이다. SA_$L(2,1)$ 은 다음과 같은 파라메타를 적용하였다.

$$T_0 = 1000,\ T_s = 20,\ M = 400,\ a = 0.89$$

3 개 알고리즘에 대한 프로그램은 자바 언어를 이용하여 구현 되었으며 실험 결과는 모두 Desktop PC (2.33 GHz, 3G RAM) 로부터 도출 되었다. 또한 GA_$L(2,1)$ 은 유전 연산자로서 이전 연구 [7] 에서 제시된 CX (cycle crossover) 및 SIM (simple inversion mutation) 을 적용 하였으며 또한 iteration = 2000, population size = 60, crossover rate = 0.8 및 mutation rate = 0.01 을 사용하였다. 따라서 GA_$L(2,1)$ 의 경우 약 $60 \times 2000 \times (0.8 + 0.01) = 97{,}200$ 개

표 2. FFA, GA_$L(2,1)$ 및 SA_$L(2,1)$ 실험결과
Table 2. Results of FFA, GA_$L(2,1)$ and SA_$L(2,1)$

Graph	FFA				GA_$L(2,1)$				SA_$L(2,1)$				
	best	worst	ave	time (min)	best	worst	ave	time (min)	best	worst	ave	time (min)	average Iteratraion
G100-1	180	188	184.2	2.9	143	152	148.5	3.2	110	110	110.0	1.1	44,720
G100-2	186	193	189.0	3.0	153	164	158.9	3.3	123	123	123.0	1.2	45,800
G100-3	175	187	181.8	2.9	139	148	143.7	3.0	105	105	105.0	0.9	35,708
G100-4	182	190	185.4	2.9	145	158	152.3	3.2	113	115	113.2	1.1	42,840
G100-5	184	190	187.3	2.9	145	157	152.8	3.2	117	118	117.1	1.1	44,360
G200-1	367	383	375.9	23.5	332	341	336.7	26.4	213	217	214.0	10.8	61,360
G200-2	370	380	375.9	23.4	329	341	334.3	26.6	216	219	216.8	10.9	60,360
G200-3	364	383	375.0	23.2	328	342	333.0	25.9	213	218	214.3	10.6	59,400
G200-4	364	380	374.5	23.4	325	345	334.7	26.4	218	220	218.6	10.8	60,000
G200-5	372	385	378.0	23.7	329	343	336.7	27.0	234	238	234.9	11.6	62,000
G300-1	561	573	566.8	83.0	519	538	526.6	95.4	330	336	332.8	45.7	76,440
G300-2	561	572	567.8	82.0	510	527	520.5	94.1	315	323	318.7	43.9	76,680
G300-3	563	576	566.7	83.2	514	531	523.6	96.1	340	345	342.3	48.5	79,160
G300-4	562	577	572.0	83.7	517	539	529.2	97.3	343	348	346.3	48.0	78,040
G300-5	561	578	568.4	83.5	514	533	524.7	96.0	347	354	349.3	45.4	71,240

의 새로운 해를 생성한 결과이다.

표 2 로부터 모든 그래프들에 대하여 FFA 의 결과는 나머지 2 개 알고리즘의 결과보다 현저히 낮은 효율을 보임을 알 수 있다. GA_$L(2,1)$ 의 경우 모든 15 개 그래프로부터 최적의 값을 계산하지를 못 하였지만 SA_$L(2,1)$ 의 경우 $G100$-3 그래프에 대하여 10 회 반복 실험에서 항상 최적 값인 105 를 계산하였다. 반면에 GA_$L(2,1)$ 는 $G100$-3 그래프에 대하여 10 회 반복 실험 중 최선의 값으로 139 가 도출되었으며 평균적으로는 143.7 이 도출 되었다. 또한 정점의 크기가 증가 할수록 SA_$L(2,1)$ 는 GA_$L(2,1)$ 보다 뛰어난 효율을 보여 주고 있다. 예를 들어, $G300$-1 그래프의 경우 "best" 를 기준으로 GA_$L(2,1)$ 은 519 가 도출 되었지만 SA_$L(2,1)$ 은 330 을 도출 하였다. $G300$-1 그래프의 경우 $\lambda(G) = 308$ 이므로 SA_$L(2,1)$ 은 최적값으로부터 22 의 차이를 보이지만 GA_$L(2,1)$ 은 211 의 차이를 보이고 있다.

