The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 22, No. 2, pp.9-14, Apr. 30, 2022. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2022.22.2.9
JIIBC 2022-2-2

부분집합 합 문제의 일반화된 감산 알고리즘

A Generalized Subtractive Algorithm for Subset Sum Problem

이상운*

Sang-Un Lee*

요 약 본 논문은 부분집합 합 문제의 해를 수행 복잡도 $O(n\log n)$으로 얻는 알고리즘을 제안하였다. SSP는 집합 S의 원소가 초증가수열과 랜덤수열로 구성된 경우로 구분된다. 초증가수열 SSP의 해를 구하는 알고리즘은 수행 복잡도 $O(n\log n)$의 가산 알고리즘 (Additive Algorithm)이 제안되었다. 그러나 랜덤수열 SSP의 해를 구하는 알고리즘은 2^n-1의 가능한 모든 경우수를 확인하는 Brute-Force 방법으로 수행 복잡도는 $O(n2^n)$만이 알려져 있다. 결국, SSP는 NP-완전 (NP-Complete) 문제로 알려져 있다. 본 논문은 초증가수열과 랜덤수열 SSP에 대해 수행 복잡도 $O(n\log n)$으로 해를 구하는 감산 알고리즘 을 제안하였다. 기존 개념은 목표 값 t보다 작은 값으로 구성된 부분집합 S에 대해 부분집합의 합에서 목표값을 뺀 값을 잉여량 (Residual, r)으로 하여 잉여량 보다 작은 값들 중 최대 값을 S에서 제거하는 방법을 적용하였다. 제안된 알고리즘을 다양한 초증가수열과 랜덤수열 SSP에 적용한 결과 S의 원소 개수보다 적은 수행 횟수로 해를 빠르게 얻는데 성공하였다. 결국, 제안된 알고리즘은 SSP의 해를 얻는 일반적인 알고리즘으로 적용할 수 있을 것이다.

Abstract This paper presents a subset sum problem (SSP) algorithm which takes the time complexity of $O(n\log n)$. The SSP can be classified into either super-increasing sequence or random sequence depending on the element of Set S. Additive algorithm that runs in $O(n\log n)$ has already been proposed to and utilized for the super-increasing sequence SSP, but exhaustive Brute-Force method with time complexity of $O(n2^n)$ remains as the only viable algorithm for the random sequence SSP, which is thus considered NP-complete. The proposed subtractive algorithm basically selects a subset S' comprised of values lower than target value t, then sets the subset sum less the target value as the Residual r, only to remove from S' the maximum value among those lower than t. When tested on various super-increasing and random sequence SSPs, the algorithm has obtained optimal solutions running less than the cardinality of S. It can therefore be used as a general algorithm for the SSP.

Key Words : Subset Sum Problem, SSP, Residual, Subtractive Algorithm, Additive Algorithm

*정회원, 강릉원주대학교 과학기술대학 멀티미디어공학과
접수일자 2022년 1월 31일, 수정완료 2022년 3월 7일
게재확정일자 2022년 4월 8일

Received: 31 January, 2022 / Revised: 7 March, 2022 / Accepted: 8 April, 2022
*Corresponding Author: sulee@gwnu.ac.kr
Dept. of Multimedia Eng., Gangneung-Wonju National University, Korea

Ⅰ. 서 론

부분집합 합 문제(subset sum problem, SSP)는 계산 복잡도 이론과 암호학 분야에서 가장 활발히 연구되고 있는 문제들 중 하나이다. SSP는 유한 집합 $S=\{s_1,s_2,\cdots,s_n\}\subset N$과 목표 값 $t\in N$이 주어졌을 때 $S'\subseteq S$의 원소들의 합이 목표 값 t가 되는 S'를 찾는 문제로 식 (1)과 같이 표현된다.[1]

$$SUBSET-SUM=\{<S,t>: there\ exists\ a\ S'\subset S\ such\ that\ t=\Sigma_{s}\in S'\} \quad (1)$$

SSP는 이득(profit, p_j)과 가중치(weights, w_j)가 모두 동일한 물품($j=1,2,\cdots,n$)을 용량 C를 가진 배낭(knapsack)에 넣는 문제와 같은 개념으로, 일반적인 배낭문제(knapsack problem)의 특수한 경우로 다루고 있으며, 식 (2)로 표현된다.[2]

$$maximize \sum_{j=1}^{n} p_j x_j$$
$$\text{such that } \sum_{j=1}^{n} p_j x_j \le C, x_j\in\{0,1\} \quad (2)$$

여기서, x_j는 j번째 물품을 배낭에 넣을 경우 1, 넣지 않을 경우 0이 된다.

