韓國經營科學會誌
第21卷 第2號
1996年 8月

Simulated Annealing의 가속화와 ATM 망에서의 가상경로 설정에의 적용+

윤복식* · 조계연**

Acceleration of Simulated Annealing and Its Application for Virtual Path Management in ATM Networks.

B. S. Yoon* · G. Y. Cho**

Abstract

Simulated annealing(SA) is a very promising general purpose algorithm which can be conveniently utilized for various complicated combinatorial optimization problems. But its slowness has been pointed as a major drawback. In this paper, we propose an accelerated SA and test its performance experimentally by applying it for two standard combinatorial optimization problems(TSP(Travelling Salesman Problem) and GPP(Graph Partitioning Problem)) of various sizes. It turns out that performance of the proposed method is consistently better both in convergence speed and the quality of solution than the conventional SA or SE(Stochastic Evolution). In the second part of the paper we apply the accelerated SA to solve the virtual path management problem encountered in ATM networks. The problem is modeled as a combinatorial optimization problem to optimize the utility of links and an efficient SA implementation scheme is proposed. Two application examples are given to demonstrate the validity of the proposed algorithm.

1. 서 론

이산 변수 함수의 최적점을 찾는 조합적 최적화(combinatorial optimization) 문제는 공학, 컴퓨터, OR 등 여러 분야에 걸쳐 관심의 대상이 되고 있는 문제이다. 이러한 조합적 최적화 문제를 해결하는 알고리듬은 크게 전체

+ 본 논문은 부분적으로 1995년도 홍익대학교 교내 연구비에 의해 수행되었음.
* 홍익대학교 기초과학과
** 서울대학교 제어계측신기술연구센터

최적점(global optimum)을 찾아 주는 최적화 알고리듬(optimization algorithm)과 전체 최적점에 가까운 값을 찾는 근사 알고리듬(approximation algorithm)으로 분류할 수 있다. 그러나 현실에서 대두되는 조합적 최적화 문제는 거의 대부분 NP-hard 문제이기 때문에, 최적화 알고리듬을 찾아내기가 힘들고, 최적화 알고리듬이 존재해도 문제의 크기가 커짐에 따라 전체 최적점을 찾는 것은 매우 긴 시간과 많은 량의 기억장치를 요하게 되어 실용적인 의미가 거의 없게 된다. 따라서 조합적 최적화 문제는 대부분 그 문제의 구조 및 특성에 알맞은 근사 알고리듬을 개발하여 적용하게 되는데, 다양한 형태의 조합적 최적화 문제에 포괄적으로 적용될 수 있는 범용의 근사 알고리듬이 존재하면 새로운 조합적 최적화 문제가 나타날 때마다 새로운 알고리듬을 개발하여야 하는 부담을 덜 수 있어 매우 유용할 것이다.

Kirkpatrick et al. (1983)등에 의해 제안된 SA(Simulated Annealing)은 바로 이러한 형태의 근사 알고리듬으로서 기존의 반복적인 개선(iterative improvement)에 근거한 발견적 기법(heuristic methods)들이 국부 최소점(local minimum)에 빠져 버리는 단점을 개선한 범용의 최적화 기법으로 현재까지 CAD를 비롯한 많은 분야에서 응용되고 있다(van Laarhoven and Aarts(1987)). 기본 개념의 단순성과 범용성이 두드러지는 SA는 전체 최소점에로의 수렴성이 이론적으로 증명([3], [12])되었으며, 많은 실험 결과들([1], [4], [5])을 통해 타당성이 확인되어 해당 문제에 대해 개발된 특수한 최적화 알고리듬이 없을 때 편리하게 사용할 수 있는 알고리듬으로서 알려져 있다.

그러나 SA는 적용 상에서 아직까지 미해결된 문제점들을 가지고 있고, 특히 계산 시간이 길다는 단점을 가지고 있는데[Huang et al. (1986), Saab abd Rao(1991)] 따라서 해의 질을 떨어뜨리지 않고 계산 시간을 줄일 수 있는 SA의 변형을 고려할 필요가 있다[윤복식 외 (1995) 참조]. 본 논문에서는 윤복식 외(1995)에서 제시된 방법을 보완하여 SA의 보다 효과적인 변형 방법을 찾아보고 변형된 알고리듬을 대표적인 조합적 최적화 문제들에 적용시켜 충분한 실험을 거쳐 효율성을 확인한 후 ATM망에서의 가상경로 설정문제에 효과적으로 적용시키는 방법을 제시하는 것을 주목표로 한다.

ATM망에서 셀은 셀 헤더의 가상경로 식별자(VPI : Virtual Path Identifier) 및 가상채널 식별자(VCI : Virtual Channel Identifier)에 따라 목적지까지 전송된다. 가상경로는 전송매체 계층과 망계층사이에 논리적인 경로망을 제공함으로써 트래픽의 변동과 같은 망의 변화나 설비고장과 같은 전송매체 계층의 변화에 신축적으로 대응할 수 있게 하여, 망자원을 경제적으로 할당하고 망계층의 운영을 효율적으로 할 수 있도록 한다[Suzuki and Tobagi (1992), Chlamtac et al.(1993), Ahn et al. (1994), Arvidsson(1994), 주종혁(1994) 등 참조]. 그러나 이러한 효과는 가상경로를 효율적으로 설정할 수 있을 때에 한하여 나타날 수 있고, 또한 망의 트래픽 변화에 따라 동적으로 가상경로의 용량을 조절해 주는 것이 바람직하다. 가상경로는 가입자의 개별적 연결이 아니라 전체적으로 망에 부가되는 트래픽의 양에 의해 결정되어야 하며 이것은 망 전체의 관점에서 최적의 가상경로 토폴로지와 용량 설정이 이루어져야 한다는 것을 의미한다.

