스쿨존 위험도를 고려한 최단시간 네비게이션 경로설정(구로구 중심으로)

Fastest Path Guidance Navigation Considering School Zone Risk (Around Guro-gu)

2 조

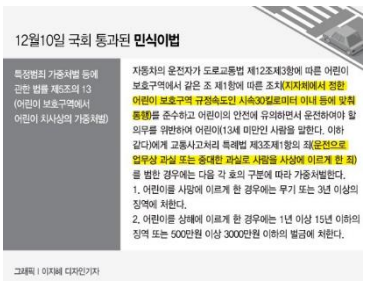
2019147019 강세정
2018112056 김민아
2019147025 신재욱
2018147026 이성민
2018199075 이수연
2019147007 장현우

목 차

Ⅰ. 서론

1. 주제 선정 이유

2019 년 9 월 11 일 충남 아산 온양중학교 정문 앞 사거리 인근 횡단보도에서 초등학교 2 학년 김민식(9)군이 교차로를 막 지난 차량에 치여 목숨을 잃는 사고가 발생했다. 당시 사고가 발생했던 지점은 어린이 보호구역(스쿨존)이었음에도 일정 간격으로 만들어진 과속방지턱을 제외하고는 신호등, 과속 단속 카메라 등과 같은 어린이 교통안전을 위한 최소한의 장치도 존재하지 않았다. 이후 이 사건을 계기로 스쿨존 내 교통안전에 대한 문제제기가 이루어졌고, 2019 년 10 월 13 일 강훈식 의원과 이명수 의원은 '민식이 법'을 발의했다. '민식이 법'(도로교통법, 특정범죄 가중처벌 등에 관한 법률 일부개정 법률안)[1]은 스쿨존에 과속 단속 카메라와 신호등 설치를 의무화하는 내용을 포함하고 있으며, 속도제한과 횡단보도에 대한 안전표지, 과속방지시설, 미끄럼방지 시설 또한 설치하도록 명시하고 있다. 스쿨존 내에서 제한 속도를 지키지 않거나 보호 의무를 위반해 어린이를 사망케 하거나 다치게 하면 가중 처벌이 되며, 어린이가 사망하면 무리 또는 3 년 이상 징역, 상해를 입히면 1 년 이상 15 년 이하의 징역 또는 500~3000 만원의 벌금형에 처하게 된다. 그러나 이 법이 논란이 되기 시작한 것은 "어린이의 안전에 유의하면서 운전하여야 할 의무를 위반하여"라는 문구 때문이다. 이 문구는 너무 모호하고 주관적이기에 개인에 따라 해석의 차이가 존재하고 무과실 판정의 기준이 별도로 존재하지 않기 때문에 실질적으로 운전자의 무죄 입증이 불가능하다. 이 때문에 스쿨존 내에서 사고가 난다면 제한 속도를 준수하였음에도 불구하고 어린이 보행자가 상해를 입을 경우, 1 년 이상의 징역 혹은 500 만원 이상의 벌금이 적용되며 사망시에는 최고 무기징역까지 적용이 가능하다.



[그림 1] 민식이 법 조문

2020 년 3 월 25 일, 민식이 법이 시행되기 시작하였으나 이후 법을 악용하여 각종 보험금을 취하려는 등 부작용이 잇따라 발생하였고, 실제로 얼마 전 한 온라인 커뮤니티에는 스쿨존에서 운전 중 초등학교 5 학년생이 차로 갑자기 뛰어들어 차 뒷문에 부딪혀서 검사를 받았는데, 그 부모가 민식이 법을 거론하며 3 백만원의 합의금과 병원비 전액을 요구한 사례가 있었다.[2] 이로 인해 운전자들의 '스쿨존 회피 현상'이 지속적으로 나타나고 있으며, 이런 여론을 인식한 듯 각종 관련 운전자 보험들이 출시되고 있다.[3]

---

[1] 특정범죄 가중처벌에 관한 법률 제5조의 13(어린이 보호구역에서 어린이 치사상의 가중처벌)

[2] 장영준, "[와글와글 커뮤니티] 민식이법 이용해 협박?... "신고 안 하겠다" 합의금 요구", 경기일보, 2020.04.29

[3] KB손해보험, 삼성화재, 현대해상 등 주요 손해보험사들은 스쿨존 벌금 지원금을 2000만원에서 3000만원으로 상향했다. 이외에도 삼성화재의 경우 스쿨존에 한해 전치 6주 미만 사고에도 교통사고 처리지원금을 500만원 한도로 보장토록

우리는 이러한 현상을 통해 현 사회의 요구사항을 고민해보게 되었다. 스쿨존인 만큼 어린이의 안전도 보장함과 동시에 운전자의 불편함을 동시에 해소할 수 있는 방법은 스쿨존의 운전자 위험도를 적절히 반영한 네비게이션이라는 생각이 들었다. 실제로 현재 스쿨존을 거치지 않고 경로를 안내하는 네비게이션이 출시되어 다운로드 시행 수가 급격하게 증가한 상태로, 스쿨존 회피 경로는 많은 운전자들의 호응을 얻고 있다는 것을 알 수 있다.[4]

그러나 현재의 네비게이션은 경로 안내 시 스쿨존을 완전히 차단하여 안내하는 방식으로, 자동차 운행 시간이 급격히 증가하는 부작용이 존재한다. 그렇기 때문에 우리는 스쿨존 내 어린이 사고와 관련된 요인들을 찾아 자동차 운행시 '스쿨존' 사고 위험도 가중치를 계산하여 운전자의 목적지까지 최소시간으로 경로를 안내하는 모델을 제시하고자 한다.