SSP는 S의 원소들이 초 증가수열(superincreasing sequence)인 경우와 랜덤수열(random sequence 또는 non-superincreasing sequence)인 경우로 구분될 수 있다. 초 증가수열은 양의 정수(positive integer) 수열 $s_1,s_2,\cdots$의 모든 원소 s_i가 자신보다 이전에 존재하는 모든 원소들의 합 $\sum_{j=1}^{i-1} s_j$보다 큰 경우로 $s_i > \sum_{j=1}^{i-1} s_j$로 표현된다. 초 증가수열에 대한 SSP는 암호학 분야에서 암호키(private key)로, 랜덤수열은 공개키(public key)로 다루고 있으며, 이에 대한 대표적인 알고리즘이 Merkle-Hellman 알고리즘이다.[3-5] 초 증가수열에 대한 부분집합의 합이 목표 값 t가 되는 S는 욕심쟁이 알고리즘(greedy algorithm)으로 $O(n)$ 수행 복잡도로 빠르게 찾는 방법이 제안되어 있다.[6-9] 이 방법을 선택 알고리즘(selective or additive algorithm)이라 하자. 그러나 랜덤수열인 경우는 NP-완전(NP-complete, NPC) 문제로 가능한 모든 조합의 경우 수 2^n-1를 나열하여 찾는 수행 복잡도 $O(2^n)$의 전수탐색 법(brute-force, exhaustive search) 또는 분기한정(branch-and-bound) 법인 정확한 알고리즘(exact algorithm)[8,10,11]이 있다. 만약, 전수탐색 법을 적용할 경우 $n=200$인 경우 1.606×10^{57}회를 수행하여야만 원하는 결과를 얻을 수 있으며 이는 컴퓨터를 활용하더라도 원하는 시간에 문제를 풀 수 없는 결과를 초래한다.

본 논문은 SSP의 초 증가수열과 랜덤수열 모두에 적용할 수 있는 수행 복잡도 $O(n\log n)$의 알고리즘을 제안한다. 만약 주어진 S의 원소들이 초 증가수열 또는 오름차순으로 정렬되어 있는 경우에는 수행 복잡도가 $O(n)$으로 선택 알고리즘과 동일하다. 그러나 랜덤하게 배열되어 있는 경우에는 오름차순 정렬에 $O(n\log n)$의 수행시간이 필요하여 일반적으로 $O(n\log n)$의 수행 복잡도가 요구된다. 결국, 전수탐색법의 수행 복잡도 $O(n2^n)$을 $O(n\log n)$으로 현저히 감소시킨 결과를 얻었다. $n=200$인 경우 1.606×10^{57}회를 460회로 감소시킬 수 있다.

2장에서는 SSP 관련연구와 문제점을 고찰한다. 3장에서는 SSP의 해(solution)를 빠르게 도출하는 알고리즘을 제안한다. 4장에서는 다양한 SSP 데이터들을 대상으로 제안된 알고리즘의 적용성을 검증한다.