본 논문에서는 각 링크의, 미리 설정된 가상경로 트래픽 이외의 여분의 트래픽에 대한 우회경로로의 용도를 고려하여 망의 관점에서 링

크의 이용도를 최대화하는 가상경로 설정 문제의 모형을 제시한다. 이 모형은 목적함수가 비선형 함수인 비선형 정수형(nonlinear integer programming) 조합적 최적화 문제가 되어 최적화 알고리듬을 얻기가 거의 불가능한데 본 연구에서는 가속화된 SA를 사용한 해법을 제시한다.

2. SA 알고리듬의 가속화

일반적인 SA 알고리듬은 아래와 같은 형태이다.

```
ALGORITHM SA
begin ;
INITIALIZE(X, T, L) ;
repeat
for i=1 to L do
   Y=PERTURB(X) ;
   if (C(Y) ≤ C(X)) or (exp ((C(X) - C(Y))/ T) > random (0, 1))
      then X=Y ; {accept the movement}
   endif ;
endfor ;
UPDATE (T, L) ;
until (Stop-criterion)
end
```

[그림 1] SA 알고리듬

현재의 해로부터 적절한 방법으로 근방에 새로운 해를 만들어서(PERTURB) 새로운 해의 목적함수의 값이 현재 해의 값보다 작으면 새로운 해를 최적해 후보로 받아들이고, 그렇지 않을 경우에도 무조건 기각하지는 않고 적절한 확률로 새로운 해를 받아들여서 국부최소점에서 빠져 나올 수 있도록 하여 전체 최소점을 찾아가게 하는 것이 SA의 요점이다.

보통 목적함수값을 증가시키는 상승이동(up-hill movement)을 받아들이는 확률은 $\exp(\Delta C/T)$ (Metropolis criterion)로 주어지는데 여기서 ΔC는 목적함수 값의 차이이고, T는 상승이동을 받아들이는 확률을 통제하는 물리적 어닐링(annealing)과정에서의 온도에 해당하는 컨트롤 파라미터이다. T는 초기에 높게 설정되었다가 알고리듬이 진행되어 감에 따라 점점 줄어들게 된다. 이것은 알고리듬이 진행되어 감에 따라 상승이동을 할 확률이 점점 줄어듦을 의미한다. 따라서 SA는 처음에는 랜덤 워크(random walk)의 양태를 보이다가 시간이 흐

름에 따라 T가 작아지면 국부 탐색법(local search)의 양태를 보이는 알고리듬으로 볼 수 있다.

SA에 관한 많은 연구에서 관측되고 있는 거의 일치된 결론은 충분한 시간이 주어진다면 SA가 최적해에 만족할 정도로 가까이 간다는 사실인데 수렴시간이 상당히 길다는 것이 단점으로 지적되고 있고 현재까지 수렴시간을 단축시키려는 노력이 계속되고 있다. 윤복식 외(1995)에서는 SA의 수렴속도를 개선시키기 위한 여러 가지 방안이 제시되고 소개된 바 있는데 본 연구에서 주로 초기 컨트롤 파라미터의 설정 방법, 냉각 스케줄, 종료조건의 개선 및 변형을 통해 다음과 같은 목표를 가지고 SA의 개선을 시도 한다.

(1) 수행시간 : 최적에 가까운 해에 빨리 도달하게 한다.

(2) 초기설정 파라미터 : 사용자가 미리 설정해 주어야 하는 파라미터의 수를 되도록 줄인다.

(3) 수렴성 : 되도록 SA의 기본틀을 유지하여 이론적인 수렴성을 손상시키지 않는다.

2. 1. 초기 컨트롤 파라미터

초기 컨트롤 파라미터는 높은 상승 이동도 받아들일 수 있도록 높게 설정되어야 하는데, 일반적으로 초기 실험을 통해서 결정하게 된다. 여기에서는 해를 받아들이는 비율(acceptance ratio)을 이용하여 초기 컨트롤 파라미터를 결정하도록 한다. 여기서 특정 T에서의 해를 받아들이는 비율을 $R(T)$라고 하면, 정해진 컨트롤 파라미터 T에서 일정 횟수 동안 반복을 한 후

$$R(T)=\frac{number\ of\ acceptance}{number\ of\ perturbation} \qquad (1)$$

와 같이 구할 수 있다.

초기 컨트롤 파라미터를 결정하기 위해서 먼저 목표로 삼은 $R(T)$를 $R(T)_0$라고 하자. 여기서 $R(T)_0$는 거의 모든 해를 받아들일 수 있도록 1에 가까운 값으로 잡는다. 처음에는 T를 아주 작은 값으로 설정한 다음 그 때의 T에서의 $R(T)$를 계산하여 $R(T)_0$ 이상이 되면 그 T를 초기 컨트롤 파라미터로 하고, 그렇지 않으면 $R(T)$와 $R(T)_0$의 차이에 따라 컨트롤 파라미터를 올리는 폭을 다르게 하여 그 다음 T를 결정하여 위의 과정을 반복한다. 즉 $R(T)_0$를 만족하지 못한 T에 대해서 그 다음 T_{new}를 다음 식(2)와 같이 결정한다.

$$T_{new}=T(1+(R_0-R(T))\ *\text{상수} \qquad (2)$$

이렇게 해를 받아들이는 비율에 따라 T를 올리는 폭을 달리함으로써 원하는 $R(T)_0$에 빠르게 도달할 수 있고, 불필요하게 높은 T를 초기 컨트롤 파라미터로 설정하는 것을 피할 수 있다.