2. 표본 지역 선정 기준

본 연구에서는 인구가 가장 많고, 도로가 복잡한 서울시를 중심으로 알고리즘을 구성하고자 한다. 본격적인 연구에 앞서 스쿨존 내 어린이 교통사고와 관련한 통계 자료들을 살펴본 결과 유의미한 결과를 도출하기 위해 스쿨존 내 어린이 교통사고 건수가 비교적 많은 지역들(2017-2019 기준)[5]을 선정하기로 결정한 후 후보군을 네 군데로 추렸다.

그 중에서도 스쿨존 내 어린이 교통사고 감소를 위해 지자체 적인 노력을 기울인 것으로 보이는 지역을 선정하기로 하였다. 현재 구로구는 민식이 법을 바탕으로 '어린이 교통사고 예방 종합대책'을 수립하였으며, 이에 따라 어린이 보호구역 내 과속단속카메라 증설, 태양광 LED 안전표지판 설치, 등하굣길 보행신호기 신설, 노후 및 파손된 안전펜스 교체, 횡단보도 앞 대기공간 확장, 그리고 보호구역 연계 보행안전시설물 정비 등의 신규 사업을 펼친다. 더 나아가 스쿨존의 시인성을 높이기 위해 '노란신호등'과 보행자를 쉽게 인식할 수 있게 하는 '횡단보도 집중조명' 설치도 새로 실시하기로 하였으며, 구로형 어린이 보행특화거리인 '아마존' 조성, 옐로카펫, 노란발자국 설치 등의 사업도 지속적으로 추진한다는 계획을 세웠다. 이러한 데이터를 바탕으로 우리는 본 연구의 표본 지역을 구로구로 설정하였다.[6]

시도	강서구	구로구	강남구	은평구	중랑구	성북구	종로구	서초구	노원구	양천구	강동구	동작구
사고건수	10	9	6	6	4	4	2	2	2	2	1	1

[표 1] 각 행정구역 별(서울시 기준) 스쿨존 내 어린이 교통사고 건수

운전자보험 담보를 변경하고 가입자 모두에게 소급 적용을 해주기로 했다.

[4] 신화섭, "아틀란 내비, '스쿨존 회비' 업데이트 후 다운로드 6배 급증", 모터그래프, 2020.04.01

[5] "스쿨존내 어린이사고 다발지역", TAAS 교통사고분석시스템, http://taas.koroad.or.kr/gis/mcm/mcl/initMap.do?menuId=GIS_GMP_ABS#

[6] 임규수, "구로구, 어린이 교통사고 예방 위해 발 벗고 나섰다", TGN (땡큐 굿 뉴스), 2020.01.29

본 연구는 시간적 한계 때문에 구로구로 한정하여 진행했지만, 이후 이미 제시되어 있는 표준 노드와 링크의 데이터를 활용하여 보다 넓은 지역으로 확장해 적용하는 가능성을 염두에 두고 프로젝트를 진행하고자 하였다.

3. 목표 및 모델의 핵심 기능 소개

기존의 네비게이션은 최단시간만을 고려하거나, 스쿨존을 반영한 경우 완전히 피해가는 경로를 제시하고 있다. 이에 본 연구는 사용자 설정에 따른 스쿨존 위험도 및 최단시간 가중치를 고려하여 경로를 안내하는 네비게이션의 알고리즘을 구현하고자 한다.

본 연구의 목표는:

첫째, 사용자가 원하는 기준에 따라 스쿨존 위험 가중치의 크기를 다르게 설정할 수 있게 하는 것이다. 이때 각 가중치 설정은 총 4 단계로 이루어지며, 가중치 설정에 따라 기존의 네비게이션과 동일하게 최소시간만을 목적으로 하는 방식도 설정 가능하다.

둘째, Dijkstra 알고리즘을 이용하여 사용자만의 목적함수를 최소화하는 경로를 안내하는 것이다. 사용자에게 경로를 안내할 때는 지도에 경로를 선으로, 스쿨존을 원의 형태로 표시해 추천 경로가 스쿨존을 피해감을 시각적으로 제시한다.