Ⅱ. 관련연구와 문제점

NPC 문제는 빠른 해법이 알려져 있지 않다는 것이 가장 주목할 만한 특징으로 문제의 크기가 커질수록 현재 알려진 어떤 알고리즘으로도 문제를 해결하는데 소요되는 시간은 매우 빠르게 증가한다. 그 결과, 제법 큰 문제를 푸는데 소요되는 시간은 현재의 컴퓨터를 활용하더라도 수백 또는 수천 년이 걸린다. 결국, 이들 문제를 빠르게 풀 수 있는지 여부를 결정하는 것이 오늘날 컴퓨터과학 분야에서 주요한 미해결 문제들 중 하나로 남아 있다.[10]

SSP도 NPC 문제들 중 하나로, 해를 얻는 정확한 방법인 전수탐색 법으로 S의 n개의 원소에 대해 가능한 모든 경우를 나열하면 2^n-1개의 경우수가 존재한다. 결국, 2^n-1회를 확인해야 최적 해를 얻을 수 있다.[11] 예로, 다음의 SSP_1[11]에 대해 S'의 가능한 경우 수 $2^5-1=31$을 나열하여 각 부분집합의 합이 t가 되는지 검증해야 해를 찾을 수 있다. 참고로 SSP_1의 해는 존재하지 않는다.

SSP_1 : $S=\{2,7,13,15,17\}$, $t=38$

S' ={{2}=2, {7}=7, {13}=13, {15}=15, {17}=17, {2,7}=9, {2,13}=15, {2,15}=17, {2,17}=19, {7,13}=20, {7,15}=22, {7,17}=24, {13,15}=28, {13,17}=30, {15,17}=32, {2,7,13}=22, {2,7,15}=24, {2,7,17}=26, {2,13,15}=30, {2,13,17}=32, {2,15,17}=34, {7,13,15}=35, {7,13,17}=37, {7,15,17}=39, {13,15,17}=45, {2,7,13,15}=37, {2, 7,13,17}=39, {2,7,15,17}=41, {2,13,15,17}=47, {7,13, 15,17}=52, {2,7,13,15,17}=54}

결국, 정확한 알고리즘의 가장 큰 문제점은 n인 큰 경우, 컴퓨터를 활용하여도 2^n-1의 가능한 경우수를 검증하는데 과다한 시간이 요구되어 암호학 분야에 적용할 수 없다.

만약, S의 n개의 원소가 SSP_2와 같이 초 증가수열로 구성되어 있는 경우에는 가산 또는 선택(additive or selective) 알고리즘으로 n회만 수행하면 쉽게 해를 얻는다. 가산 알고리즘은 다음과 같이 수행된다.[6-9,11]

(1) $s_i > t$인 값은 배낭에 넣을 수 없으므로 제외시킨다.
(2) 남아 있는 $s_i \le t$인 값들에 대해 배낭의 여유분 t 보다 작은 값들 중에서 가장 큰 값을 선택하여 배낭에 넣고, 배낭의 여유분을 계산한다. 이 과정을 배낭의 여유분이 0가 될 때까지 반복적으로 수행한다.

SSP_2[7]에 가산 알고리즘을 수행한 결과는 다음과 같다.

$SSP_2 : S$={2,5,13,21,42,84,168,336,672,1344}
t =889

	S'	$t-\sum_{i=1}^{k} s_i$	추가
초기치	{ø}	889	672
알고리즘 수행	{672}	889-672=217	168
	{672,168}	217-168=49	42
	{672,168,42}	49-42=7	5
	{672,168,42,5}	7-5=2	2
	{672,168,42,5,2}	2-2=0	-

결국, 초 증가수열을 사용하면, 암호화된 메시지를 수신하였을 때, SSP를 적용하여 이 메시지를 쉽게 복호화 시킬 수 있는 장점이 있다. 그러나 가장 큰 문제점은 제3자가 동일한 기법을 적용하여 도청이 가능하다는 것이다.[11,12] 결국, SSP를 암호학 분야에 적용하기 위해서는 초 증가수열이 아닌 랜덤수열 SSP로 개인키를 만들 수 있어야 성공할 수 있다. 따라서 랜덤수열 SSP 문제를 빠르고 간단히 해결하는 알고리즘이 절실히 요구된다.

Ⅲ. SSP의 일반화된 감산 알고리즘

본 장에서는 초 증가수열 뿐만 아니라 랜덤수열의 SSP에 대해서도 해를 빠르게 도출하는 알고리즘을 제안한다.