2. 2. 내부 루프와 냉각 스케줄

SA에서 내부 for 루프의 반복 횟수와 냉각 스케줄은 서로 깊은 관계를 갖는다. 내부 루프에서의 해의 변동 과정을 마코프 체인으로 모형화할 수 있는데, 보통 수렴성을 유지하기 위해 정해진 T에서 마코프 체인이 안정상태에 도달하도록 충분한 횟수만큼 내부 루프의 반복한 후에 T를 내려 주어야 한다. 현재 일반적으로 사용하고 있는 내부 루프의 횟수는 문제의 크기 (예를 들면 근방(neighbourhood)의 크기)에 일정 상수를 곱하는 형태로 설정하여 미리 충분히 크게 잡아 주거나 또는 매번 그 컨

트를 파라미터에서 얻어지는 목적함수 값들의 적절한 통계량을 이용하여 내부 루프의 횟수를 갱신하는 방식을 따르고 있다.

본 논문에서는 불필요하게 길어지는 내부 루프를 줄여 알고리듬의 속도를 향상시키기 위해 내부 루프의 횟수를 필요할 때만 늘려 주는 방법을 제안한다. 즉, 매회 내부 루프를 끝낸 후 만약 그 직전의 T에서의 목적함수와 비교하여 비용 감소가 없으면 T를 그대로 두고 있으면 T를 내린다. 수렴의 촉진을 위해서 냉각 스케줄은 $a*T(a=0.9$ or $0.95)$ 형태의 기하학적인 스케줄을 사용한다. 이렇게 하는 이유는 앞에서도 언급하였듯이 이론적으로 고정된 T에 대해 내부 루프의 전이를 L번 일으킨 마코프 체인이 안정상태에 도달해야 하는데, 안정상태에서는 목적함수의 평균값이 T값이 낮아짐에 따라 낮아져야 한다. 따라서 내부 루프를 돌고 난 후에도 목적함수 값이 떨어지지 않으면 아직도 그 T에서 안정상태에 도달하지 못했다고 간주할 수 있으므로 T를 고정시킨 채 내부루프를 계속 돌린다. 물론 최악의 경우 고정된 T에서의 내부 루프가 무한번 돌 수 있는 가능성이 있기 때문에 고정된 T에서의 내부 루프의 수에 상한을 잡아 주어야 한다. 이렇게 함으로써 L을 최소로 줄일 수 있어서 내부루프를 필요 이상 많이 돌리는데 따른 시간의 낭비를 방지할 수 있고 개별 문제에 따라 적절한 L을 설정해야 하는 번거러움을 예방할 수 있다. 실제로 이러한 방법을 통해서 내부루프의 수를 가능한 한 줄일 수 있을 뿐 아니라 마코프 체인의 평형성을 더욱 강화하였기 때문에 많은 계산상의 이득을 볼 수 있었다(3장 참조).

또한, 현재까지의 최소의 목적함수 값을 준 해를 계속 기억해 나가면서 그 값을 개선시키는 컨트롤 파라미터에서는 내부 루프를 한 번 더 돌리는 방식도 수용하였는데 왜냐하면 좋은 해의 주변에는 좋은 해가 모여 있을 가능성이 크기 때문에 똑 같은 조건에서 내부 루프를 더 돌리면 더 좋은 해를 찾을 가능성을 높일 수 있기 때문이다.

2. 3. 종료 조건

종료 조건은 외부 루프를 M번 반복하는 동안 비용의 변화가 없거나 또는 내부 루프 안에서 최소점을 갱신하는 사이의 간격을 이용하여 최소점이 발견되면 Counter2=0으로 놓고 아니면 Counter2를 증가시켜 Counter2가 N(내부루프수*상수) 보다 커지면 종료하는 방식을 이용한다. 이렇게 하는 이유는 일반적으로 목적함수의 변화 여부를 검사하여 외부루프를 M번 반복하는 동안 비용의 변화가 없을 경우만을 종료조건으로 사용하면, 이 조건을 만족하기 위해서 해의 개선 없이 낮은 컨트롤 파라미터에서 헛되이 보내는 시간이 너무 길다는 것을 실험을 통해서 확인할 수 있었기 때문이다. 또한 최소점이 갱신되는 점들의 사이 간격들을 살펴보면 낮은 컨트롤 파라미터에서 어느 순간부터 점점 길어짐을 알 수 있는데, 이때 낮은 컨트롤 파라미터는 acceptance ratio를 가지고 판단할 수 있다.

따라서, 본 연구에서는 목적함수의 변화 여부를 이용한 종료조건을 사용하면서 이 경우 낮은 컨트롤 파라미터에서 허비되는 시간을 단축시키기 위해 최소점이 갱신되는 사이 간격을 이용하여 두 조건 중 어느 하나가 먼저 만족되면 알고리듬이 종료하는 방식을 따른다.

2. 4. 알고리듬 SA1

이상을 고려하면 [그림 2]와 같은 변형된 알고리듬 SA1을 얻을 수 있다. SA1은 SA의 변형 알고리듬 중 하나이며 효율성이 높다고 보고된 SE(Saab and Rao(1991))[그림 3 참조]와 달리 기본적으로 SA의 기본 구조를 변화시키지 않아 SA의 장점을 그대로 유지할 수 있고 오히려 내부루프에서의 마코프 체인의 평형성을 더욱 강화하였기 때문에 다음 장의 실험에서 보듯이 수렴속도가 매우 빨라짐을 알 수 있다.

```
Algorithm SA1
   INITIALIZE(X, T, L);
   X_best=X;
   Counter1=0;
   Counter2=0;
   repeat
      Costold=C(X);
      Check=0;
      for i=1 to L do
         Y=PERTURB(X);
         if (C(Y) ≤ C(X)) or
              (exp ((C(X) - C(Y))/ T) > random (0, 1))
         then
         X=Y;{accept the movement}
      endif;
      if (C(X) < C(X_best)) then
         X_best=X;
         Counter2=0;
         Check=1;
        else
         Counter2=Counter2+1;
        endif;
      endfor;
      Costnew=C(X);
      UPDATE (T, Costnew, Costold, Check);
      if(Costnew=Costold) then
         Counter1=Counter1+1;
      else
         Counter1=0;
      endif;
   until (Counter1 > M or Counter2 > N);

UPDATE(T, Costnew, Costold, Check)
   if(Check=1 or Costnew < Costold) then
      T=aT;
   endif;
```