Ⅱ. 본론

1. 문제 정의 및 모델 개요

우리는 구로구 내 스쿨존 교통사고 위험도를 고려한 최소시간 네비게이션 경로설정 방법을 탐색한다. 우선 구로구 내에서 경로구간을 정의한다. 구로구의 주요 도로들을 노드, 링크로 정리해 Dijkstra's algorithm 을 적용할 네트워크로 구현한다. 다음으로 구간별 스쿨존 위험도 가중치 설정을 위한 parameter 데이터를 수집하고 적용한다. 스쿨존 위험도 가중치에는 구간별 스쿨존 개수, 시간대별 스쿨존 사고 빈도[7], 차량 교통량, 이 세 가지 요인의 데이터를 반영한다. 이처럼 기본 가중치를 구간별로 설정한 후, 사용자 선호도에 따라 앞의 세 요인 중 더 중요시하고자 하는 요인이 있을 경우 사용자 맞춤 설정 기능을 통해 위험가중치 수준에 반영한다. 이러한 방식으로 최종 스쿨존 위험도 가중치를 도출, 네트워크의 링크인 각 구간의 weight 로 입력한다. 그래프 상의 두 노드의 최단거리를 구하는 문제이므로 Dijkstra's algorithm 을 적용, 최소시간 이동경로를 구한다.

2. 문제 정의 및 모델 개요

a. Parameter Data 설정

Parameter:

1) T_{ijt}: 시간 구역이 t일 때 node i에서 j까지의 소요 시간
2) a_{ij}: arc (i,j)에서 스쿨존이 차지하는 비율
3) b_{ijt}: arc (i,j)의 시간대별 교통량
   (1/(시간대=t 일 때, node i에서 j까지의 주행속도))
4) c_t: 시간별, 요일별 교통사고량에 따른 사고 위험도
5) x,y,z: 사용자의 선호도 ($x=\{0,1,2,3\}, y=\{0,1,2,3\}, z=\{0,1,2,3\}$)
   x: 링크의 높은 스쿨존 포함률 기피 정도
   y: 높은 통행량 기피 정도
   z: 시간 구역에 따른 높은 스쿨존 사고율 기피 정도
6) t: 가중치 중 시간에 따라 변하는 값들을 그룹화하기 위해 도입한, '평일/주말, 일별 4 개의 시간대'로 나뉜 8 개의 시간 구역

b. Parameter Data 적용

최소시간 경로만 제시하는 기존의 네비게이션과 달리, 스쿨존에서의 사고위험도를 추가로 고려한 경로를 제시하기 위해 목적함수에 위험 가중치를 반영하였다. 경로에서 스쿨존의 비율(a_{ij})이 적을수록, 교통량(b_{ijt})이 적을수록, 시간별 요인별 사고위험도(c_t)가 적을수록 목적함수의 값에 유리하다. 이때, 각 요인들의 범위를 통일하는 작업을 거쳐 a,b,c는 0 에서 1 의 값을 가진다.

사용자는 본인의 선호도에 따라 각 위험가중치가 반영되는 정도를 조절할 수 있다. a,b,c 세 가지 요인 각각에 대한 반영비율을 x,y,z 요인을 자체 설정하는 방식으로 조절할 수 있으며, 모든 요인을 최대로 고려하면 위험도 측면에서 가장 보수적인 경로가 탐색되고 모든 요인을 고려하지 않으면 기존의 네비게이션처럼 최소시간 경로를 제시한다.

---

[7] 홍상연, 정재훈, 2018, 「도로교통사고 예방 위한 위험도 평가기법 모색」, 서울연구원, p.16

요일별, 시간대별로(t) 그 값이 결정되는 parameter 들이 있다. 해당 parameter 들은 시간대에 따라 유의미하게 차이나는 스쿨존 어린이 교통사고량을 기반으로, ANOVA table 과 Tukey's test 를 이용해 통계학적으로 그룹화하였다. (시간 구역 구분과 그 과정은 첨부파일 1 참고)

c. LP Model Formulation

1) Objective Function:

$minimize\ z = \sum_i \sum_j [T_{ij}(1 + a_{ij})^x * (1 + b_{ij})^y * (1 + c_t)^z] * X_{ij}$(for given t)

2)Decision variables:

X_{ij}: 노드 i에서 j로의 링크 선택 여부 판별 boolean var(0 or 1)

3) Constraints:

(flow-conservation constraint)

$\sum_{i=1} \alpha_{ij} X_{ij} = \sum_{k=1} \alpha_{jk} X_{jk}$ for $\forall j - \{start, end\}$

$1 + \sum_{i=1} \alpha_{i,start} X_{i,start} = \sum_{k=1} \alpha_{start,k} X_{start,k}$

$\sum_{i=1} \alpha_{i,end} X_{i,end} = \sum_{k=1} \alpha_{end,k} X_{end,k} + 1$

k_{ij}: set covering matrix $[\alpha]_{i \times j}$

3. Data 수집 과정

a. 구로구 내 표준 노드, 링크[8] (첨부파일 2 참고)

: '지능형교통체계 표준노드링크관리시스템'에서 제공하는 우리나라의 표준 노드, 링크 데이터를 적용하였다. 데이터가 SHP 파일 (위치 정보를 제공하는 파일)에서 TM 좌표를 제공하는 형식이어서 추후 구현 과정에서의 사용에 용이하도록 위도, 경도를 바꾸어 주었다.

노드 속성 정보에 활용하는 코드 값은 아래와 같다.