제안 알고리즘은 Yasinsac[6], Merkel과 Hellman[7]와 Nacin[11]에서 거론된 가산 알고리즘과 반대 개념으로 감산 알고리즘(delete, subtractive, or reverse-selective algorithm)이라 하자. 감산 알고리즘의 기본적인 개념은 초 증가수열에 대해서는 그림 1과 같이 간단히 수행된다. 여기서 $_{\max}S$는 S' 집합 원소들 중 최대치를 가진 원소이다.

Do until $\sum_{i=1}^{k} s_i - t = 0$
  $\sum_{i=1}^{k} s_i - t$ 계산.
  if $\sum_{i=1}^{k} s_i - t = s_i \in S'$ then s_i 삭제, exit
  else if $\sum_{i=1}^{k} s_i - t > {}_{\max}S'$ then ${}_{\max}S'$ 삭제
  else if $\sum_{i=1}^{k} s_i - t < {}_{\max}S'$ then $\sum_{i=1}^{k} s_i - t$ 이하 값들 중 최대치 삭제
Loop

그림 1. SSP 감산 알고리즘의 기본 개념
Fig. 1. Basic Concept of Subtractive SSP Algorithm

SSP_2에 대해 감산 알고리즘을 수행한 결과는 다음과 같이 가산 알고리즘과 동일한 결과를 얻는다.

$SSP_2 : S$={2,5,13,21,42,84,168,336,672,1344},
t =889

S'	$\sum_{i=1}^{k} s_i - t$	삭제
{2,5,13,21,42,84,168,336,672}	454	336
{2,5,13,21,42,84,168,672}	118	84
{2,5,13,21,42,168,672}	34	21
{2,5,13,42,168,672}	13	13
{2,5,42,168,672}	0	-

그러나 랜덤수열에 대해서는 기본적인 감산 알고리즘이 적용될 수도 있고 실패할 수도 있다. 따라서 랜덤수열도 포함하여 일반적으로 적용할 수 있는 SSP 알고리즘을 제안하여야 한다. 이를 위해 본 논문에서는 기본적인 감산 알고리즘을 확장하여 그림 2와 같이 일반적인 SSP 감산 알고리즘을 제안한다. 본 알고리즘은 목표 값 t를 만족하는 부분집합 S'가 반드시 존재한다는 가정에 기반하고 있다.

초기치 : $S=\{s_1, s_2, \cdots, s_n\}$, t
$S=\{s_1, s_2, \cdots, s_n\}$를 오름차순 정렬.

if $\max s > 0.4t$ then $S' = \{s_1, s_2, \cdots, s_k\} \setminus \{s_{k+1}, s_{k+2}, \cdots, s_n\}$
$s.t.\ 1 \le k \le n$, $s_{k+1}, s_{k+2}, \cdots, s_n > t$ 선택.
else if $\max s < 0.4t$ then $r<0$일 때까지 $\max s$부터 내림차순 선택,
$r>0$일 때까지 $\min s$부터 오름차순 선택.
$S \leftarrow S'$, $L=1, H=k$, $s_L = \min s_i \in S$, $s_H = \max s_i \in S$.
$\min s$: S' 원소들 중 최소 값, $\max s$: S' 원소들 중 최대 값.
$r = \sum_{i=1}^{k} s_i - t$, s_d : 삭제 대상 원소, $s_i \in S'$
$\min s_{i+1}$: S 원소들 중 최소 값 $\min s_i$ 다음 값
$\max s_{i+1}$: S 원소들 중 최대 값 $\max s_i$ 다음 값
r_{i-1} : S'에서 r 보다 작은 값들 중 최대 값 (r 바로 이전 값)
r_{i+1} : S'에서 r 보다 큰 값들 중 최소 값 (r 바로 이후 값)