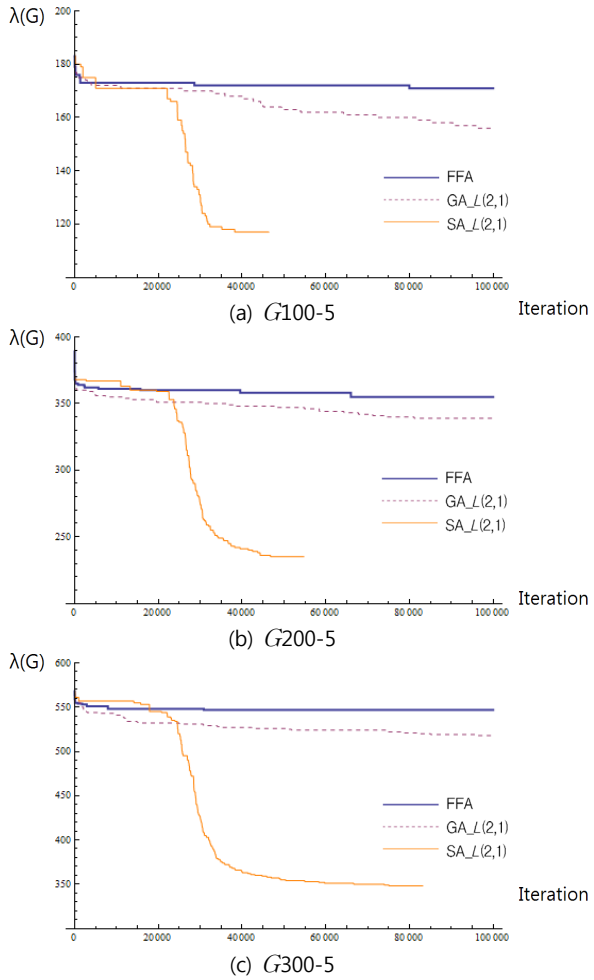

그림 7. FFA, GA_L(2, 1) 및 SA_L(2, 1)의 수렴성 비교
Fig. 7. Convergence rates of FFA, GA_L(2, 1) and SA_L(2, 1)

그림 7 은 $G100$-5, $G200$-5 및 $G300$-5 등 세 개 그래프에 대하여 세 개 알고리즘의 해의 수렴성 및 결과 값의 변화를 보여준다. 본 결과는 세 개 알고리즘에 대하여 각각 총 100,000 회의 반복을 허용한 실험 결과이지만 SA_$L(2,1)$ 의 경우 세 개 그래프에 대하여 모두 100,000 회 반복 이전에 수렴이 이루어 졌음을 알 수 있다. 예를 들어, 그림 7 의 (c) 를 보면 $G300$-5 그래프에 대하여 실행 초기에는 GA_$L(2,1)$ 이 SA_$L(2,1)$ 보다 좋은 결과를 보여주지만 반복이 진행 될수록 SA_$L(2,1)$ 은 GA_$L(2,1)$ 보다 월등히 좋은 결과를 보여준다. 이는 실행 초기에 SA_$L(2,1)$ 의 경우 높은 온도로 인하여 다양한 해를 수용하기 때문이다. 그러나 반복 횟수가 25,000 에 근접할 때부터 매우 빠른 수렴성을 보이며 반복횟수가 60,000 이 되면 거의 일정한 값에 수렴함을 알 수 있다.

표 3 은 $G300$ 계열의 그래프에 대한 GA_$L(2,1)$ 및 SA_$L(2,1)$ 의 10 회 반복된 실험 결과에 대한 표준 편차 (SD) 및 변동 계수 (CV) 를 보여주고 있다. 표로부터 GA_$L(2,1)$ 은 SA_$L(2,1)$ 에 비하여 약 3 배 정도의 편차를 보임을 알 수 있다. 또한 변동 계수 측면에서 SA_$L(2,1)$ 은 GA_$L(2,1)$ 에 비하여 상대적으로 변동성이 작다는 것을 알 수 있으며 이로부터 SA_$L(2,1)$ 은 GA_$L(2,1)$ 보다 안정적인 결과를 생성한다는 것을 확인 할 수 있다.