[그림 2] 알고리듬 SA1

```
Algorithm SE
INITIALIZE(X,T)
X_best=X;
Counter=0;
REPEAT
    Costold=C(X);
    Y=PERTURB(X,T);
    Costnew=C(Y);
    UPDATE(T,Costold,Costnew)
    IF (C(Y) < C(X-best)) THEN
        X_best=Y;
        Counter=Counter - R;
    ELSE
    Counter=Counter+1;
    ENDIF;
UNTIL (Counter > R);

PERTURB(X, T)
  FOR EACH (m M) DO
    Y=MOVE(X, m);
    GAIN(m)=C(X) - C(Y);
    IF(GAIN(m) > RANINT(-T, 0)) THEN
      X=Y;
    ENDIF;
  ENDFOR;
  X=MAKE-STATE(X);
  RETURN(X)

UPDATE(T, Costold, Costnew)
  IF(Costold=Costnew) THEN T=f (T);
  ELSE  T=To;
  ENDIF
```

[그림 3] SE 알고리듬

3. 효율성 검증

본 장에서는 2장에서 제시된 알고리듬 SA1을 대표적인 조합 최적화 문제인 TSP(Travelling Salesman Problem), GPP(Graph Partitioning Problem)에 적용한 결과를 제시한다. 그리고 그 결과를 Kirkpatrick et al. (1982)이 제시한 기존의 SA 알고리듬(SA0로 표기)과 SA의 변형의 대표적인 알고리듬인 SE 알고리듬의 결과와 해의 질과 수행 시간의 두 가지

측면에서 비교를 행한다. 통계적 유의성을 보장하기 위하여 모든 실험 결과는 같은 문제에 대해서 20번씩 풀었으며 해의 질과 수행시간 각각에 대해서 t-test를 행한 결과도 아울러 제시하였다. 그리고 모든 문제는 랜덤하게 발생시켰으며 사용한 컴퓨터 기종은 IBM 486 PC이다.

3. 1. TSP에 대한 실험

TSP문제는 n개의 도시가 주어지고 i, j 도시 사이의 거리가 주어졌을 때 출발 도시에서 가장 최소의 거리로 모든 도시를 한번씩 거쳐서 다시 출발 도시로 돌아오는 경로를 찾는 문제이다. 따라서 하나 하나의 해는 모든 도시를 한번씩 거치는 임의의 순서이고, 비용은 거리의 합이 된다. 그리고 해의 이동(perturbation)은 임의의 두 도시를 선택하여 그 사이의 순서를 반대로 하는 방법을 사용한다. 다음 [그림 4]에서 i, j가 선택되었을 때의 예를 볼 수 있다.

[그림 4] TSP의 해의 이동

예제는 2차원 평면상에 주어진 개수의 도시의 x, y좌표를 랜덤하게 발생시켰고 하나의 예제에 대해서 동일한 초기해를 가지고 세 가지의 알고리듬을 수행하였다. 아래 〈표 1〉에 도시의 개수에 따른 SA1, SA, SE를 20번 수행한 결과의 평균을 나타내었다.

〈표 1〉 TSP 실험 결과(평균)

Node		SA0	SA1	SE
100	Cost	832.46	844.92	889.13
	Time(sec)	71.38	38.47	42.91
150	Cost	1027.53	1021.74	1089.54
	Time(sec)	146.39	79.43	91.73
200	Cost	1174.43	1174.25	1242.84
	Time(sec)	234.27	125.48	175.81
250	Cost	1282.82	1283.62	1354.36
	Time(sec)	321.19	178.23	334.56

앞의 표에서 알 수 있듯이 SA1은 SA와 목적함수의 값에 있어서는 별다른 차이를 보이지 않지만 수행시간에 있어서 상당한 속도의 개선이 있음을 알 수 있다. 그리고 SE와 비교해 보면 목적함수 값과 수행시간 모든 면에서 SA1이 우월하다는 것을 알 수 있다.

그리고 아래 〈표 2〉에 20번 수행한 결과 중 최소의 목적함수의 값과 그 때의 수행시간을 나타내었다. 역시 앞의 표에서와 동일한 결론을 얻을 수 있다.

〈표 2〉 TSP 실험 결과(최소)

Node		SA0	SA1	SE
100	Cost	814.68	823.34	877.12
	Time(sec)	72.94	39.49	56.41
150	Cost	1011.54	1003.33	1064.51
	Time(sec)	142.75	80.68	98.81
200	Cost	1164.41	1163.46	1214.53
	Time(sec)	244.75	116.94	203.78
250	Cost	1263.47	1267.91	1327.83
	Time(sec)	331.81	187.52	366.24

위의 결론에 통계적 유의성을 보장하기 위하여 SA0와 SA1, SA1과 SE에 대해서 수행시간, 목적함수의 값에 대해서 t-test를 수행하였다. 유의수준은 5%로 하였고 그 결과는 아래 〈표 3〉, 〈표 4〉에 나타내었다.