DO Until $r=0$
  if $r = s_i \in S'$ then s_i 삭제, 알고리즘 종료
  else if $(r \notin S') \cap (r > \min s_i)$ then $s_d = r_{i-1}$
    if $r - s_d \ge \min s_i$ then
      if $r - s_d \notin S'$ then s_d 삭제
      else if $r - s_d = s_i \in S'$ then s_d, s_i 삭제
      end if
    else if $r - s_d < \min s_i$ then
      if $s_d = \min s_i$ then s_L, s_{L+2} 삭제, s_{L+1}, $\max s_{i+1}$ 추가
      else if $s_d \neq \min s_i$ then
        if $\min s_i \neq s_L$ then r_{i-1}, s_{i-1} 삭제
        else if $(\min s_i = s_L) \cap (\max s_i \neq s_H)$ then
          if $r > \max s_i$ then $\min s_i$ 삭제
          else if $r < \max s_i$ then $s_d, \max s_i$ 삭제, $\max s_{i+1}$ 추가
          end if
        else if $(\min s_i = s_L) \cap (\max s_i = s_H)$ then $s_d, \max s_{i-1}$ 삭제
        end if
      end if
    end if
  else if $(r \notin S') \cap (r < \min s)$ then $s_d = r_{i-1}$
    if $r - s_d = s_i \in S'$ then s_d, s_i 추가
    else if $r - s_d \notin S'$ then s_d 추가.
    end if
  end if
Loop

그림 2. 일반화된 SSP 감산 알고리즘
Fig. 2. Generalized SSP Subtractive Algorithm

제안된 감산 알고리즘의 수행 복잡도는 S 집합의 원소들을 오름차순 정렬하는데 $O(n \log n)$, t를 만족하는 부분집합 합을 찾는데 $O(n)$으로 결국 $O(n \log n)$의 수행시간으로 수행되며, 초 증가수열과 랜덤수열 모두에서 해를 찾을 수 있는 성능을 갖고 있어 단순하면서도 빠른 장점을 갖고 있다.

Ⅳ. 알고리즘 적용 및 결과 분석

본 장에서는 그림 3의 다양한 SSP를 대상으로 제안된 알고리즘이 해를 빠르게 찾을 수 있는지 검증해본다. 실험에 적용되는 데이터는 SSP_3은 Blair[13], SSP_4와 SSP_9는 Pope[14], SSP_5는 Yasinsac[6], SSP_6은 Fee[15], SSP_7은 Dogar et al.[16], SSP_8은 Umans[17], SSP_{10}은 Bakker[18], SSP_{11}은 James et al.[19], SSP_{12}는 Nacin[11], SSP_{13}은 Moffat[20]에서 인용되었다.

[초증가수열]
SSP_3 : S={1,3,6,14,27,60,150,300,650,1400}, t=836
SSP_4 : S={1,3,5,11,21,44,87,173,349,701}, t=157
SSP_5 : S={3,5,9,18,38,75,155,312}, t=178
SSP_6 : S={76,103,225,464,906,1798,3587,7234,14425, 28916,57748,115573,231098,462180,924405,184 8749}, t=1,679,661
[랜덤수열]
SSP_7 : S={40,45,50,55,80}, t=140
SSP_8 : S={1,7,28,3,2,5,9,32,41,11,8}, t=30
SSP_9 : S={43,129,215,473,903,302,561,1079,697, 1523}, t=3,145
SSP_{10} : S={575,436,1586,1921,5690,721,1183,1570, 1344}, t=6,665
SSP_{11} : S={1.56,1.21,1.02,0.99,0.73,0.65}, t=4±0.05
SSP_{12} : S={181,543,905,1810,3620,7240,3641,7242}, t=10,136
SSP_{13} : S={34,38,39,43,55,66,67,84,85,91,101,117,128, 138,165,168,182,184,186,234,238,241,276,279,2 88,386,387,388,389}, t=1,000

그림 3. SSP 실험 데이터
Fig. 3. Experimental Data for SSP

SSP_3 : S={1,3,6,14,27,60,150,300,650,1400}, t=836

S'	r	s_d	$r-s_d$	삭제	추가
{1,3,6,14,27,60,150,300,650}	375	300	75	300	-
{1,3,6,14,27,60,150,650}	75	60	15	60	-
{1,3,6,14,27,150,650}	15	14	1	14 1	-
{3,6,27,150,650}	0	알고리즘 종료			

SSP_4 : $S=\{1,3,5,11,21,44,87,173,349,701\}$, $t=157$

S'	r	s_d	$r-s_d$	삭제	추가
{1,3,5,11,21,44,87}	15	11	4	11	-
{1,3,5,21,44,87}	4	3	1	3 1	-
{5,21,44,87}	0	알고리즘 종료			