영문명	한글명	코드값	노드유형 설명	노드설정위치
NODE_TYPE	노드유형	101	교차로	교차로
		102	도로시.종점	도로시.종점
		103	속성변화점	교통통제점, 도로운영변환점, 기타
		104	도로시설물	도로구조변환점, 교통진출입점
		105	행정경계	행정구역변환점
		106	연결로접속부	진출입로 또는 IC 및 JC 연결로의 접속지점
		108	IC 및 JC	IC 및 JC (하나의 노드로 통합 시)
TURN_P	회전제한유무	0 1	무 유	-

[표 2] 노드 속성 정보 코드

[8] "지능형교통체계 구축기준", iTS 표준노드링크, http://nodelink.its.go.kr/intro/intro06.aspx

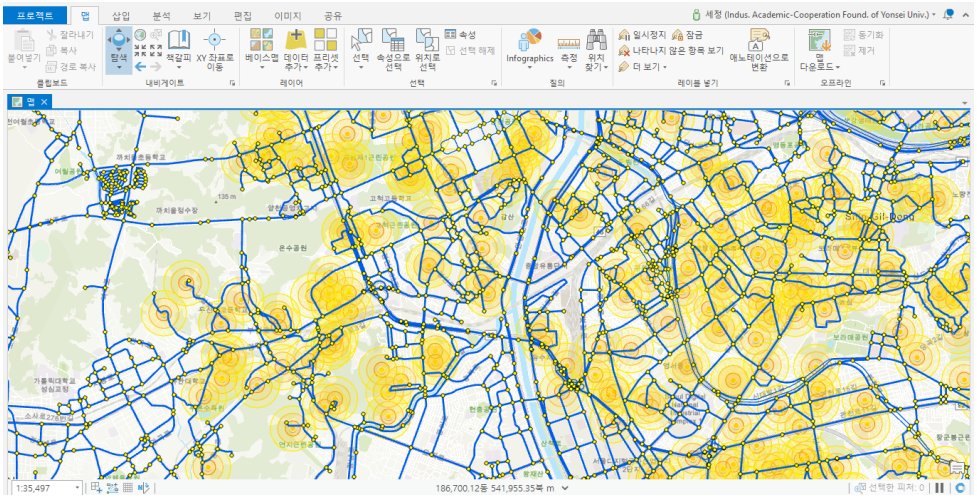
링크 속성 정보에 활용하는 코드 값은 아래와 같다.

영문명	한글명	코드	코드정보
ROAD_RANK	도로등급	101 102 103 104 105 106 107 108	고속국도 도시고속국도 일반국도 특별.광역시도 국가지원지방도 지방도 시.군도 기타
ROAD_TYPE	도로유형	000 001 002 003 004	일반도로 고가차도 지하차도 교량 터널
ROAD_USE	도로사용여부	0 1	사용 미사용
MULTI_LINK	중용구간여부	0 1	독립구간 중용구간
CONNECT	연결로코드	000 101 102 103 104 105 106 107 108	연결로 아님 고속국도 연결로 도시고속국도 연결로 일반국도 연결로 특별.광역시도 연결로 국가지원지방도 연결로 지방도 연결로 시.군도 연결로 기타도로 연결로
REST_VEH	통행제한차량	0 1 2 3 4 5 6	모두통행가능 승용차 승합차 버스 트럭 이륜차 기타

[표 3] 링크 속성 정보 코드

b. 어린이보호구역 데이터 (첨부파일 2 참고)

: '공공데이터포털'에서 제공하는 전국 어린이 보호구역 표준데이터를 사용하였다. 아래의 [그림 2]에서는 구로구에 위치한 어린이보호구역을 100m, 200m, 300m 간격의 노란색 원형태로 그려 놓았다.



[그림 2] ArcGIS 로 구현한 구로구 표준 노드, 링크, 어린이보호구역

c. 시간별, 요일별 스쿨존 어린이 교통사고 데이터

: 'TAAS 교통사고분석시스템'에서 제공하는 어린이 교통사고 데이터를 사용하였다. 데이터는 2017 년부터 2019 년까지 3 년을 기준으로 하였고 이 자료를 활용하여 c_t 를 계산하였다.

시간대별 교통사고								
시간	2017년 사고건수	2018년	2019년	합계	평균	c_t1	1+c_t1	
0시~2시	0	0	1	1	0.333333333	0.04547082	1.045471	구간1
2시~4시	0	0	2	2	0.666666667			
4시~6시	1	0	2	3	1			
6시~8시	1	3	5	9	3			
8시~10시	54	52	62	168	56	0.523809524	1.419799	구간2
10시~12시	26	23	21	70	23.33333333			
12시~14시	66	47	56	169	56.33333333			
14시~16시	124	119	157	400	133.3333333	1	2	구간3
16시~18시	133	120	147	400	133.3333333			
18시~20시	58	59	93	210	70	0.419799499	1.52381	구간4
20시~22시	14	11	18	43	14.33333333			
22시~24시	2	1	3	6	2			

[표 4]시간대별 교통 사고량

요일별 (사고건수)								
요일	2017년	2018년	2019년	합계	평균	c_t2	1+c_t2	
월	87	69	88	244	81.33333	0.8660377	1.866038	평일
화	73	87	99	259	86.33333			
수	94	71	91	256	85.33333			
목	82	68	93	243	81			
금	77	87	122	286	95.33333			
토	39	31	49	119	39.66667	0.1061321	1.106132	주말
일	27	22	25	74	24.66667			

