Journal of Korean Institute of Intelligent Systems, Vol. 22, No. 4, August 2012, pp. 414-424
http://dx.doi.org/10.5391/JKIIS.2012.22.4.414

난이도-거리 상관관계 기반의 문제 인스턴스 공간 분석

Analyzing Problem Instance Space Based on Difficulty-distance Correlation

전소영* · 김용혁**†

So-Yeong Jeon and Yong-Hyuk Kim*†

*한국과학기술원 전산학과

**광운대학교 컴퓨터소프트웨어학과

요 약

문제 인스턴스 탐색 혹은 자동 생성은 알고리즘 분석 및 테스트에 적용될 수 있으며, 하드웨어, 소프트웨어 프로그램, 계산이론 등 다양한 수준에서 연구되어온 주제이다. 본 연구에서는 해(解) 공간에 사용된 목적값-거리 상관관계 분석을 문제 인스턴스 공간에 적용하였다. 문제 인스턴스의 목적값은 문제에 따라 알고리즘의 수행 시간과 최적해를 잘 구하는 정도로 정의하였다. 이러한 정의는 문제 인스턴스의 난이도로 해석할 수 있다. 상관관계는 3가지 측면에서 분석하였다: 첫째, 알고리즘과 거리 함수에 따른 상관관계 차이, 둘째, 알고리즘의 개선 전/후의 상관관계 변화, 셋째, 문제 인스턴스 공간과 해당 문제의 해 공간 사이의 연관성. 본 논문은 문제 인스턴스 공간에 상관계수 분석이 어떻게 적용될 수 있는지 보여주며, 문제 인스턴스 공간 분석을 본격적으로 다루는 첫번째 시도이다.

키워드 : 문제 인스턴스 공간, 목적값-거리 상관관계, 정렬 문제, 0/1 배낭 문제, 순회판매원 문제

Abstract

Finding or automatically generating problem instance is useful for algorithm analysis/test. The topic has been of interest in the field of hardware/software engineering and theory of computation. We apply objective value-distance correlation analysis to problem spaces, as previous researchers applied it to solution spaces. According to problems, we define the objective function by (1) execution time of tested algorithm or (2) its optimality; this definition is interpreted as difficulty of the problem instance being solved. Our correlation analysis is based on the following aspects: (1) change of correlation when we use different algorithms or different distance functions for the same problem, (2) change of that when we improve the tested algorithm, (3) relation between a problem instance space and the solution space for the same problem. Our research demonstrates the way of problem instance space analysis and will accelerate the problem instance space analysis as an initiative research.

Key Words : Problem instance space, objective value-distance correlation, sorting problem, 0/1 knapsack problem, travelling salesperson problem

1. 서 론

문제 인스턴스(problem instance)는 문제를 푸는 알고리즘에 구체적으로 주어진 입력 데이터이다. 가령 (1,4,3,2) 이라는 순열을 (1,2,3,4)라는 순열로 정렬해야 한다면 (1,4,3,2)가 문제 인스턴스가 된다. 문제 인스턴스 탐색(자동 생성)은 알고리즘의 결함이나 최적/최악 성능을 실험적으로 증명하기 위해 수행된다. 문제 인스턴스 탐색은 문제 인스턴스에 관해 정의된 목적 함수로 얻은 목적값(objective value)을 최대화 혹은 최소화하는 문제 인스턴스를 찾는 문제로 정식화(formulate) 될 수 있다. 예를 들어 알고리즘의 최악의 성능을 알기 위해서 문제 인스턴스의 목적 함수를 문제 인스턴스를 푸는데 걸리는 시간으로 정의해보자. 그러면 목적값을 최대화하는 인스턴스를 탐색함으로써 최악의 성능을 실험적으로 확인할 수 있다.

문제 인스턴스와 알고리즘을 다양한 수준에서 추상화시킨다면, 문제 인스턴스 탐색은 다양한 분야에서 주목받는 연구 주제이다. 논리 회로 분야에서는 Saab 등[1]과 Srinvas 등[2], 그리고 Rudnick 등[3]이 조합 회로 혹은 순서 회로의 폴트(fault)를 찾기 위해 회로의 입력으로 들어갈 테스트 벡터 혹은 시퀀스를 생성하고자 하였다. 마이크로프로세서 분야에서는 Corno 등[4]이 마이크로프로세서의 검사(validation)를 위해 테스트

접수일자: 2011년 12월 28일
심사(수정)일자: 2011년 12월 28일
게재확정일자 : 2012년 7월 26일
† 교신 저자
이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임 (2011-0004215)

프로그램을 생성하고자 하였다. 소프트웨어 분야에서 수행된 문제 인스턴스 탐색 연구들은 McMinn[5]이 조사한 것을 참고하자. 알고리즘 분야에서는 Cotta와 Moscato[6], Hemert[7]가 시간 복잡도 측면에서 알고리즘의 최악의 문제 인스턴스를 찾고자 하였다. Johnson과 Kosoresow[8], Jeon과 Kim[9]은 최적성(optimality)의 관점에서 최악의 문제 인스턴스를 찾고자 하였다. 한편, 최선의 문제 인스턴스[1] 탐색 또한 중요함을 보여준 연구도 있었다[10].

임의의 프로그램(알고리즘)에 대해 주어진 목적값을 최대/최소화하는 입력 데이터(문제 인스턴스)를 탐색하는 문제는 McMinn[5]이 언급한 것처럼 결정 불능 문제(undecidable problem)이다. 이러한 문제에 대해서는 유전 알고리즘(genetic algorithm, GA)[11]과 같은 메타휴리스틱 혹은 이에 기반한 알고리즘이 널리 사용되어 왔다.([1, 2, 3, 4, 6, 7, 8] 모두 GA에 기반한 탐색 알고리즘을 사용하였다.) 그러나, 문제 인스턴스를 탐색하는 연구가 이처럼 다양함에도 불구하고, 탐색해야할 문제 인스턴스 공간에 대한 연구는 저자가 아는 한 거의 없는 것으로 보인다.

문제 인스턴스 공간을 연구해야 하는 이유는 무엇인가? 이는 탐색 공간에 대한 지식을 쌓아 탐색을 보다 빠르게 하기 위한 것이다. 해(解) 공간의 분석의 경우, Boese 등[12]은 순회판매원 문제(travelling salesperson problem, TSP)와 그래프 이등분 문제에 대해 비용-거리(cost-distance) 상관관계를 분석하여 해 공간의 목적값이 그리는 지형을 추측하였다. Kim과 Moon[13]은 [12]의 해 공간 분석을 좀더 확장하여 그래프 이분할 문제에서 지역 최적점(local optimum)들의 중앙점이 우수한 목적값을 가짐을 보였다. 이는 해 공간의 분석을 통해 얻어낸 지식이 해의 탐색에 도움을 준 사례이다. 문제 인스턴스 공간의 분석도 문제 인스턴스 공간의 탐색에 도움을 주기 위한 것이다.

어떤 방법으로 문제 인스턴스 공간을 분석할 것인가? 본 논문은 [12]에서 사용되었던 상관관계 분석 방법을 사용한다. [12]의 분석 기법은 진화연산 분야에서 탐색 공간의 연구에 널리 사용되어온 기법이다[14, 15, 16]. 분석을 통해 얻은 상관관계를 이용하면, 유전 알고리즘으로 주어진 공간을 탐색하는 것이 얼마나 어려운 지 가늠할 수 있다[14]. 본 논문에서는 문제 인스턴스 공간을 탐색 공간으로 삼아 상관관계를 분석하려고 한다.