SSP_5 : $S=\{3,5,9,18,38,75,155,312\}$, $t=178$

S'	r	s_d	$r-s_d$	삭제	추가
{3,5,9,18,38,75,155}	125	75	50	75	-
{3,5,9,18,38,155}	50	38	12	38	-
{3,5,9,18,155}	12	9	3	9 3	-
{5,18,155}	0	알고리즘 종료			

SSP_6 : $S=\{76,103,225,464,906,1798,3587,7234,14425,28916,57748,115573,231098,462180,924405,1848749\}$, $t=1,679,661$

S'	r	s_d	$r-s_d$	삭제	추가
{76,103,225,464,906,1798,3587,7234,14425,28916,57748,115573,231098,462180,924405}	169,077	115,573	53,504	115,573	-
{76,103,225,464,906,1798,3587,7234,14425,28916,57748,231098,462180,924405}	53,504	28,916	24,588	28,916	-
{76,103,225,464,906,1798,3587,7234,14425,57748,231098,462180,924405}	24,588	14,425	10,163	14,425	-
{76,103,225,464,906,1798,3587,7234,57748,231098,462180,924405}	10,163	7,234	2,929	7,234	-
{76,103,225,464,906,1798,3587,57748,231098,462180,924405}	2,929	1,798	1,131	1,798	-
{76,103,225,464,906,3587,57748,231098,462180,924405}	1,131	906	225	906 225	-
{76,103,464,3587,57748,231098,462180,924405}	0	알고리즘 종료			

그림 4. 초증가수열 SSP의 해
Fig. 4. The Solution of Superincreasing Sequence SSPs

SSP_7 : $S=\{40,45,50,55,80\}$, $t=140$

S'	r	s_d	$r-s_d$	삭제	추가
{40,45,50,55,80}	130	80	50	80 50	-
{40,45,55}	0	알고리즘 종료			

SSP_8 : $S=\{1,7,28,3,2,5,9,32,41,11,8\}$, $t=30$

S'	r	s_d	$r-s_d$	삭제	추가
{1,2,3,5,7,8,9,11,28}	44	28	16	28	-
{1,2,3,5,7,8,9,11}	16	11	5	11 5	-
{1,2,3,7,8,9}	0	알고리즘 종료			

SSP_9 : $S=\{43,129,215,473,903,302,561,1079,697,1523\}$, $t=3,145$

S'	r	s_d	$r-s_d$	삭제	추가
{43,129,215,302,473,561,697,903,1079,1523}	2,780	1,523	1,257	1,523	-
{43,129,215,302,473,561,697,903,1079}	1,257	1,079	178	1,079	-
{43,129,215,302,473,561,697,903}	178	129	49	129	-
{43,215,302,473,561,697,903}	49	43	6	43 215	129 1,079
{129,302,473,561,697,903,1079}	999	903	96	697	-
{129,302,473,561,903,1079}	302	302	0	302	-
{129,473,561,903,1079}	0	알고리즘 종료			

SSP_{10} : $S=\{575,436,1586,1030,1921,5690,721,1183,1344,1570\}$, $t=6,665$

S'	r	s_d	$r-s_d$	삭제	추가
{436,575,721,1030,1183,1344,1570,1586,1921,5690}	9,391	5,690	3,701	5,690	-
{436,575,721,1030,1183,1344,1570,1586,1921}	3,701	1,921	1,780	1,921	-
{436,575,721,1030,1183,1344,1570,1586}	1,780	1586	194	436	-
{575,721,1030,1183,1344,1570,1586}	1,344	1,344	0	1,344	-
{575,721,1030,1183,1570,1586}	0	알고리즘 종료			

SSP_{11} : $S=\{1.56,1.25,1.21,1.02,0.99,0.73,0.65\}$, $t=4\pm0.05$

S'	r	s_d	$r-s_d$	삭제	추가
{0.65,0.73,0.99,1.02,1.21,1.25,1.56}	2.76	1.56	1.20	1.56	-
{0.65,0.73,0.99,1.02,1.21,1.25}	1.20	1.21	-0.01	1.21	-
{0.65,0.73,0.99,1.02,1.25}	-0.01	알고리즘 종료			