[표 5]요일별 교통 사고량

d. 링크별 이동 시 소요시간, 주행속도

: 이동 과정에서의 소요시간, 주행속도의 경우에는 네이버맵 길찾기(첨부파일 3 참고)를 사용하였다. 이 자료를 사용하여 T_{ijt}, b_{ijt}를 계산하였다.

b_{ijt}는 통행량에 대한 가중치로 교통량의 직접적 데이터 수집이 어려워 이는 통행 속도를 이용하여 추정을 하였다. 즉, "1/속도"를 교통량을 추정하는 데에 사용하였다. 교통량을 이 같은 방식으로 추정함은 '백재진(2008), 교통량 예측 기반 최적경로 탐색 엔진의 개발 및 구현'에서와 같은 방식이며, 토연구원의 '교통사고에 안전한 국토 구현'을 주제로 한 국토연 2014-22 보고서[9] 에 따르면 '운전시 속도를 올려야겠다고 생각하는 경우는 교통량이 감소할 때가 가장 많았으며, 마찬가지로 속도를 내려야겠다고 생각하는 경우도 교통량이 증가할 때가 가장 많았다.' 라는 것을 그 근거로 하였다.

[9] 김준기, 「교통사고에 안전한 국토 구현」, 경기도:국토연구원, 2014

또한 높은 교통량이 스쿨존의 교통사고로 이어진다고 생각하여 교통량에 대한 가중치를 계수를 추가한 것은 경기도교통정보 센터의 '주요 사회경제지표와 교통지표간 상관관계 분석'을 주제로 한 보고서[10]에 근거를 두고 있다. 이에 따르면 '세부적인 증감율을 분석한 결과, 교통사고 감소폭이 가장 큰 의정부시, 과천시의 경우 교통량의 증가도 비교적 낮은 수준인 것으로 분석되었으며, 가장 높은 사고 발생건수 증가를 보인 안양시의 경우 교통량의 증가폭도 가장 큰 것으로 분석되었다.'라고 되어있다.

```
[1]: import requests
     import json

[2]: header = {
         'user-agent': 'Mozilla/5.0 (iPhone; CPU iPhone OS 11_0 like Mac OS X) AppleWebKit/604.1.38 (KHTML, like Gecko) Version/11.0 Mobile/15A372 Safa
         'x-requested-with': 'XMLHttpRequest'
     }

[3]: s = requests.Session()
     s.headers.update(header)

[4]: def SEARCH_DISTANCE_URL(start, end):
         return 'https://m.map.naver.com/spirra/findCarRoute.nhn?route=route3&output=json&result=web3&coord_type=latlng&search=2&car=0&mileage=12.4' +

[5]: def GET_INFO(start, end):
         res = s.get(SEARCH_DISTANCE_URL(start, end)).text
         res_dict = json.loads(res)

         target = res_dict['routes'][0]['summary']
         distance = target['distance']
         sec = target['duration']
         taxi_fare = target['taxi_fare']
         #print('검색결과')
         #print('출발지: {}, 도착지: {}'.format(start, end))
         #print('추천경로 이동 거리: {}km'.format(distance/1000))
         #print('예상 소요시간: {}분'.format(sec/60))
         #print('예상 택시요금: {}원'.format(taxi_fare))
         print('출발지: {}, 도착지: {}, 추천경로 이동 거리: {}km'.format(start, end, distance/1000))
         print('예상 소요시간: {}초, 속도: {}m/s'.format(sec, distance/sec))
         return sec, distance/sec

     import time
     import random

     import datetime

     time_lst = []
     speed_lst = []

     c = datetime.datetime.now()
     print(c)
     for key in link_dict.keys():
         while True:
             try:
                 x1, y1, x2, y2, d = link_dict[key]
                 start = str(y1) + "," + str(x1)
                 end = str(y2) + "," + str(x2)
                 t, speed = GET_INFO(start,end)
                 time_lst.append(t)
                 speed_lst.append(speed)
                 time.sleep(1+random.random()*2)
                 break
             except:
                 print(datetime.datetime.now())
                 time.sleep(60)
     print(c)
     print(datetime.datetime.now())

2020-06-13 00:05:34.384856
출발지: 126.86249550000001,37.50773692, 도착지: 126.86005520000002,37.50677069, 추천경로 이동 거리: 0.239km
예상 소요시간: 52초, 속도: 4.596153846153846m/s
출발지: 126.86005520000002,37.50677069, 도착지: 126.86249550000001,37.50773692, 추천경로 이동 거리: 0.239km
예상 소요시간: 40초, 속도: 5.975m/s
출발지: 126.86444350000001,37.50867362, 도착지: 126.86249550000001,37.50773692, 추천경로 이동 거리: 0.201km
예상 소요시간: 36초, 속도: 5.583333333333333m/s
출발지: 126.86249550000001,37.50773692, 도착지: 126.86444350000001,37.50867362, 추천경로 이동 거리: 0.201km
```