표 3. GA_$L(2,1)$ 및 SA_$L(2,1)$ 의 10 회 반복 실험 결과의 표준 편차 (SD) 및 변동 계수 (CV)
Table 3. Standard deviations and coefficient of variation based on 10 repeated experimental results for GA_$L(2,1)$ and SA_$L(2,1)$

Graph	GA_$L(2,1)$		SA_$L(2,1)$	
	SD	CV	SD	CV
$G300$-1	6.88	1.31	2.04	0.61
$G300$-2	6.20	1.19	2.63	0.83
$G300$-3	5.74	1.10	1.64	0.48
$G300$-4	7.51	1.42	1.57	0.45
$G300$-5	5.31	1.01	2.11	0.60

4. 결 론

본 연구의 목적은 $L(2,1)$-labeling 문제에 적용 가능한 효율적인 SA 알고리즘을 개발하는 것으로서 이를 위하여 제안된 알고리즘을 다양한 complete k-partite graph 에 적용 및 그 결과를 다른 알고리즘의 결과 및 최적 값과 비교 분석하였다. 제안된 SA 알고리즘은 이전 연구 에서 제안된 유전 알고리즘 보다 효율성이 높다는 것을 보였지만 최적값과 비교하면 여전히 차이가 존재하므로 더욱 효율적인 SA 또는 유전 알고리즘의 개발이 요구됨을 알 수 있다. 본 연구에서 제안된 알고리즘은 First-Fit 알고리즘을 반복적으로 실행시키는데 이 계산의 복잡도는 $O(n\Delta^2(G))$ 이다. 그러나 만일 $\Delta(G) \in O(n)$ 이라면 First-Fit 알고리즘의 계산 복잡도는 $O(n^3)$ 이 되므로 이에 대한 개선이 요구된다.

참 고 문 헌

[1] W. K. Hale, "Frequency assignment," *Theory and application, Proc. IEEE*, vol. 68, pp. 1497 - 1514, 1980.
[2] F. S. Roberts, "T-colorings of graphs : Recent results and open problems," *Discr. Math.* vol. 93,

pp. 229 - 245, 1991.
[3] J. R. Griggs and R. K. Yeh, "Labeling graphs with a condition at distance two," *SIAM J. Discr. Math.*, vol. 5, pp. 586-595, 1992.
[4] G. J. Chang and D. Kuo, "The L(2,1)-labeling on graphs," *SIAM J. Discr. Math.*, vol. 9, pp. 309 - 316, 1996.
[5] D. Král and R. Škrekovski, "A theorem about channel assignment problem," *SIAM J. Discr. Math.*, vol. 16, pp. 426 - 437, 2003.
[6] D. Goncalves, "On the L(p,1)-labeling of graphs," *Discr. Math. and Theor. Comp. Science AE*, pp. 81-86, 2005.
[7] 한근희, 김찬수, "유전자 알고리즘 이용한 그래프에서 L(2,1)-labeling 문제 연구," *정보처리학회논문지 B*, 제 15-B 권, 제 2 호, pp. 131-136, 2008
[8] Zbigniew Michalewicz, "Genetic Algorithms + Data Structures = Evolution Programs," *Springer-Verlag*, Berlin, 1992.
[9] West. B, "Introduction to Graph Theory," *Hall & Co.*, 2000.
[10] S. Kirkpatrick, C.D. Gelatt, M.P., Vecchi, Optimization by simulated annealing, *Science* 220, pp. 671-680, 1983.
[11] Hans L. Bodlaender, Ton Kloks, Richard B. Tan and Jan van Leeuwen, Approximations for λ -coloring of graphs, *The Computer Journal* 47, pp. 193-204, 2004.
[12] D. Coppersmith and S. Winograd, "Matirx multiplication via arithmetic progression", *Proceedings of the 19th ACM Symposium on Theory of Computing*, pp. 1-6, 1987.

저 자 소 개

한근희(Keun-Hee Han)
1986년 : 건국대학교 물리학과 졸업(학사)
1992년 : Univ. of Central Oklahoma 응용수학과 졸업(이학석사)
1996년 : Univ. of Oklahoma 컴퓨터과학과 졸업(이학박사)
1996년 ~ 2000년 : 한국전자통신연구원
1999년 ~ 2000년 : 미국 NIST 객원연구원
2000년 ~ 현재 : 공주대학교 응용수학과 교수

관심분야 : 그래프 알고리즘, Genetic Algorithm
Phone : +82-41-850-8564
Fax : +82-41-850-8560
E-mail : kehan@kongju.ac.kr

이용진(Yong-Jin Lee)
2007년 : 공주대학교 응용수학과 졸업 (학사)
2009년 : 공주대학교 응용수학과 졸업 (이학석사)
2009년 ~ 현재 : 공주대학교 응용수학과 박사과정

관심분야 : Genetic Algorithm, 진화 알고리즘
Phone : +82-41-850-0939
Fax : +82-41-850-8560
E-mail : leeyongjin@kongju.ac.kr