Jeon과 Kim[17]은 [12]의 상관관계 분석 기법을 문제 인스턴스 공간에 적용하여 0/1 배낭 문제 (0/1 knapsack problem, 0/1KP)의 탐욕 알고리즘을 분석하였다. 그러나 [17]에서는 분석 대상 알고리즘, 그리고 목적 함수 및 거리 함수를 한 가지로 고정하였고, 알고리즘 개선 후의 상관관계 분석은 수행되지 않았다. 문제 인스턴스 공간은 분석 대상 알고리즘, 거리 함수, 목적 함수에 따라서 결정되므로, 이러한 요소들이 상관관계 분석에 어떤 영향을 미치는 지 알아보는 것이 문제 인스턴스 공간을 분석하는 기본적인 접근일 것이다. 또한 위의 요소들에 대한 분석은 일반적으로 모든 문제에 활용할 수 있다. 본 논문은 이러한 요소들을 토대로 상관관계 분석을 다양한 측면에서 수행하되, 잘 알려진 몇 가지 문제를 예로 들어 그 결과를 보일 것이다.

2절에서는 상관관계 분석 방법을 설명한다. 3절에서는 정렬 문제(sorting problem, SORT)에 대해, 다양한 정렬 알고리즘, 다양한 거리 함수를 적용하여 문제 인스턴스 공간을 분석한다. 4절에서는 0/1 배낭 문제를 푸는 탐욕 알고리즘의 개선 후-문제 인스턴스 공간과 개선 전-문제 인스턴스 공간을 비교한다. 5절에서는 순회판매원 문제의 문제 인스턴스 공간과 해 공간을 비교해 볼 것이다. 마지막으로 6절에서 결론을 맺는다.

2. 실험 방법: 지역 최적점과 상관계수

본 논문에서는 두 가지 목적 함수를 다룬다. 하나는 주어진 알고리즘으로 해당 문제 인스턴스를 풀었을 때 기본 연산(basic operation)의 수행횟수이다. 다른 하나는 최적화 문제에 적용되는 목적 함수이다. 주어진 문제 인스턴스를 풀었을 때, 알려진 최적 알고리즘으로 구한 최적값과 주어진 알고리즘으로 구한 최적값을 비교하여 목적값을 계산한다. 최적 알고리즘이 없다면 최적에 가장 가까운 것으로 알려진 알고리즘을 사용할 수 있다.

본 논문에서는 주어진 알고리즘이 해당 문제 인스턴스를 빠르게 풀거나 최적해와 가까운 해를 얻는 경우에 목적값이 낮아지도록 정하였다. 정의에 의해, 목적값이 높은 문제 인스턴스는 주어진 알고리즘이 풀기에 어려운 문제 인스턴스라고 볼 수 있으므로 본 논문에서 정의한 목적 함수들은 난이도 관점에서 정의한 것이다.

[12]에서 사용된 상관관계 분석 방법은 탐색 공간 상의원소들 간의 거리와 목적 함수 사이의 상관계수를 분석하는 방법이다. 본 논문은 문제 인스턴스 공간을 탐색 공간으로 다루므로 임의의 두 문제 인스턴스 A와 B 사이의 거리, $d(A,B)$를 정의해야 하며, 본 논문에서는 이를 거리 함수로 부른다. 거리 함수는 두 점 간의 유사도를 나타내기 위한 것으로, 해 공간에서 다루어 왔던 거리 함수들을 참고해서 정의할 수 있다.

문제 인스턴스 공간은 다루기 어려울 정도로 크다(intractable). 따라서 상관관계를 분석하기 위해서는 문제 인스턴스를 탐색 공간상에서 일부 추출해 분석한다. 본 논문에서는 [12, 13]과 같이 지역 최적점들을 탐색 공간상에서 추출하여 상관관계를 분석한다. 본 논문에서의 탐색 공간은 문제 인스턴스 공간임을 상기하자. 어떤 점에 대한 지역 최적점이란 주어진 점과 가까운 점들 중 가장 우수한 목적값을 가진 점을 의미한다. 우수한 목적값이란, 목적값을 최대화하는 문제 인스턴스를 탐색하는 경우 목적값이 높은 것을 의미하며, 목적값을 최소화하는 문제 인스턴스를 탐색하는 경우는 목적값이 낮은 것을 의미한다. 본 논문에서는 목적값을 최대화하는 문제 인스턴스를 탐색하는 것과 목적값을 최소화하는 문제 인스턴스를 탐색하는 것을 각각 '목적값 최대화'와 '목적값 최소화'로 표기한다. 지역 최적화 알고리즘 (local optimization algorithm)이란 주어진 점의 지역 최적점을 찾는 알고리즘이며, 가까운 거리에 대한 기준과 탐색 방법에 따라 다양한 알고리즘을 생각

1) 최적성의 관점에서 최선의 문제 인스턴스란, 주어진 알고리즘이 알려진 최적해에가장 가까운 해를 찾는 문제 인스턴스를 의미하며, 최악의 인스턴스는 이와 정확히 반대 개념이다.

할 수 있다.

이제, 거리 함수와 목적 함수, 그리고 지역 최적화 알고리즘이 주어졌다고 하자. 문제 인스턴스 공간으로부터 추출한 지역 최적점들의 집합을 Ω라고 하자. 가장 우수한 목적값을 지닌 지역 최적점은 두 개 이상일 수 있으므로 이 집합을 Ω^{Best}라 하자. 지역 최적점 $A \in \Omega$에 대해 dOthers(A)와 dBest(A)를 다음과 같이 정의하자.

$$\mathrm{dOthers}(A) = \frac{\sum_{B \in \Omega \setminus A} d(A,B)}{|\Omega \setminus A|} \quad (1)$$

$$\mathrm{dBest}(A) = \min_{B \in \Omega^{Best}} d(A,B) \quad (2)$$

그러면 다음과 같은 두 가지 상관관계를 생각할 수 있다.

(1)지역 최적점의 목적값과 dOthers(.)값

(2)지역 최적점의 목적값과 dBest(.)값

상관관계(1),(2)를 조사했을 때 강한 상관관계가 나타날수록 유전 알고리즘으로 목적값을 최적화하는 문제 인스턴스를 찾는 것이 쉽다고 알려져 있다[14]. 강한 상관관계란, 목적값 최대화의 경우 목적값이 높을수록 dOthers(.)와 dBest(.)값이 작은 경향을 의미하며 목적값 최소화의 경우 목적값이 낮을수록 dOthers(.)와 dBest(.)값이 작은 경향을 의미한다.

각 실험의 상관관계 분석은 임의로 추출한 문제 인스턴스들 10,000개에 대한 지역 최적점들로 수행된다(중복된 지역 최적점은 제거됨). 실험에 사용한 지역 최적화 알고리즘과 거리 함수에 대해서는 각 문제 별로 좀 더 자세히 언급한다.

3. 정렬 문제: 다양한 거리 함수 및 알고리즘

본 논문에서는 정렬 문제를 주어진 순열의 원소들을 오름차순으로 정렬하는 것으로 정의한다. 실험에서는 크기 N의 문제 인스턴스를 1부터 N까지의 자연수를 가진 순열로 나타낸다. 순열 간의 거리와 정렬 알고리즘은 다양하게 알려져 있으므로, 정렬 문제는 거리와 알고리즘에 따른 상관관계를 비교 분석하기에 적합한 문제이다.

본 논문에서는 퀵 정렬, 머지 정렬, 힙 정렬, 삽입 정렬[18], 쉘 정렬[19], 개선된 퀵 정렬[18]을 다룬다. 개선된 퀵 정렬은 주어진 (부분)수열에서 임의로 3개 원소를 뽑아 이들 중 중앙값을 피봇[18]으로 삼는다. 쉘 정렬에서 사용할 증분 수열(sequence of increments)은 아래 $\{h_n\}$ 수열의 역순열이며, 수열의 상한은 문제의 크기 N이다[20].

$$h_n = \begin{cases} 1, & \text{if } n = 0 \\ 3h_{n-1}+1, & \text{if } n > 0 \end{cases} \quad (3)$$

문제 인스턴스의 목적값은 주어진 알고리즘으로 정렬했을 때 수행한 비교 연산 횟수로 정의한다[2]. 지역 최적화 알고리즘은 주어진 순열에 대해 가능한 모든 원소들의 쌍들을 하나씩 차례대로 조사해 목적값이 개선되는 경우에 한해 원소들의 위치를 서로 바꾼다. 더 이상 개선이 없거나 일정한 개선 횟수에 도달하면 이 알고리즘은 종료된다 (알고리즘 1을 참고). 두 문제 인스턴스(순열) 간의 거리는 원소들이 놓인 인덱스를 기준으로 맨하탄 거리(Mahattan distance)$d_{Manhattan}$, 해밍 거리(Hamming distance) $d_{Hamming}$, 스왑 거리(swap distance) d_{swap}을 사용하며, 각각 다음과 같이 정의될 수 있다.

$$d_{Manhattan}(A,B) := \sum_{i=1}^{N} |A[i] - B[i]| \quad (4)$$

$$d_{Hamming}(A,B) := \sum_{i=1}^{N} diff(A[i],B[i]) \quad (5)$$

여기서, $diff(A[i],B[i]) = \begin{cases} 0, & \text{if } A[i] = B[i] \\ 1, & otherwise \end{cases}$.