〈표 3〉 SA0와 SA1의 T-Test (TSP)

	Cost	Time
100	유의한 차이가 없음	유의한 차이가 있음
150	유의한 차이가 없음	유의한 차이가 있음
200	유의한 차이가 없음	유의한 차이가 있음
250	유의한 차이가 없음	유의한 차이가 있음

〈표 4〉 SE와 SA1의 T-Test (TSP)

	Cost	Time
100	유의한 차이가 있음	유의한 차이가 없음
150	유의한 차이가 있음	유의한 차이가 없음
200	유의한 차이가 있음	유의한 차이가 있음
250	유의한 차이가 있음	유의한 차이가 있음

위의 표에서 알 수 있듯이 변형된 알고리듬 SA1은 기존의 알고리듬 SA0 알고리듬과 비교하였을 때 목적함수의 값은 유의한 차이가 없고, 수행시간에는 유의한 차이가 있음을 알 수 있다. 즉 해의 질은 떨어지지 않고 수행시간이 줄었음을 알 수 있다. 그리고 SE와 비교하였을 경우에는 목적함수의 값에 있어서는 모두 SA1이 통계적으로도 우월하였으며, 수행시간의 경우 도시의 갯수가 많은 경우에 통계적으로 유의한 차이가 있음을 알 수 있다.

3. 2. GPP에 대한 실험

GPP(Graph Partitioning Problem)은 그래프 G(V, E)가 주어졌을 때 vertex V를 똑 같은 개수로 V1과 V2로 나누었을 때, 두 집합 V1과 V2를 연결하는 edge의 수를 최소로 하는 V1과 V2의 집합을 찾는 것이다. GPP를 풀 때의 해, 해의 이동, 비용에 대한 정의는 Johnson et al. (1989)의 논문을 따른다.

먼저 해에 대한 정의는 $V=V1 \cup V2$가 되는 임의의 V의 분할 (partition)이 되고, 해의 이동은 임의의 vertex하나를 그 vertex가 속한 집합에서 다른 집합으로 옮기는 것이다. 그리고 하나의 분할 (V1, V2)에 대한 비용은 아래 식 (3)과 같이 정의한다.

$$c(V1,V2)=|\{\{u, v\}\in E : u\in V1 \& v\in V2\}|+ \alpha(|V1|-|V2|)^2 \quad (3)$$

여기서, $\alpha(|V1|-|V2|)^2$: 벌금 항목(penality term)

이렇게 해, 해의 이동, 비용을 정의하는 것은 비가능 해를 허용함으로써 해의 이동을 쉽게 할 수 있고, 국부 최소점에서 빠져나갈 수 있는 다양한 길을 줄 수 있으며, GPP의 경우 하나의 해의 근방의 크기(neighbourhood size)가 작아져 수행시간을 줄일 수 있는 이점이 있다. 대신 비용에 벌금 항목을 제곱의 형태로 첨가하여 낮은 컨트롤 파라미터에서는 V1과 V2의 집합이 거의 균형을 이룰 수 있도록 한다. 그리고 최종해가 비가능 해일 경우에는 greedy 법을 사용하여 최종 가능해를 구한다.

실험의 대상이 되는 그래프는 우선 100*100의 2차원 평면상에 랜덤하게 원하는 개수만큼 vertex를 발생시킨 다음 vertex사이의 거리에

반비례하는 확률로 두 vertex사이를 연결하는 edge를 발생하였다.

TSP와 마찬가지로 모든 문제에 대해서 20번씩 알고리듬을 수행하였으며, 초기해는 동일한 해를 사용하였다. 아래 〈표 5〉에 SA0, SA1, SE알고리듬의 결과를 평균을 내어서 나타내었다.

〈표 5〉 GPP에 대한 실험 결과(평균)

Node		SA0	SA1	SE
100	Cost	46.0	46.4	46.3
	Time(sec)	85.64	14.93	68.20
200	Cost	211.5	214.9	246.0
	Time(sec)	393.25	110.65	368.54
300	Cost	443.2	450.8	687.8
	Time(sec)	1752.99	415.51	1075.88
400	Cost	432.0	431.5	660.6
	Time(sec)	3687.96	1127.15	1528.41

위의 표에서 알 수 있듯이 SA1은 SA0와 비교하면 목적함수의 값에 있어서는 별다른 차이를 보이지 않지만 수행시간에 있어서는 상당한 속도의 개선이 있음을 볼 수 있다. 그리고 SE와 비교해 보면 SA1이 수행시간과 목적함수의 값에 있어서 모두 우월함을 알 수 있다.

다음 〈표 6〉은 20번의 수행 결과 중 최소의 목적함수의 값과 그때의 수행시간을 정리한 표이다. 표에서 알 수 있듯이 SA1이 거의 모든 경우에 가장 빠른 시간에 가장 작은 최소값을 찾음을 볼 수 있다.

〈표 6〉 GPP의 실험 결과(최소값)

Node		SA0	SA1	SE
100	Cost	46	46	46
	Time(sec)	51.08	12.35	60.75
200	Cost	205	204	246
	Time(sec)	319.44	111.61	305.99
300	Cost	433	433	450
	Time(sec)	1529.62	353.77	2052.56
400	Cost	432	427	653
	Time(sec)	3529.51	1029.47	1592.29

TSP와 마찬가지로 통계적 유의성을 보장하기 위하여 유의 수준을 5%로 하고 t-test를 수행한 결과를 아래 〈표 7〉, 〈표 8〉에 정리하였다.