SSP_{12} : $S=\{181,543,905,1810,3620,7240,3641,7242\}$, $t=10,136$

S'	r	s_d	$r-s_d$	삭제	추가
{181,543,905,1810,3620,3641,7240,7242}	15,046	7,242	7,804	7,242	-
{181,543,905,1810,3620,3641,7240}	7,804	7,240	564	7,240	-
{181,543,905,1810,3620,3641}	564	543	21	543 3,641	7,240
{181,905,1810,3620,7240}	3,620	3,620	0	3,620	-
{181,905,1810,7240}	0	알고리즘 종료			

SSP_{13} : $S=\{34,38,39,43,55,66,67,84,85,91,101,117,128,138,165,168,182,184,186,234,238,241,276,279,288,386,387,388,389\}$, $t=1,000$

$S'=\{389,388,34,38,39,43,55,66\}$	r	s_d	$r-s_d$	삭제	추가
{34,38,39,43,55,66,388,389}	52	43	9	43 388	-
{34,38,39,55,66,389}	-379	288	-91	-	288 91
{34,38,39,55,66,91,288,389}	0	알고리즘 종료			

그림 5. 랜덤 수열 SSP의 해
Fig. 5. The Solution of Random Sequence SSPs

제안된 감산 알고리즘은 표 1과 같이 원소 개수가 5 ~ 29개인 집합 S에 대해 감산을 최소 1회, 최대 6회 수행으로 해 S'를 얻는데 성공하였다. 결국, SSP에 대해 감산 알고리즘을 수행하여 해를 구하면 잔수탐색 법보다 수행 횟수를 평균 98.85% 감소시키는 효과를 얻을 수 있다.

표 1 SSP의 알고리즘 수행 횟수 비교
Table 1. The Number of Trials for SSP Algorithm

SSP	n		수행횟수		
	S	S'	Brute-Force 알고리즘	감산 알고리즘	Brute-Force 대비 비율
SSP_2	10	9	**511**	**4**	**0.78%**
SSP_3	10	9	**511**	**3**	**0.59%**
SSP_4	10	7	**127**	**2**	**1.57%**
SSP_5	8	7	**127**	**3**	**2.36%**
SSP_6	16	15	**32,767**	**6**	**0.02%**
SSP_7	5	5	**31**	**1**	**3.23%**
SSP_8	11	9	**511**	**2**	**0.39%**
SSP_9	10	10	**1,023**	**6**	**0.59%**
SSP_{10}	10	10	**1,023**	**4**	**0.39%**
SSP_{11}	7	7	**127**	**2**	**1.57%**
SSP_{12}	8	8	**255**	**4**	**1.57%**
SSP_{13}	29	8	**255**	**2**	**0.78%**
평균					**1.15%**

V. 결 론

본 논문은 초 증가수열과 랜덤수열 등 주어진 집합 S의 원소 수열에 상관없이 S의 부분집합 S'의 합이 목표값 t를 만족하는 S'를 찾는 알고리즘을 제안하였다. 제안된 알고리즘은 목표 값 t가 허용오차(tolerance)를 가진 경우에도 일반적으로 적용할 수 있는 장점을 갖고 있다.

제안된 알고리즘은 부분집합의 합에서 목표 값을 뺀 잉여량 $r = \sum_{i=1}^{k} s_i - t$값을 기준으로 하여 S'에서 원하는 값을 빼거나 더하는 과정을 거쳐 단순히 해를 구하였다. 제안된 알고리즘은 S'의 원소 개수보다 작은 횟수로 해를 빠르게 구하는 장점을 갖고 있다.

본 논문은 단지 수행 복잡도 $O(n \log n)$으로 SSP의 해를 빠르게 도출할 수 있는 알고리즘으로 증명되었다. 결국, 본 알고리즘은 SSP의 해를 도출하는 일반적인 알고리즘으로 적용할 수 있을 것이다.

References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to Algorithms," McGrew-Hill Book Company, 2005.