Input: 정렬 문제의 인스턴스, 순열 A.
Output: A의 지역 최적점.
//N: 순열 A의 크기
//$A[i]$: 순열 A의 i번째 원소
개선 횟수=0;
while(개선 횟수< $\lfloor N \times 0.25 \rfloor$)
{
 changed=false;
 순열 A에서 두 원소를 고르는 모든 조합 중 차례대로 하나 를 택하여 다음을 수행:
 {
  //$A[i]$와 $A[j]$를 선택했다고 하자.
  $A[i]$와 $A[j]$의 자리를 바꿈;
  if(A의 목적값에 변화가 없거나 개선[1]이 없음)
   $A[i]$와 $A[j]$의 자리를 원래대로 되돌림;
  else
  {
   changed=true;
   개선횟수를 1 증가;
   if(개선횟수가 $\lfloor N \times 0.25 \rfloor$ 에 도달)
    return A;
  }
 }
 //모든 조합을 돌아본 후 다음을 검사
 if(changed 가 false)
  return A;
}

[1]: 목적값이 개선되었다는 것은 목적값 최대화의 경우, 목적값이 증가되었다는 것이며, 목적값 최소화의 경우는 그 반대임.

알고리즘 1. 정렬 문제 인스턴스 공간에서 사용한 지역 최적화 알고리즘.

Algorithm 1. Local optimization algorithm adopted for sorting problem instance space.

$d_{swap}(A,B) :=$ 순열 A가 순열 B로 변환되기 위해 A가 가진 두 원소의 자리를 서로 바꾸는 것을 반복할 때, 자리를 바꾸는 최소 횟수 (6)

여기서, $A[i]$는 순열의 i번째 원소를 의미한다.

2) 단, 개선된 퀵 정렬은 비교횟수가 난수에 따라 바뀌므로 50번 독립적으로 정렬해 얻은 비교횟수의 평균을 목적값으로 정의한다.

3.1 실험 범위 및 분류

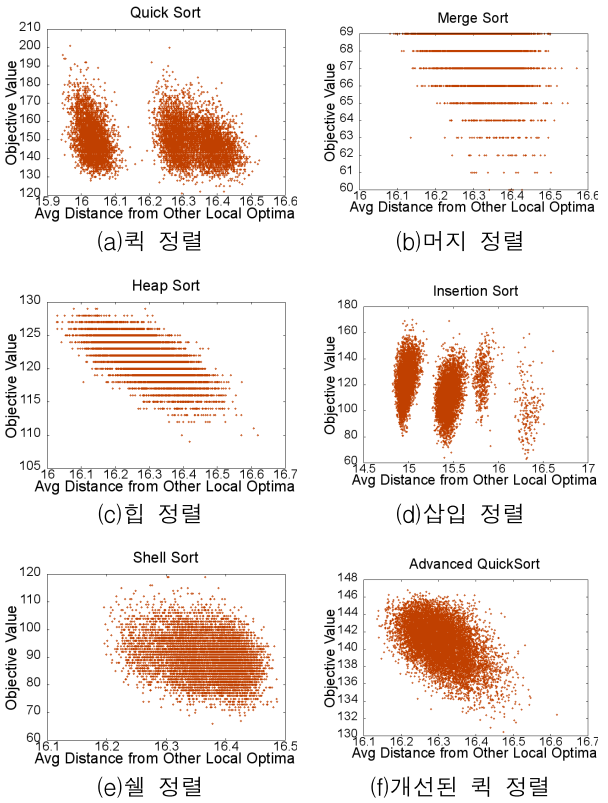
실험은 순열의 크기(10, 20, 30, 40)과 정렬 알고리즘의 종류(앞에서 언급한 6가지 알고리즘), 목적값에 대한 최적화 방향(목적값 최대화, 목적값 최소화), 적용한 거리 함수에 따라 분류된다. 단, 개선된 퀵 정렬은 크기 20까지만 실험하였다. 지역 최적화 알고리즘에서 문제 인스턴스의 최대 개선 횟수는 순열의 크기 N의 경우 $\lfloor 0.25 \times N \rfloor$ 이다.

3.2 실험 결과 및 분석

표 1은 수열의 크기 40에서 문제 인스턴스 공간 분석에서 얻은 통계이다. 통계량에서 Q는 주어진 목적 함수로 얻은 목적값을 의미하고 D는 주어진 거리 함수로 얻은 두 문제 인스턴스의 거리값을 의미한다. 'max', 'min', 'avg', 'std'는 각각 최대값, 최소값, 평균, 표준편차를 의미한다. 'Cov'과 'Corr'은 각각 공분산과 상관계수를 의미한다. 예를 들면, 'Corr(dOthers, Q)'는 지역 최적점들의 목적값(Q)과 'dOthers' 간의 상관계수를 의미한다.

표 1. [정렬 문제] 문제 인스턴스 크기 40일 때의 문제 인스턴스 공간에 관한 통계
Table 1. [SORT] Statistics of problem instance space for instance size 40

		인스턴스 크기 40									
		최대화					최소화				
정렬 알고리즘		퀵	머지	힙	삽입	쉘	퀵	머지	힙	삽입	쉘
# of local optima		10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000
# of best local optima		1	5545	1	1	1	2	2	1	1	1
best Q		531	177	336	581	329	240	133	264	268	145
avg Q		393.96	175.70	321.38	435.81	251.77	266.42	152.13	285.39	415.01	177.09
std Q		23.87	1.94	4.16	42.72	18.22	6.24	3.78	5.49	42.67	10.27
맨하탄 거리	max D	782	772	778	722	788	768	780	738	710	798
	min D	238	208	226	192	200	222	188	206	182	228
	avg D	525.97	514.78	524.53	468.39	528.97	505.62	524.58	472.15	460.27	522.44
	std D	55.13	57.47	54.84	52.10	57.67	56.43	61.54	52.79	50.68	55.61
	Cov(dOthers.Q)	-8.25	-4.67	-12.08	106.32	-23.72	48.25	9.71	37.77	-150.40	21.10
	Corr(dOhters,Q)	-0.05	-0.23	-0.39	0.24	-0.19	0.52	0.26	0.51	-0.38	0.24
스왑 거리	max D	39	39	39	39	39	39	39	39	39	39
	min D	23	24	24	22	24	24	24	23	23	24
	avg D	35.62	35.64	35.66	34.00	35.70	35.44	35.66	34.43	34.12	35.66
	std D	1.66	1.66	1.65	1.74	1.64	1.71	1.66	1.79	1.74	1.65
	Cov(dOthers.Q)	-0.26	-0.02	-0.11	-3.61	-0.10	0.42	0.05	0.64	1.18	0.11
	Corr(dOhters,Q)	-0.11	-0.17	-0.58	-0.23	-0.19	0.35	0.15	0.30	0.18	0.17
해밍 거리	max D	40	40	40	40	40	40	40	40	40	40
	min D	29	28	30	27	30	28	28	28	27	29
	avg D	38.89	38.92	38.93	37.27	38.99	38.72	38.95	37.77	37.83	38.94
	std D	1.05	1.05	1.03	1.23	1.01	1.13	1.04	1.27	1.47	1.03
	Cov(dOthers.Q)	-0.28	-0.02	-0.11	-3.34	-0.08	0.41	0.04	0.49	0.97	0.11
	Corr(dOhters,Q)	-0.12	-0.16	-0.61	-0.19	-0.16	0.34	0.15	0.23	0.18	0.16