〈표 7〉 SA0와 SA1의 t-Test (GPP)

	Cost	Time
100	유의한 차이가 없음	유의한 차이가 있음
200	유의한 차이가 없음	유의한 차이가 있음
300	유의한 차이가 없음	유의한 차이가 있음
400	유의한 차이가 없음	유의한 차이가 있음

〈표 8〉 SE와 SA1의 t-Test (GPP)

	Cost	Time
100	유의한 차이가 없음	유의한 차이가 있음
200	유의한 차이가 없음	유의한 차이가 있음
300	유의한 차이가 있음	유의한 차이가 있음
400	유의한 차이가 있음	유의한 차이가 없음

앞의 T-test 결과표를 보면 SA1은 SA0에 비해 목적함수의 값은 차이가 나지 않지만 수행시간은 유의한 차이가 있음을 알 수 있다. 그리고 SE와는 vertex의 수가 작은 문제에 있어서는 수행시간에 있어 유의한 차이가 있었고, vertex의 수가 큰 문제에 있어서는 목적함수와 수행시간에 있어 유의한 차이가 있었다.

이상의 TSP와 GPP의 실험 결과를 종합해 보면, SA1은 SA0와 비교하였을 때 목적함수의 값에 있어서는 유의한 차이가 나지 않고, 수행시간은 상당히 짧아졌다. 그리고 SE보다는 거의 모든 경우에 있어서 수행시간과 목적함수의 값에 있어서 성능이 우수하였다.

4. ATM망에서의 가상경로 설정문제에의 적용

4. 1. 문제의 모형화

ATM 망에서의 가상경로 설정의 문제는 물리적 전송망, 링크의 용량, 트래픽 수요 등이 주어졌을 때 요구하는 트래픽 쌍간에 최상의 라우팅과 용량을 결정하는 문제이다. 이것을 조합 최적화 문제로 모형화 하기 위하여 용량을 U를 단위로 하여 이산화 한다(Lee and Yee(1989) 참조). 편의상, U의 단위를 bit/second라고 하자. 예를 들면, 본 모형에서 링크 e의 용량이 C_e(정수)라면 실제 용량은 U * C_e 가된다. 또한 노드 (o, d)쌍간의 가상 경로는 o에서 d로의 단순 경로(simple path)라고 부른다. 본 모형에서는 트래픽의 요구가 비대칭적일 수 있고, 하나의 노드 쌍간에 여러 개의 가상경로를 허용한다.

문제의 모형화를 위해서 변수를 정의하면 다음과 같다.

■ 변수 정의

p_i : 트래픽 요구가 있는 i 노드쌍

C_e : 링크 e의 용량

r_i : 노드쌍 p_i의 단순 경로 집합

tp_i : 노드쌍 p_i의 트래픽 요구량

E_i : 링크 i를 지나는 모든 단순 경로의 집합

f_i : 링크 i를 지나는 단순 경로에 할당된 용량의 합

L : 모든 링크의 집합

P : 트래픽 요구가 있는 모든 노드쌍의 집합

R : 모든 단순 경로의 집합

X^j_{pi} : i노드쌍의 j경로에 할당된 용량

가상경로는 미리 노드쌍간의 트래픽 수요를 예측한 후 설정하게 되는데, 가상경로를 설정한 후에 가상경로가 설정되지 않은 노드쌍간의 트래픽이나 할당된 가상경로의 용량이 넘어서는 트래픽에 대해서는 링크의 남은 용량을 고려해서 교환기를 통해 적절한 경로로 트래픽을 전송하여야 한다. 따라서 망 전체의 관점에서는 가상경로를 요구하는 모든 노드쌍간에 원하는 양만큼의 가상경로를 설정하되 링크에서의 가상경로에 할당된 용량을 제외한 여분의 용량이 많아야 한다. 따라서 본 논문에서는 링크의 가상경로에 할당된 양과 여분의 양의 비의 전체 링크에서의 합을 식 (4)와 같이 목적함수로 설정하는데 이것은 결국 전체 망을 Jackson network로 가정하고 각 노드를 M/M/1으로 모형화 했을 때의 평균 지연시간과 유사한 값으로 파악될 수 있다.

$$L_r=\sum_{e\in L}\frac{f_e}{C_e-f_e} \tag{4}$$

이때 하나의 링크에 할당된 용량의 합은

$$f_k=\sum_{P_i\in P}\sum_{j\in E_k}X_{p_i}^j \quad k\in L \tag{5}$$

$$f_k\le C_k, \qquad k\in L \tag{6}$$

를 만족해야 하고 트래픽 요구가 있는 모든 노드쌍 p_i에 대해 r_i에 할당된 용량은 트래픽 요구량과 같아야 하므로

$$\sum_{j\in r_i}X_{p_i}^j=tp_i, \qquad \forall p_i\in P \tag{7}$$

의 조건이 얻어진다.

이상을 정리하면 아래와 같은 비선형 정수계획법 문제가 얻어진다.

$$\text{Minimize} \quad \sum_{e\in L}\frac{f_e}{C_e-f_e} \tag{8}$$

$$\text{s.t.} \quad \sum_{j\in r_i}X_{p_i}^j=tp_i, \quad \forall p_i\in P \tag{9}$$

$$f_k=\sum_{P_i\in P}\sum_{j\in E_k}X_{p_i}^j \quad \forall k\in L$$

$$f_k\le C_k, \qquad k\in L \tag{11}$$

$$X_{p_i}^j \text{ is nonnegative integer, } \forall p_i\in P, \quad \forall j\in r, \ \forall r_i\in R \tag{12}$$

$$f_i \text{ is nonnegative integer, } i\in L \tag{13}$$

4. 2. 개선된 SA1을 이용한 해법

Simulated Annealing으로 문제를 풀기 위해서는 해의 집합(solution set), 근방 구조(neighbourhood structure), 비용함수를 결정하여야 하는데 우선 대한 비용함수는 앞의 식(8)이 된다. 가상경로 문제는 앞에서도 언급하였듯이 라우팅과 용량의 두 가지를 동시에 결정하는 문제이다. 위와 같이 모형화를 할 경우 하나의 변수($X_{p_i}^j$)만으로 두 문제를 동시에 결정할 수 있다. 예를 들어 $X_{p_i}^3=5$라고 하는 것은 노드쌍 p_i에 대해서 3번 경로로 5만큼의 용량을 할당함을 의미한다. 따라서 해는 제약식을 만족하는 $X_{p_i}^j$값들이 된다.