[2] Wikipedia, "List of Knapsack Problems," http://en.wikipedia.org/wiki/List_of_knapsack_problems, Wikimedia Foundation Inc., 2012.

[3] B. Ho, "Further Exploration in Public Key Encryption," The MIT Undergraduate Journal of Mathematics, 2000.

[4] K. Scott, "CS 413, Computer and Data Security: The Knapsack Problem and Merkle Hellman Encryption,", 2007.

[5] J. C. Gao and E. R. Zhou, "Introduction to Knapsack Cipher," http://zhouer.org/courses/2003-IIS/slide.pdf, 2003.

[6] A. Yasinsac, "Cryptography: Public Key Algorithm Cryptography based on the Knapsack Problem," University of South Alabama, 2008.

[7] R. C. Merkel and M. E. Hellman, "Hiding Information and Signatures in Trapdoor Knapsacks," IEEE Trans. on Information Theory, Vol. 24, No. 5, pp. 523-530, Sep. 1978, https://doi.org/10.1109/ TIT.1978.1055927

[8] G. J. Woeginger, "CO2a: Combinatorial Optimization - Open Problems Around Exact Algorithms," Department of Mathematics and Computer Science, TU Eindhoven, 2008.

[9] C. Christensen, "Cryptology: The Knapsack Problem," Department of Mathematics Northern Kentucky University, 2005.

[10] Wikipedia, "NP-Complete," http://en.wikipedia.org/wiki/NP-Complete, Wikimedia Foundation Inc., Retrieved Jan. 2022.

[11] D. Nacin, "The Subset-Sum Problem," Department of Mathematics, William Paterson University, 2008.

[12] S. K. Chong, G. Farr, L. Frost, and S. Hawley, "On Pedagogically Sound Examples in Public-key Cryptography," Conferences in Research and Practice in Information Technology, Vol. 48, No. 1, pp. 63-68, Australian Computer Society Inc., Jan. 2006.

[13] C. Blair, "Notes on Cryptography," Business Administration Dept., University of Illinois, 1994.

[14] B. Pope. "Cryptography Again: Algorithmic Problem Solving," CSSE, The University of Melbourne, 2008.

[15] G. Fee, "Math 342: Elementary Number Theory," Department of Mathematics and Statistics, SFU, 2000.

[16] F. R. Dogar, L. Aslam, Z. A. Uzmi, S. Abbasi, and Y. C. Kim, "Connection Preemption in Multi-Class Networks," IEEE Globecom, 2006.

[17] C. Umans, "CS21: Decidability and Tractability," Computer Science, California Institute of Technology, 2008.

[18] J. Bakker, "The Knapsack Problem and The LLL Algorithm," http://www.math.ucsd.edu/~crypto/Projects/JenniferBakker/Math187/index.html, 2004.

[19] R. J. W. James and R. H. Storer, "Techniques for Solving Subset Sum Problems within a Given Tolerance," International Trans. in Operational Research, Vol. 12, No. 4, pp. 437-453, Jul. 2005, https://doi.org/10.1111/j.1475-3995.2005.00517.x

[20] A. Moffat, "Example for the book "Programming, Problem Solving, and Abstraction with C," Pearson Sprint Print, Sydney, Australia, 2003.

저 자 소 개

이 상 운(정회원)

- 1987년 : 한국항공대학교 항공전자공학과 (학사)
- 1997년 : 경상대학교 컴퓨터과학과 (석사)
- 2001년 : 경상대학교 컴퓨터과학과 (박사)
- 2003년 : 강원도립대학 컴퓨터응용과 전임강사
- 2004년 ~ 2007년 2월 : 국립 원주대학 여성교양과 조교수
- 2007년 3월 ~ 2015년 3월 : 강릉원주대학교 멀티미디어공학과 부교수
- 2015년 4월 ~ 현재 : 강릉원주대학교 멀티미디어공학과 정교수
- 관심분야 : 소프트웨어 프로젝트 관리, 개발 방법론, 분석과 설계 방법론, 시험 및 품질보증, 소프트웨어 신뢰성, 인공지능과 빅데이터분석, 최적화 알고리즘
- e-mail : sulee@gwnu.ac.kr