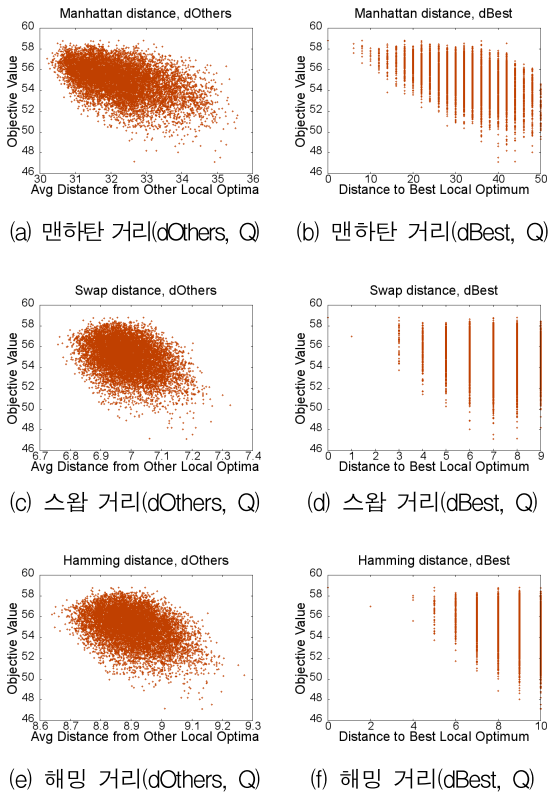
(a)퀵 정렬 (b)머지 정렬

(c)힙 정렬 (d)삽입 정렬

(e)쉘 정렬 (f)개선된 퀵 정렬

그림 1. [정렬 문제, 크기 20, 목적값 최대화, 스왑 거리] 각 알고리즘 별 목적값-거리 상관관계

Fig. 1. [SORT, size 20, objective value maximization, swap distance] Objective value-distance correlation for each sorting algorithm



(a) 맨하탄 거리(dOthers, Q) (b) 맨하탄 거리(dBest, Q)

(c) 스왑 거리(dOthers, Q) (d) 스왑 거리(dBest, Q)

(e) 해밍 거리(dOthers, Q) (f) 해밍 거리(dBest, Q)

그림 2. [정렬 문제, 크기 10, 목적값 최대화, 개선된 퀵 정렬[18]] 거리 함수 별 목적값-거리 상관관계

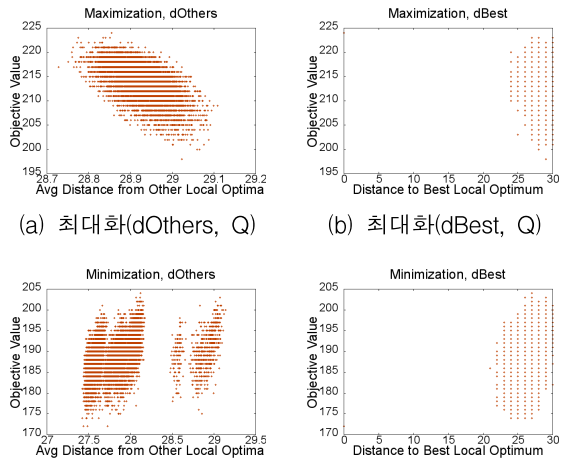
Fig. 2. [SORT, size 10, objective value maximization, advanced quick sort[18]] Objective value-distance correlation for each distance function

표 1을 참고할 때, 특정 거리 함수가 다른 거리 함수에 비해 모든 알고리즘에서 강한 상관관계를 보이지는 않는다.

예를 들면 맨하탄 거리는 퀵 정렬에서 목적값 최소화로 얻은 상관관계가 다른 거리 함수에 비해 강하다. 해밍 거리는퀵 정렬에서 목적값 최대화로 얻은 상관관계가 다른 거리 함수에 비해 강하다. 스왑 거리는 삽입 정렬에서 목적값 최대화로 얻은 상관관계가 다른 거리 함수보다 강하다.

강한 상관관계가 나타나는 거리 함수를 사용할수록 유전 알고리즘을 이용한 문제 인스턴스 탐색이 보다 쉬워지게 됨을 상기해보면, 논문에서 구한 실험결과는 같은 정렬 문제라도 주어진 알고리즘에 따라서 적합한 거리 함수가 다를 수 있음을 보여준다.

그림 1은 문제 인스턴스 크기 20에서 스왑 거리를 적용하였을 때 알고리즘 별 상관관계(dOthers, Q)를 나타낸다.



(a) 최대화(dOthers, Q) (b) 최대화(dBest, Q)

(c) 최소화(dOthers, Q) (d) 최소화(dBest, Q)

그림 3. [정렬 문제, 크기 30, 해밍 거리, 힙 정렬] 목적함수 최대화와 최소화일 때의 목적값-거리 상관관계

Fig. 3. [SORT, size 30, Hamming distance, heap sort] Objective value-distance correlation for objective value maximization and for minimization

같은 목적 함수의 정의와 실험 조건이라도 알고리즘에 따라 완전히 다른 상관관계를 보여주고 있음을 알 수 있다. 이는 정렬 문제의 인스턴스 탐색이 알고리즘에 따라서 쉬워질 수도 혹은 어려워질 수도 있음을 의미한다. 이러한 결과는 퀵 정렬의 최악의 경우(the worst case)가 삽입정렬의 최악의 경우와 다른 것처럼 알고리즘 마다 최악의 경우가 다르다는 사실과 부합한다.

그림 2는 문제 인스턴스 크기 10, 개선된 퀵 정렬에서 목적값 최대화를 수행할 때 거리 함수 별 목적값-거리 상관관계를 나타낸다. 스왑 거리와 해밍 거리는 서로 비슷한 플롯팅을 보여주었다. 그림 3은 힙 정렬에 대해 문제 인스턴스 크기 30에서 해밍 거리를 적용했을 때 목적값의 최대화와 최소화에 따른 상관관계를 나타낸다. 이를 참고할 때, 목적값 최소화일 경우와 최대화일 경우의 상관관계는 연관성이 없음을 알 수 있다. 그림 3에서, 목적값 최대화의 경우 힙 정렬의 문제 인스턴스 공간이 목적값 최소화의 경우보다 강한 상관관계를 보이므로, 목적값을 최대화하는 문제 인스턴스를 찾는 것은 목적값을 최소화하는 문제 인스턴를 찾는 것보다 더 쉬움을 알 수 있다. 따라서, 문제 인스턴스 탐색에 있어서 목적값의 최적화 방향도 문제 인스턴스 공간의 모양을 결정하는 주요 요소임을 알 수 있다.

4. 0/1 배낭 문제: 알고리즘 개선 전과 후

0/1 배낭 문제[21]에서는 가치와 무게를 가진 물품들의 목록과 함께 배낭의 무게 한도가 주어진다. 배낭에 담기로 선택한 물품들이 이 문제의 해가 되며, 배낭의 물품들의 가치 합이 최대가 되는 해가 최적해이다. 본 논문에서는 무게 대비 가치(가치 밀도)가 높은 물품을 우선으로 선택하는 탐욕 알고리즘(가치밀도가 같은 경우 무게가 낮은 것을 우선)과 이를 개선한 탐욕 알고리즘[17]에 대해 살펴본다. 개선된 알고리즘은 기존 알고리즘이 배낭에 담기로 선택한 물품들과 선택하지 않은 물품들을 서로 교환해보는 방식으로 더 좋은 해를 찾는다.