근방 구조(neighbourhood structure)는 하나의 해에서 그 다음 해로의 이동을 결정하게 되는데 해집합의 원소들 사이를 원활하게 움직일 수 있으면서 되도록 단순한 이동이 되도록 만들어 주어야 한다. 본 연구에서는 근방 구조를 결정하기 위하여 다음의 변수를 정의한다.

RM_e : 링크 e의 남아 있는 용량
$=C_e-f_e$

현재의 해에서 다음 해로의 해의 이동은 임의로 하나의 $X_{p_i}^j$를 선택하여 거기에 할당된 값의 일부를 같은 노드쌍의 다른 경로에 할당하는 것으로 정의한다. 이 과정은 아래와 같이 이루어진다.

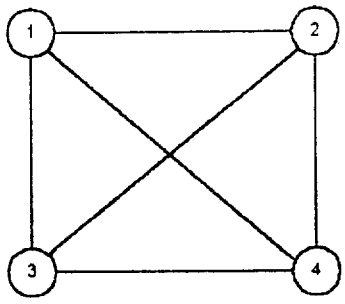
Step 1. 먼저 모든 링크에서는 자기의 RM_e를 유지한다. 노드쌍 p_i를 랜덤하게 선택한 다음 그 노드쌍에 대해 0보다 큰 $X_{p_i}^j$를 랜덤하게 선택하고 같은 노드쌍 p_i에서 임의의 다른 경로($X_{p_i}^k$)를 선택한다(이것은 $X_{p_i}^j$에 할당된 용량의 일부를 $X_{p_i}^k$로 옮기려고 할 때 그 대상을 선택하는 과정이다).

Step 2. 해의 이동의 대상이 되는 $X_{p_i}^j$가 선택

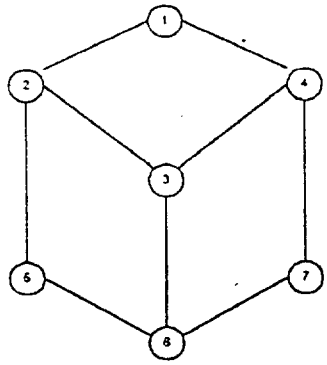
되면 옮길 수 있는 최대의 용량을 결정한다. 보다 구체적으로 먼저 선택된 $X^k_{p_i}$의 경로 k가 거쳐가는 모든 링크의 RM_e중에서 최소값을 찾는다. 그 최소값을 MR이라 하면 결국, MR은 $X^k_{p_i}$에 할당할 수 있는 최대의 용량이 된다. 따라서 $X^k_{p_i}$에 Uniform[1, min($X^j_{p_i}$, MR)]의 양을 더해주고 그 양만큼 $X^j_{p_i}$에서 빼준다. 그 다음 마지막으로 노드쌍 p_i의 j와 k경로가 지나가는 링크의 RM_e를 보정한다.

4. 3. 예제

4. 2절의 구현 방법을 사용하여 SA1의 알고리듬을 적용하는 실험은 아래의 두 가지 망에 대해서 수행하였다. SA1 알고리듬을 수행하기 위해서 먼저 트래픽의 요구가 있는 노드쌍간의 모든 경로를 찾는 알고리듬을 이용하여 후보가상경로를 찾아 입력하였고, 각 노드쌍간의 트래픽의 요구량은 양방향간에 비대칭적이 되도록 준비하였고 초기해는 임의로 주었다. 실험은 동일 문제에 대해서 10번씩 486PC에서 수행하여 수행시간과 계산 결과를 측정하였다. 실험의 대상이 되는 망은 다음 [그림 5-a], [그림 5-b]와 같다.



[그림 5] (a) A망(노드 : 4, 링크 :6)



(b) B망 (노드 : 7, 링크 : 9)

A망의 입력 자료로 각 링크의 용량과 노드쌍간의 트래픽 요구량은 아래 〈표 9〉, 〈표 10〉과 같이 준비하였다.

〈표 9〉 A망의 링크의 용량

링크	용량	링크	용량
1-2	200	2-3	300
1-3	250	2-4	100
1-4	150	3-4	200

〈표 10〉 A망의 트래픽 요구 행렬

from \ to	1	2	3	4
1	0	50	70	100
2	25	0	30	50
3	10	40	0	30
4	20	60	30	0

그리고, B망의 입력 자료로 각 링크의 용량과 노드쌍간의 트래픽 요구량은 아래 〈표 11〉, 〈표 12〉 와 같이 준비하였다.

〈표 11〉 B망의 링크의 용량

링크	용량	링크	용량
1-2	200	3-6	200
1-4	250	4-7	200
2-3	500	5-6	300
2-5	300	6-7	250
3-4	350		

〈표 12〉 B망의 트래픽 요구 행렬

from \ to	1	2	3	4	5	6	7
1	0	50	45	20	50	30	0
2	40	0	100	50	30	0	25
3	0	0	0	0	30	30	20
4	30	20	100	0	0	20	40
5	10	10	0	70	0	0	10
6	0	70	40	10	10	0	0
7	20	0	0	10	30	20	0

다음 〈표 13〉는 문제 각각에 대해서 10번 수행한 결과를 평균으로 정리해 놓은 표이다.

〈표 13〉 가상경로 실험 결과

	Time (sec)	Cost
A망	795.82	7.2827
B망	2921.98	24.9200

A망의 경우에 초기해의 목적함수 값은 24.9142였는데 10번의 실험에서 모두 동일하게 목적함수의 값이 7.2827인 해를 나타내는 매우 안정된 해를 보였다. 그리고 B망의 경우에도 초기해의 목적함수 값은 756.2293이었는데 10번의 실험에서 거의 편차를 보이지 않고 평균 24.92의 최종해를 나타내었다.