[그림 3] 각 링크 별 주행시간 데이터 수집 코드

각 링크의 거리와 예상 소요 시간, 속도 데이터를 추출하였다. 모든 도로를 세세하게 고려할 수는 없었기 때문에, 대신 링크를 구성하는 두 노드의 위도와 경도 값을 네이버 길찾기를 API 로 크롤링하여 나온 결과값을 필요한 데이터의 근사값으로 사용하였다. 앞서 시간 t 를 평일/주말 각각 4 개씩 총 8 개의 그룹으로 만들어 주었기 때문에 거리,

[10] (No.2015-02)주요 사회경제지표와 교통지표간 상관관계, 경기도 교통DB센터, 2015

시간 데이터 또한 8 번에 걸쳐서 적절한 시간에 코드를 실행시키는 방법으로 데이터를 수집하였다.[11]

e. 링크별 스쿨존 포함률 데이터

: '어린이노인장애인보호구역 통합지침'에 따르면, 보호구역이 겹칠 경우 하나로 통합해서 관리한다. 따라서 각 링크가 몇 개의 보호구역을 지나가는지가 아닌, 각 링크 중 보호구역에 포함되는 비율로 가중치를 고려하는 것이 적합하다고 판단했다.

어린이 보호구역의 비율을 구하기 위해서, 각 링크에서 출발 노드와 도착 노드의 위도와 경도 값을 구한 다음 100 등분하였다. 이 100 개의 점 각각에 대해 구 위에서의 거리를 구하는 공식인 haversine 공식을 사용하여 어린이 보호구역이 있는지를 계산하였다. (주석)

만약 어린이 보호구역의 범위인 300m 이내에 들어오는 경우가 있다면 그 점은 보호구역에 속하는 것으로 판단하였고, 총 100 개의 점 중에서 어린이 보호구역에 속하는 점의 개수를 구해서 0~1 사이의 비중으로 환산하였다. 이 값을 parameter a_{ij} 로 사용하였다. (첨부파일 4 참고)

```
child = pd.read_csv("dataset/보호구역.csv", encoding = "cp949")
```

```
from math import radians, cos, sin, asin, sqrt
# 위도: lat, 경도: lon
def haversine(lon1, lat1, lon2, lat2):
    """
    Calculate the great circle distance between two points
    on the earth (specified in decimal degrees)
    """
    # convert decimal degrees to radians
    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])
    # haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    # Radius of earth in kilometers is 6371
    km = 6371 * c
    return km * 1000
```

```
child_data = [(child["위도"][i],child["경도"][i]) for i in range(len(child["위도"]))]
```

```
freq = []
d = []
for key in link_dict.keys():
    x1, y1, x2, y2, distance = link_dict[key]
    tmp = 0
    for i in range(100):
        x = (x1-x2) * i / 99 + x2
        y = (y1-y2) * i / 99 + y2
        for a,b in child_data:
            if haversine(y,x,b,a) < 300:
                tmp += 1
                break
    freq.append(tmp/100)
```

```
freq
```

```
1.0,
0.85,
1.0,
1.0,
1.0,
0.85,
0.35,
1.0,
1.0,
0.34,
```

[그림 4] 어린이 보호구역 비중 코드[12]

[11] https://github.com/GodMoonGoodman/NaverMap-Car-minimum-distance

[12] https://stackoverrun.com/ko/q/11234571

4. 모델 구현

a. Dijkstra algorithm 구현 (첨부파일 5 참고)

다익스트라 함수 구현하기

```python
import heapq

def dijkstra(graph, start, end):
    distances = {vertex: [float('inf'), start] for vertex in graph}
    distances[start] = [0, start] # 시작 지점과 시작 지점까지 거리 0으로 초기화

    queue = []
    heapq.heappush(queue, [distances[start][0], start])

    while queue:
        # 큐에서 정점을 하나씩 꺼내 인접한 정점들의 가중치를 모두 확인하여 업데이트합니다.
        current_distance, current_vertex = heapq.heappop(queue)

        # 더 짧은 경로가 있다면 무시한다.
        if distances[current_vertex][0] < current_distance:
            continue

        for adjacent, weight in graph[current_vertex].items():
            distance = current_distance + weight
            # 만약 시작 정점에서 인접 정점으로 바로 가는 것보다 현재 정점을 통해 가는 것이 더 가까울 경우에는
            if distance < distances[adjacent][0]:
                # 거리를 업데이트합니다.
                distances[adjacent] = [distance, current_vertex]
                heapq.heappush(queue, [distance, adjacent])

    path = end
    path_output = [end]
    while distances[path][1] != start:
        path_output.append(distances[path][1])
        path = distances[path][1]
    path_output.append(start)

    return distances[end][0], path_output[::-1]
```

[그림 5]Dijkstra algorithm 구현하기

1. 그래프를 (시작 노드, 도착 노드, 비용)의 집합으로 만들어 준다.

2. 구로구 내에서 출발점과 도착점의 주소를 입력하면 네이버 API 에 주소를 검색하여 위도, 경도 값을 알아온 다음에 각 노드와의 거리를 비교하여 출발 노드와 도착 노드를 설정한다. 각 거리의 계산은 haversine 공식을 사용하였다.

3. 비어 있는 큐를 하나 만들고, 각 노드에 대해 탐색한 적이 있는지 확인하기 위한 빈 리스트를 만든다.