Input: 0/1 배낭 문제의 인스턴스 A.
Output: A의 지역 최적점.
//$A[i]$: 인스턴스 A의 i번째 물품(i-th item)
for each i-th item $A[i]$
{
  //A[i]=(v,w)라고 하자
  다음 8개의 물품들을 고려하여 A의 목적값을 가장 개선시키는 물품을 택한다:(v,w+1),(v+1,w+1), (v+1,w), (v+1,w−1), (v,w−1), (v−1,w−1). (v−1,w). (v−1,w+1). (v,w)의 경우가 가장 목적값이 높다면 (v,w)를 선택한다;
  선택한 물품으로 $A[i]$를 바꾼다;
}
return A;

알고리즘 2. 0/1 배낭 문제 인스턴스 공간에서 사용한 지역 최적화 알고리즘.
Algorithm 2. Local optimization algorithm adopted for 0/1 knapsack problem instance space.

4절에서는 0/1 배낭 문제의 탐욕 알고리즘에 대해 어떤 유형의 상관관계가 나타날 수 있는 지 알아보고, 알고리즘의 개선이 문제 인스턴스 공간을 어떻게 바꾸는지를 살펴볼 것이다.

문제 인스턴스의 목적 함수는 $1-(P/O)$로 정의한다. 여기서 P는 분석할 알고리즘, 즉 탐욕 알고리즘이 구한 배낭의 물품들의 최대 가치 합이고, O는 알려진 최적 알고리즘이 구한 최대 가치 합이다. 본 논문에서는 문제 인스턴스를 물품들의 배열로 보고 각 물품을 (가치, 무게) 쌍으로 나타낸다. 정의에 의해 목적값은 0과 1 사이에 존재한다. 본 논문에서 사용한 지역 최적화 알고리즘은 각 물품에 대해 가치와 무게 값을 최대 ±1만큼 변화시켜 목적값을 개선한다(알고리즘 2를 참고). 두 문제 인스턴스 A와`$B$에서`각각 물품들이`가치`밀도`순으로`정렬`되어있다고``가정할 때, 두 문제 인스턴스 A와 B간의 거리 d_{01KP}는 다음과 같이 정의한다.

$$d_{01KP}(A,B) := \sum_{i=1}^{N} |pd(A[i]) - pd(B[i])| \qquad (7)$$

여기서, $A[i]$는 문제 인스턴스 A에게 주어진 i번째 물품을 가리키며, $pd(A[i])$는 $A[i]=(v,w)$의 가치밀도, 즉, $\frac{v}{w}$이다.

4.1 실험 범위 및 분류

문제의 정의에서 물품의 가치와 무게는 정수이다. 본 논문에서는 실험이 용이하도록 가치와 무게의 범위를 [1,100] 구간에 있는 정수로 제한한다. 실험은 주어진 물품들의 개수와 목적값의 최적화 방향(목적값 최대화, 목적값 최소화), 그리고 무게 제한 계수의 값(0.25, 0.5, 0.75)에 따라 분류한다. 무게 제한 계수는 주어진 모든 물품들의 무게 합에 곱하여 배낭의 무게 한도를 결정한다.

4.2 실험 결과 및 분석

표 2는 탐욕 알고리즘의 개선 전과 후에 대한 문제 인스턴스 공간의 통계이다. 통계량의 표기는 3절에서 설명한 것과 동일하다. 그림 4는 문제 인스턴스 크기 10, 목적값 최대화, 무게 제한 계수 0.25에서 탐욕 알고리즘의 개선 전과 개선 후 목적값-거리 상관관계를 나타낸다.

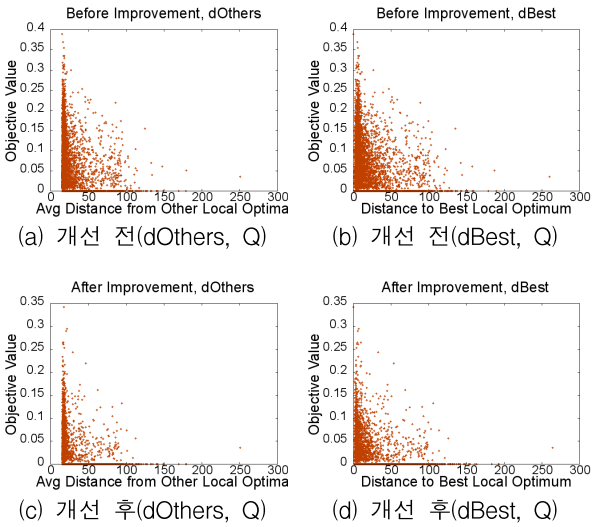
목적값이 0인 인스턴스가 다수 존재하므로 임의로 문제 인스턴스를 추출하는 것만으로도 목적값을 최소화하는 문제 인스턴스를 쉽게 찾을 수 있으며, 탐욕 알고리즘이 대부분의 문제 인스턴스에 대해 최적해를 구함을 알 수 있다. 이는 Jeon과 Kim[10]의 연구와 부합된다.

한편, 표 2를 볼때 알고리즘의 개선 여부와 관계없이 전반적으로 Corr(dOthers, Q) 값이 0에 가까울 정도로 매우 낮았다. 그림 4에서 dOthers(.)와 dBest(.)값이 작을 때에 다양한 목적값을 지닌 문제 인스턴스가 존재함을 알 수 있다. 이러한 상관관계에서는 유전 알고리즘으로 목적값을 최대화하는 문제 인스턴스를 탐색하는 것이 적합하지 않아 보인다.

표 2에서, 목적값의 평균(avg Q)과 최우수 목적값(best Q)를 보면 개선 후의 경우가 개선 전의 경우보다 값이 적은 것을 알 수 있어 알고리즘이 확실히 개선되었다는 것을 알 수 있다. 상관계수 값에서는 큰 차이가 없지만, 목적값이 전체적으로 감소했으므로 상관관계가

표 2. [0/1 배낭문제] 탐욕 알고리즘의 개선 전과 후[17]의 문제 인스턴스 공간에 관한 통계
Table 2. [0/1KP] Statistics of problem instance space before and after improvement[17] of a greedy algorithm

				# of best		best Q		avg Q		Corr(dOthers, Q)	
물품 개수	최대화/ 최소화	무게 제한 계수	# of local optima	개선 전	개선 후	개선 전	개선 후	개선 전	개선 후	개선 전	개선 후
10	최대화	0.25	10,000	1	1	0.39	0.34	0.03	0.01	-0.05	-0.03
		0.5	10,000	1	1	0.21	0.19	0.02	0.00	-0.04	-0.03
		0.75	10,000	1	1	0.16	0.15	0.01	0.00	0.01	0.01
	최소화	0.25	10,000	7,589	9,610	0.00	0.00	0.01	0.00	-0.03	0.00
		0.5	10,000	7,577	9,659	0.00	0.00	0.01	0.00	-0.01	-0.01
		0.75	10,000	8,361	9,868	0.00	0.00	0.00	0.00	0.03	0.02
20	최대화	0.25	10,000	1	1	0.18	0.11	0.02	0.01	-0.04	-0.03
		0.5	10,000	1	1	0.09	0.07	0.01	0.00	-0.04	-0.02
		0.75	10,000	1	1	0.06	0.04	0.01	0.00	-0.02	-0.02
	최소화	0.25	10,000	6,959	9,468	0.00	0.00	0.01	0.00	-0.01	0.01
		0.5	10,000	6,994	9,500	0.00	0.00	0.00	0.00	-0.01	0.00
		0.75	10,000	8,155	9,818	0.00	0.00	0.00	0.00	0.02	0.01
30	최대화	0.25	10,000	1	1	0.08	0.07	0.02	0.01	-0.03	-0.03
		0.5	10,000	1	1	0.05	0.04	0.01	0.00	-0.02	-0.01
		0.75	10,000	1	1	0.03	0.02	0.00	0.00	-0.01	-0.01
	최소화	0.25	10,000	6,805	9,490	0.00	0.00	0.00	0.00	-0.01	0.00
		0.5	10,000	7,078	9,554	0.00	0.00	0.00	0.00	0.02	0.01
		0.75	10,000	8,110	9,795	0.00	0.00	0.00	0.00	0.01	0.00