가상 경로가 설정되고 난 후의 각 링크의 남은 용량은 아래 〈표 14〉, 〈표 15〉과 같다.

〈표 14〉 A망의 링크의 남은 용량

링크	용량	링크	용량
1-2	125	2-3	158
1-3	168	2-4	66
1-4	52	3-4	42

〈표 15〉 B망의 링크의 남은 용량

링크	용량	링크	용량
1-2	42	3-6	54
1-4	83	4-7	49
2-3	67	5-6	165
2-5	212	6-7	59
3-4	64		

A망의 경우 초기해는 각 노드쌍의 최단경로부터 용량을 할당하여 좋은 해를 초기해로 잡았고, B망은 무작위로 초기해를 잡았다. A망의 초기해의 링크의 남은 용량의 합은 495, B망은 87이었는데, 최종해의 경우 A망은 611, B망은 795로써 초기해에 관계없이 매우 좋은 해를 도출시켰다.

이처럼 개선된 SA1알고리듬은 본 가상경로 문제에 매우 안정되고 융통성 있게 잘 적용됨을 알 수 있었다.

5. 결론 및 추후 연구 과제

본 논문에서는 SA 알고리듬의 효과적인 변형을 통한 가속화를 시도하고 대표적인 조합적 최적화 문제에서 타당성을 검증한 후 얻어진 알고리듬을 가상경로 설정문제에 적용하는 과정을 보였다.

먼저 SA알고리듬의 효과적인 변형에 있어서는 TSP, GPP의 실험 결과에서 알 수 있었듯이 기존의 SA알고리듬에 비하여 해의 질을 떨어뜨리지 않으면서 계산 시간이 많이 향상되었고, SE보다는 두 가지 측면에서 모두 우월하였다 그리고 변형된 SA알고리듬을 사용하여 가상경로 설정 문제에 적용한 결과 변형된 SA알고리듬이 가상경로 설정 문제에 매우 안정적이고, 융통성 있게 적용됨을 알 수 있었다.

본 논문에서는 SA알고리듬에서 컨트롤 파라미터 T를 내릴 때 수렴속도를 고려하여 내리는 비율이 일정한 기하학적인 쿨링 스케줄을 사용하였으나, 온도를 내리는 비율을 매번 같게 할 것이 아니라 Entropy개념을 사용하여 매 단계에서의 비율을 결정하는 연구가 뒤따라야 할 것이고, 가상 경로 설정 문제에 있어서는 망의 성능평가 척도로 다른 적절한 척도를 찾아내고 본 논문에서 고려하지 않은 서비스 등급별 QOS등을 고려하여 SA를 이용한 가상경로를 구축하는 연구가 진행되어야 할 것이다.

참 고 문 헌

[1] E. H. L. Aarts & P. J. M. Van Laarhoven, *Simulated Annealing: Theory and Applications*, Reidel, Boston, 1987.

[2] S. Ahn, R. P. Tsang, S. R. Tong, and David H. C. Du., "Virtual Path Layout Des ign on ATM Networks", *INFOCOM'94*, 1994.

[3] A. Arvidsson, "Management Of Reconfigurable Virtual Path Networks", *ITC 14*, 1994.

[4] B. Hajek, "Cooling Schedules For Optimal Annealing", *Mathematics of Operations Research*, Vol. 13, No. 2, pp. 311-329, 1988.

[5] I. Chlamtac, A. Farago and T. Zhang, "How to Estabilish and Utilize Virtual Paths in ATM Networks", IEEE, 1993.

[6] David S. Johnson, C. R. Aragon, L. A. Mcgeoch, and C. Scevon, "Optimization

by Simulated Annealing : An experime ntal evaluation : Part I, graph partitioning", *Operations Research*, Vol. 37, No. 6, pp. 865-892, 1989.

[7] P. C. Fetterolf and G. Anandalingam, "Optimizing Interconnections of Local Ar- ea Networks : An Approach Using Simulated Annealing", *ORSA Journal on Compu ting*, Vol. 3, No. 4, pp. 275-287, 1991.

[8] M. D. Huang, F. Romeo and A. Sangiovanni-Vincentelli, "An efficient general cooling schedule for simulated annealing", *Proc. Int. Conference on CAD*, 1986.

[9] M. J. Lee and Yee, J. R., "An Efficient Near-Optimal Algorithm for the Joint Tra ffic and Trunk Routing Problem in Self-Planning Networks", *GLOBECOM'89*, 127-135, 1989.

[10] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing", *Science*, v. 220, pp. 671-680, May 1983.

[11] D. Mitra, F. Romeo and A. Sangiovanni-Vincentelli, "Convergence and Finite-Time Behavior of Simulated Annealing", *Journal of Applied Probability*, v. 18, pp. 747-771, 1986.

[12] Youssef G. Saab and Vasant B. Rao, "Combinational Optimization by Stochastic Evolution", *IEEE Transations On Computer-Aided Design*, Vol. 10, No. 4, pp. 525-535, 1991.

[13] H. Suzuki and F. A. Tobagi, "Fast Bandwidth Reservation Scheme With Multi-Link & Multi-Path Routing In ATM Networks", *INFOCOM'92*, 1992.

[14] 주종혁, *ATM*망에 있어서 가상경로의 동적 재구성에 관한 연구, 서울 대학교 공학 박사 학위논문, 1994.

[15] 윤복식, 송낙운, "Simulated Annealing의 효과적인 변형 및 HLS에의 적용," 대한산업공학회지, 21권, 1호, pp. 33-49, 1995.