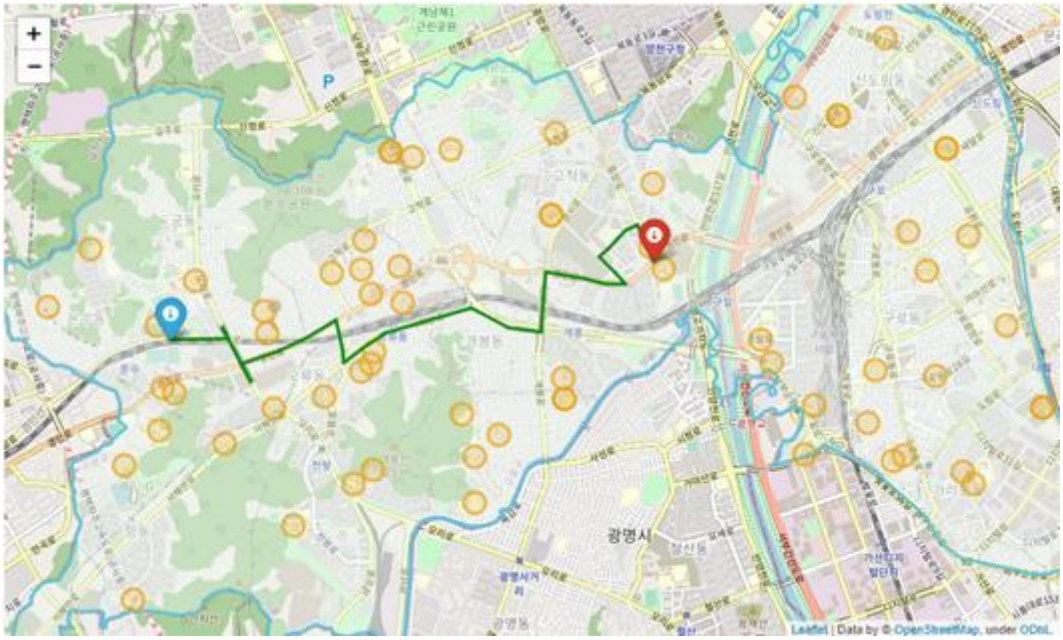
4. 출발 노드를 큐에 넣어준다.

5. 큐가 빌 때까지 DFS 와 유사하게 탐색을 수행한다. 즉, 반복문을 한번 수행할 때마다 큐의 원소를 하나 뺀 다음에 인접한 노드들에 대해서 아직 탐색한 적이 없거나 더 비용이 싼 경로를 찾은 경우 큐에 노드를 넣어준다. 이때 거리와 이전 노드 정보를 같이 넣어줘 비용 뿐만 아니라 이동 경로도 함께 알 수 있도록 해준다.

6. 탐색이 끝났으면 도착 노드에 저장된 정보 (비용, 이동 경로)를 return 한다.[13]

[13] https://www.fun-coding.org/Chapter20-shortest-live.html

b. 알고리즘 결과 시각화



[그림 6]시각화 한 알고리즘

- 프로젝트에서 설정한 범위인 구로구의 경계를 파란색 선으로 나타냈다.
- 구로구 내의 어린이 보호구역의 위치와 대략적인 반경을 노란색 원으로 나타냈다.
- 출발 지점을 파란색 마커로, 도착 지점을 빨간색 마커로 표시하고 경로를 초록색 선으로 표시하였다.

c. 구현된 기능

구현된 기능들은 다음과 같다.

1. 사용자가 원하는 기준에 따라 스쿨존 위험 가중치의 크기 다르게 설정(각각 4 단계, 기존의 최소시간만을 목적으로 하는 네비게이션으로도 이용 가능)