(a) 개선 전(dOthers, Q) (b) 개선 전(dBest, Q)

(c) 개선 후(dOthers, Q) (d) 개선 후(dBest, Q)

그림 4. [0/1 배낭 문제, 크기 10, 무게제한계수 0.25, 목적값 최대화] 탐욕 알고리즘 개선 전과 후[17]의 목적값-거리 상관관계

Fig. 4. [0/1KP, 10 items, weight coefficient 0.25, objective value maximization] Objective value-distance correlation before/after improvement[17] of a greedy algorithm

약해진 것으로 추측된다. 실험에서 사용한 개선된 탐욕 알고리즘[17]은 모든 경우에 대해서 개선 전 알고리즘보다 같거나 보다 우수한 해를 구한다. 이러한 경우 목적 함수의 정의에 따라서 목적값이 큰 인스턴스의 수가 줄어들 것이므로 목적값을 최대화 하는 문제 인스턴스를 탐색하는 것이 어려울 것이다. 이러한 점으로 볼 때 상관관계가 약해진 것은 놀랍지 않은 결과이다.

Input: 순회판매원 문제의 문제 인스턴스, 인접행렬 A.
Output: A의 지역 최적점. //$A(i,j)$: A의 (i,j) 원소
for each $A(i,j)$
{
 $A(i,j)+1$, $A(i,j)-1$, $A(i,j)$ 중 A의 목적값을
 가장 개선시키는 값을 선택한다;
 선택한 값으로 $A(i,j)$를 바꾼다;
}
return A;

알고리즘 3. 순회판매원 문제 인스턴스 공간에서 사용한 지역 최적화 알고리즘.

Algorithm 3. Local optimization algorithm adopted for travelling salesperson problem instance space.

5. 순회판매원 문제: 해(解)공간과 비교

순회판매원 문제[21]는 방문해야 할 N개 도시들 간의 여행비용이 주어질 때, 모든 도시를 한 번씩만 방문하는 여행 경로를 찾는 것이며, 가장 적은 비용이 드는 해가 최적해이다. 순회판매원 문제의 해 공간에 대한 연구는 많이 진행되어왔으며 Kim과 Moon[13]의 연구는 그 중 상관관계 분석을 활용한 연구이다. 이 절에서는 순회판매원 문제 해 공간의 잘 알려진 상관관계 분석 결과와 순회판매원 문제 인스턴스 공간의 상관관계를 비교분석하고자 한다.

본 논문의 실험에서는 문제 인스턴스를 $N \times N$ 인접행렬로 나타내었으며 인접행렬의 (i, j)원소는 i번째 도시로부터 j번째 도시까지 여행하는 데 드는 비용이다.

표 3. [순회판매원 문제] 문제 인스턴스 공간에 관한 통계
Table 3. [TSP] Statistics of problem instance space

인접 행렬	최대화 / 최소화	도시 수	# of local optima	# of best	best Q	avg Q	std Q	max D	min D	avg D	std D	Cov(dOthers, Q)	Corr(dOthers, Q)
대칭	최대화	5	10,000	1	0.32	0.00	0.01	1512.00	66.00	665.71	148.63	0.00	0.01
		10	10,000	1	0.70	0.04	0.07	4876.00	1372.00	3000.54	315.67	0.94	0.13
	최소화	5	10,000	9,787	0.00	0.00	0.01	1512.00	64.00	665.70	148.68	0.00	0.00
		10	10,000	5,327	0.00	0.04	0.07	4869.00	1371.00	2997.21	315.40	0.88	0.13
비대칭 허용	최대화	5	10,000	1	2.89	0.01	0.08	1332.00	182.00	667.08	105.37	0.17	0.07
		10	10,000	1	2.12	0.29	0.23	4251.00	1802.00	3005.35	223.56	2.81	0.17
	최소화	5	10,000	9,099	0.00	0.01	0.08	1332.00	182.00	666.63	105.31	0.13	0.05
		10	10,000	922	0.00	0.29	0.23	4230.00	1791.00	2991.52	222.91	2.49	0.15

표 4. [순회판매원 문제] 해 공간에 관한 통계
Table 4. [TSP] Statistics of solution space

인접 행렬	주어진 문제	도시 수	# of local optima	# of best	best Q	avg Q	std Q	max D	min D	avg D	std D	Cov(dOthers, Q)	Corr(dOthers, Q)
대칭	GA worst	5	4	2	5	53.50	48.50	4.00	3.00	3.33	0.47	0.00	N/A
		10	6	2	11	120.33	88.60	8.00	5.00	6.93	1.12	7.29	0.87
	GA best	5	2	2	126	126.00	0.00	4.00	4.00	4.00	0.00	0.00	N/A
		10	34	2	189	235.65	20.99	8.00	3.00	6.48	1.14	0.80	0.18
	Random	5	2	2	183	183.00	0.00	4.00	4.00	4.00	0.00	0.00	N/A
		10	18	2	175	229.33	25.63	8.00	3.00	6.38	1.32	4.77	0.69
비대칭 허용	GA worst	5	4	1	5	139.50	78.81	3.00	3.00	3.00	0.00	0.00	N/A
		10	1,071	1	10	275.35	61.23	8.00	2.00	6.70	1.01	5.46	0.82
	GA best	5	4	1	133	164.25	29.63	3.00	3.00	3.00	0.00	0.00	N/A
		10	1,098	1	115	271.94	46.21	8.00	2.00	6.76	1.00	2.58	0.62
	Random	5	2	1	135	149.00	14.00	3.00	3.00	3.00	0.00	0.00	N/A
		10	967	1	126	281.57	43.51	8.00	2.00	6.69	1.02	2.96	0.64

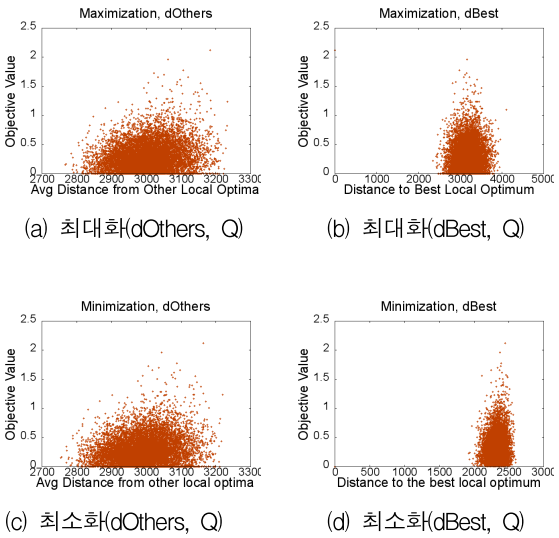
본 논문에서 분석할 알고리즘은 탐욕 접근법과 2-opt[11]를 결합한 것(이하 GR2opt)이다. 즉, 매 순간 가장 적은 비용으로 방문할 수 있는 도시를 선택하여 여행 경로를 구한다음 이 경로를 2-opt를 통해 개선하는 알고리즘이다. 본 논문에서는 이 문제의 인스턴스에 대해 목적 함수를 $(P/O)-1$와 같이 정의한다. P 는 GR2opt가 구한 최소여행비용이며, O는 알려진 최적 알고리즘이 구한 최소여행비용이다. 정의에 따라 목적값은 0 이상의 실수 값이다.

유클리드 순회판매원 문제 인스턴스와 달리 일반 순회판매원 문제 인스턴스는 도시의 위치가 주어진 것이 아니라 단지 도시 간의 방문 비용만이 주어진 것이므로 가중치가 있는 완전 그래프로 볼 수 있으며, 간선의 가중치 값을 원소로 하는 크기 $N\times N$의 벡터로 볼 수 있다. 따라서 문제 인스턴스 간의 거리로서 자연스럽게 생각할 수 있는 것은 문제 인스턴스를 벡터로 보고 다음과 같이 맨하탄 거리 $d_{Manhattan}$를 구하는 것이다.

$$d_{TSP}(A,B) := \sum_{i,j}|A(i,j)-B(i,j)| \quad (8)$$

여기서 $A(i,j)$는 인접 행렬 A의 (i,j)원소이다.

사용한 지역 최적화 알고리즘은 인접행렬의 각 원소를 차례대로 보고 최대 ±1만큼 변화시켜 목적값을 개선한다 (알고리즘 3 참고).



(a) 최대화(dOthers, Q) (b) 최대화(dBest, Q)

(c) 최소화(dOthers, Q) (d) 최소화(dBest, Q)

그림 5. [순회판매원 문제, 도시 수 10, 비대칭 인접행렬 허용] 목적값-거리 상관관계
Fig. 5. [TSP, 10 cities, allowing non-symmetric adjacency matrix] Objective value-distance correlation

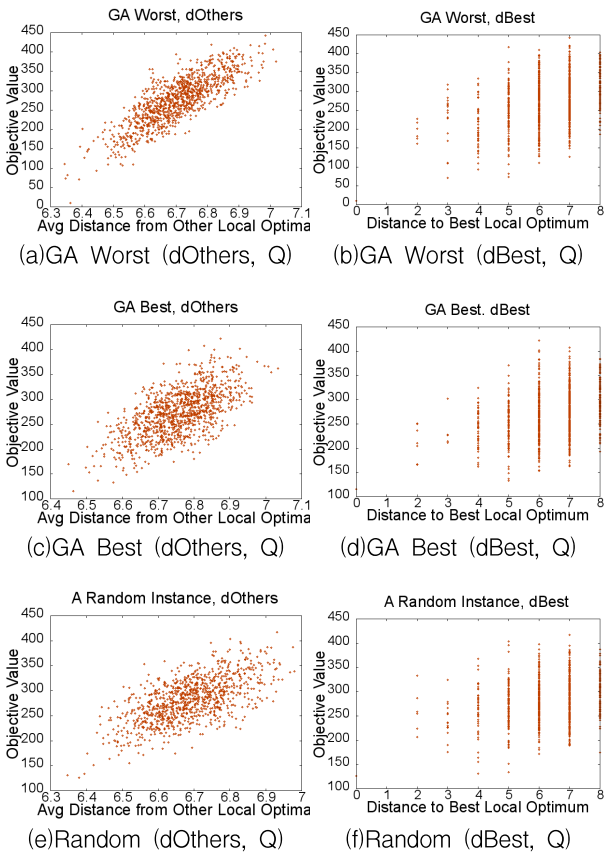


그림 6. [순회판매원 문제, 도시 수10, 비대칭 인접행렬 허용] GA Worst, GA Best, 그리고 임의의 문제 인스턴스에 대한 해 공간에서의 상관관계

Fig. 6. [TSP, 10 cities, allowing non-symmetric adjacency matrix] Objective value-distance correlation for solution space of GA Worst, GA Best and random problem instance

이 문제의 해는 방문 순서를 나열한 순열로 나타낸다. 해의 목적값은 해가 지시한 방문순서를 따라 여행한 비용이다. 지역 최적화 알고리즘으로는 2-opt를 사용한다. 해들 간의 거리는 d_{swap}(3절의 식(6))을 적용한다. 다만, 같은 해라도 최초의 방문 도시에 따라 순열이 달라질 수 있으므로 한쪽 해를 고정하고 다른 한쪽 해의 최초 방문 도시를 달리 하면서 구한 두 해 사이의 d_{swap} 값 중 최소값을 거리로 정의한다. 위와 같이 정의한 목적 함수와 거리 함수에 대해 해 공간의 목적값-거리 상관관계를 분석한다.

5.1 실험 범위 및 분류

문제의 정의에서 도시 간 여행비용은 실수 값이지만, 본 논문에서는 실험을 용이하게 하기 위해 정수로 제한하였다. 두 도시 간 방문 비용의 범위는 [1,100] 구간의 정수로 제한한다. 실험은 크게 목적값의 최적화 방향(목적값 최대화, 목적값 최소화), 인접행렬의 종류(대칭, 비대칭 허용), 도시 수(5, 10)에 따라 분류된다.

해 공간의 실험은 해를 탐색할 문제 인스턴스의 대칭성 유무(즉, 인접 행렬의 종류)와, 도시 수에 따라 분류된다. 문제 인스턴스는 3가지를 사용한다. 하나는 임의로 생성한 인스턴스이고, 나머지 둘은 저자들이 [9]에서 제시한 유전 알고리즘을 구현해 얻은 것 중 목적값이 가장 높은 문제 인스턴스('GA Worst'로 표기)와 목적값이 가장 낮은 문제 인스턴스('GA Best'로 표기)를 사용한다.

5.2 실험 결과 및 분석

표 3은 문제 인스턴스 공간에 대해 얻은 통계이다. 통계량의 표기는 3절에서 설명한 것과 동일하다. 표 4는 5.1절에서 제시한 실험 분류에 따라 3가지 문제 인스턴스의 해 공간에 대해 분석한 결과이다. 그림 5는 비대칭 허용 인접행렬로 표현된 도시 수 10의 문제 인스턴스 공간의 목적값-거리 상관관계를 나타낸다. 그림 5와 6을 참고하면 문제 인스턴스 공간이 해 공간과 다를 수 있음을 알 수 있다.

문제 인스턴스 공간과 달리 해 공간에서는 전체적으로 강한 양의 상관관계를 볼 수 있다. 이는 문제 인스턴스 공간이 해 공간에 관계없이 탐색하기 어려울 수 있음을 보여준다. 해 공간에서의 상관관계는 문제 인스턴스에 따라 차이가 있었는데, 'GA Worst' 문제 인스턴스가 다른 두 가지 문제 인스턴스보다 해 공간에서의 상관관계가 좀 더 강하게 나타났다(표 4 및 그림 6 참고). 다시 말해, GR2opt로 풀기 어려운 문제 인스턴스일 경우 유전 알고리즘으로 해당 문제 인스턴스의 해를 찾는 것은 다른 문제 인스턴스에 비해 쉬운 것으로 예측할 수 있다. 이러한 연관성은 좀 더 많은 실험을 통해 검증할 필요가 있지만, 적어도'GA worst'문제 인스턴스에 대해서는 해 탐색에 사용된 유전 알고리즘을 GR2opt에 결합함으로써 알고리즘을 쉽게 개선시킬 수 있을 것이다. 비슷한 논리로, 'GA Best'와 같은 목적값이 낮은 문제 인스턴스에 대해서는 시간 복잡도가 높은 메타휴리스틱보다 GR2opt를 적용하는 것이 최적해를 보다 효율적으로 구할 수 있을 것이다.

6. 결 론

본 논문에서는 문제 인스턴스 공간을 분석하고자 정렬 문제, 0/1 배낭 문제, 순회판매원 문제에 대해 목적값-거리 상관관계를 분석하였다. 목적값-거리 상관관계가 강할수록 주어진 목적 함수를 최적화하는 문제 인스턴스를 유전 알고리즘으로 탐색하는 것이 쉽다는 것이 알려져 있다. 본 논문에서 정의한 목적 함수는 분석 대상 알고리즘이 풀기에 어려운 문제 인스턴스를 탐색하고자 정의되었다. 해(解)들과 같이 문제 인스턴스들 또한 순열 혹은 벡터 등으로 추상화될 수 있으므로 상관관계 분석에 필요한 거리 함수와 지역 최적화 알고리즘은 해 공간에서 사용되었던 거리 함수와 지역 최적화 알고리즘의 발상을 기반으로 정의할 수 있다.

본 논문에서는 다음과 같은 측면에서 상관관계 분석을 수행하였다; 같은 문제를 푸는 서로 다른 알고리즘 및 서로 다른 거리 함수를 적용했을 때의 상관관계 변화, 알고리즘의 개선 전과 후의 상관관계 변화, 해 공간과의 연관성. 위와 같은 접근은 풀어야할 문제와 알고리즘에 관계없이 일반적으로 적용할 수 있다.

본 논문에서는 같은 문제 내에서도 알고리즘에 따라 상관관계를 강하게 나타내는 거리 함수가 다를 수 있음을 보았다. 예를 들면 본 논문에서 정의한 목적값을 최소화하는 문제 인스턴스를 탐색하기 위해서 퀵 정렬에 맨하탄 거리를 사용하는 것이 스왑 거리나 해밍 거리보다 적합하다는 것을 추측할 수 있었다. 또한 문제 인스턴스 공간은 알고리즘에 따라, 그리고 목적값을 최적화하는 방향에 따라 탐색이 쉬워지거나 어려울 수 있음을 알았다. 예를 들면 힙 정렬에 대해 목적값을 최대화하는 문제 인스턴스를 탐색하는 것이 최소화하는 문제 인스턴스 탐색에 비해 쉽다고 추측할 수 있었다.

0/1 배낭 문제에서는 탐욕 알고리즘의 개선 전/후의 문제 인스턴스 공간의 변화를 살펴보았는데, 목적값의 분포를 참고할 때 알고리즘의 개선 여부를 판별할 수 있었다. 한편 기존의 알고리즘을 개선하는 것은 문제 인스턴스 탐색을 어렵게 만들 수 있음을 알았다.

순회판매원 문제의 해 공간은 강한 상관관계를 보이는 것이 알려져 있으나[13], 문제 인스턴스 공간은 약한 상관관계를 보였다. 이는 문제 인스턴스 공간이 해 공간에 관계없이 탐색하기 어려울 수 있음을 보여준다. 한편, 순회판매원 문제에서 GR2opt 알고리즘으로 풀기 어려운 문제 인스턴스인 경우, 다른 문제 인스턴스의 경우에 비해 해 공간에서의 상관관계가 강하게 나타날 수 있음을 보았는데, 이런 경우에는 해 공간에 사용된 유전 알고리즘을 GR2opt와 결합하여 GR2opt를 개선할 수 있을 것으로 본다. 반대로, 해 공간의 상관관계가 약하게 나타나는 경우에는 시간 복잡도가 높은 메타휴리스틱 보다 GR2opt알고리즘을 적용하는 것이 효과적일 것이다.

4절의 탐욕 알고리즘이나 5절의 GR2opt 모두 목적값이 0인 인스턴스가 많은 것으로 보아 대부분의 문제 인스턴스를 잘 푼다는 것을 알 수 있다. 이들 알고리즘을 대상으로 얻은 문제 인스턴스 공간의 상관관계가 약했던 것을 상기하면 유전 알고리즘으로 이들 알고리즘이 풀기 어려운 인스턴스를 탐색하는 것은 쉽지 않을 것으로 보인다.

본 논문에서 얻은 결론을 토대로 향후 연구로서 다음을 제안한다. (1)지역 최적화 알고리즘이 목적값-거리 상관관계에 미치는 영향 조사. (2)문제 인스턴스 공간 분석을 토대로 목적 함수가 그리는 지형 예측. (3)상관관계 분석결과를 토대로 메타휴리스틱 개선. (4)문제 인스턴스 공간을 보다 잘 나타낼 수 있는 다른 분석 방법 탐색.

참 고 문 헌

[1] D. Saab, Y. Saab, and J. Abraham. CRIS: A test cultivation program for sequential VLSI circuits. In *Proceedings of 1992 IEEE/ACM International Conference on Computer Aided Design*, pp. 216-219, 1992.

[2] M. Srinvas and L. Patnaik. A simulation-based test generation scheme using genetic algorithms. In *Proceedings International Conference VLSI Design*, pp. 132-135, 1993.

[3] E. Rudnick, J. Patel, G. Greenstein, and T. Niermann. Sequential circuit test generation in a genetic algorithm framework. In *Proceedings of the 31st Annual Conference on Design Automation (DAC '94)*, pp. 698-704, 1994.

[4] F. Corno, E. Sanchez, M. Sonza Reorda, and G. Squillero. Automatic test program generation - a case study. *IEEE Design & Test*, 21(2):102-109, 2004.

[5] P. McMinn, Search-based software test data generation: a survey, *Software Testing Verification and Reliability*, 14(2):105-156, 2004.

[6] C. Cotta and P. Moscato. A mixed-evolutionary statistical analysis of an algorithm's complexity. *Applied Mathematics Letters*, 16(1):41-47, 2003.

[7] J. Hemert. Evolving combinatorial problem instances that are difficult to solve. *Evolutionary Computation*, 14(4):433-462, 2006.

[8] M. Johnson and A. Kosoresow. Finding worst-case instances of, and lower bounds for, online algorithms using genetic algorithms. *Lecture Notes in Computer Science*, 2557:344-355, 2002.

[9] S.-Y. Jeon and Y.-H. Kim. A genetic approach to analyze algorithm performance based on the worst-case instances. *Journal of Software Engineering and Applications*, 3(8):767-775, 2010.

[10] S.-Y. Jeon and Y.-H. Kim. Finding the best-case instances for analyzing algorithms: comparing with the results of finding the worst-case instance. In *Proceedings of the Korea Computer Congress 2010*, vol. 37, no. 2(C), pp. 145-150, 2010. (in Korean)

[11] 문병로, *쉽게 배우는 유전 알고리즘-진화적 접근법*, 한빛미디어, 2008.

[12] K. D. Boese, A. B. Kahng, and S. Muddu. A new adaptive multi-start technique for combinatorial global optimization. *Operations Research Letters*, 16(2):101 - 113, 1994.

[13] Y.-H. Kim and B.-R. Moon. Investigation of the fitness landscapes in graph bipartitioning: an empirical study. *Journal of Heuristics*, 10(2): 111-133, 2004.

[14] T. Jones and S. Forrest. Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pp. 184-192, 1995.

[15] C. Reeves and T. Yamada. Genetic algorithms, path relinking, and the flowshop sequencing problem. *Evolutionary Computation*, 6(1):45-60, 1998.

[16] T. Stützle and H. Hoos. MAX-MIN Ant System. *Future Generation Computer Systems*, 16(8):889-914, 2000.

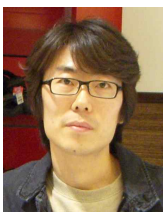
[17] S.-Y. Jeon and Y.-H. Kim. New trials on test

data generation: analysis of test data space and design of improved algorithm, In *Proceedings of the International Conference on Software Engineering Research and Practice*, pp. 352-356, 2011.
[18] M. Main and W. Savitch. *Data Structures and Other Objects Using C++*, 3rd ed., Pearson/ Addison-Wesley, 2004.
[19] R. Sedgewick. *Algorithms in C*, Parts 1-4: Fundamentals, Data Structures, Sorting, Searching, 3rd ed., Addison-Wesley, 1998.
[20] D. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching*, 2nd ed., Addison-Wesley, 1998.
[21] R. Neapolitan, and K. Naimipour. *Foundations of Algorithms Using C++ Pseudocode*, 3rd ed., Jones and Bartlett Publishers, Inc., 2008.

저 자 소 개

전소영(So-Yeong Jeon)
2011년 : 광운대학교 컴퓨터소프트웨어학과 (학사)
2011년～현재 : 한국과학기술원 전산학과 (석사과정)

관심분야 : 휴리스틱, 그래픽스, 소프트웨어공학
Phone : 010-2663-6069
E-mail : presentover@kaist.ac.kr



김용혁(Yong-Hyuk Kim)
1999년 : 서울대학교 전산과학 전공 (학사)
2001년 : 서울대학교 컴퓨터공학부 (석사)
2005년 : 서울대학교 컴퓨터공학부 (박사)

2005년～2007년 : 서울대학교 반도체공동연구소 연구원
2007년～2012년 : 광운대 컴퓨터소프트웨어학과 조교수
2012년～현재 : 광운대 컴퓨터소프트웨어학과 부교수

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