2. 가중치 중 시간에 따라 변하는 값들은 '평일/주말, 일별 4 개의 시간대'로 나뉜 8 개의 시간 구역에 따라 다르게 지정해 효용성 증가

3. 사용자가 원하는 출발 날짜, 시간 설정

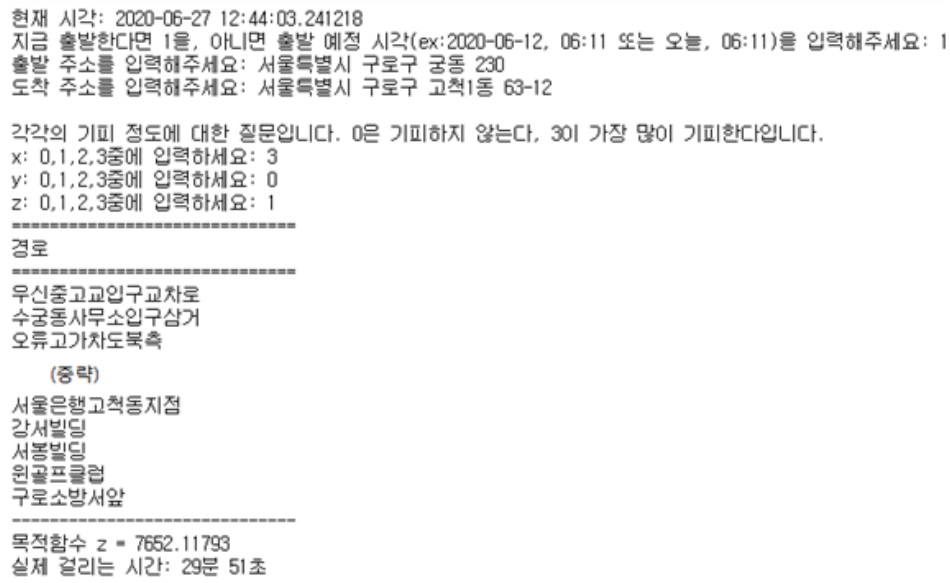
4. 표준 노드, 링크를 모두 활용해 최대한 자세한 경로 안내

5. 임의의 도로명 주소로 출발지, 도착지를 입력하면 가장 가까운 표준 노드 자동 인식

6. Dijkstra's algorithm 을 이용하여 사용자만의 목적함수를 최소화하는 경로 안내: 지도에 경로를 선으로 표시, 스쿨존을 원으로 표시해 추천 경로가 스쿨존을 피해감을 인지

7. 경유하는 모든 표준 노드를 나열해 경로를 자세히 안내

8.네이버 지도 API 를 활용해 보다 정확한 예상 소요 시간 안내



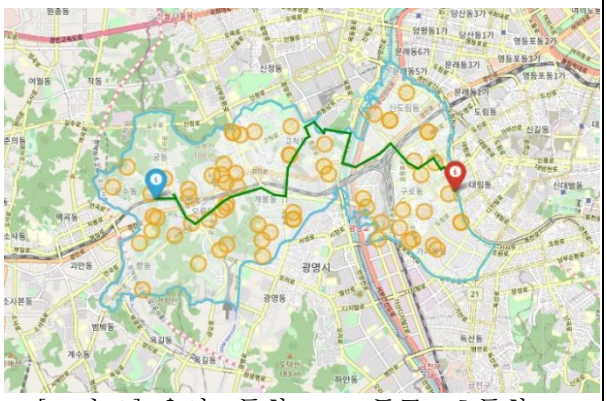
[그림 7] 사용자 입력 예시

사용자 입력은 다음과 같이 출발 시각, 출발 주소, 도착 주소, 그리고 x, y, z 값을 입력 받는 형식으로 이루어진다. 그 아래에 목적함수 결과 값과 실제 소요 시간, 경로가 출력된다. (여백상 경로 일부분을 생략하였다.)

5. 모델 적용(첨부파일 5 참고)

a. 기존 네비게이션과의 비교

<기존>



[그림 8] 우신고등학교 → 동구로초등학교

2020-06-26, 16:11

출발: 서울특별시 구로구 궁동 230
(우신고등학교)

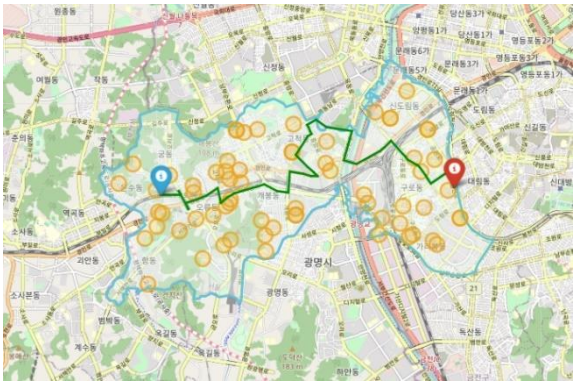
도착: 서울특별시 구로구 구로동 93
(동구로초등학교)

$(x, y, z) = (0,0,0)$

목적함수 $z = 3508.0$

실제 걸리는 시간: 58분 28초

<스쿨존을 고려한 경우>



[그림 9] 우신고등학교 → 동구로초등학교

2020-06-26, 16:11

출발: 서울특별시 구로구 궁동 230
(우신고등학교)

도착: 서울특별시 구로구 구로동 93
(동구로초등학교)

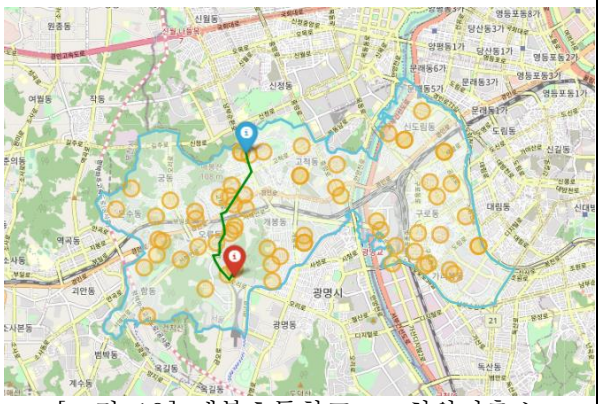
$(x, y, z) = (3,2,3)$

목적함수 $z = 231026.17643000008$

실제 걸리는 시간: 61분 6초

기존 네비게이션의 경로와 달리 스쿨존을 피해서 경로를 추천하는 것을 확인할 수 있다. 특히, 스쿨존 사고 위험율이 높은 시간 구역대인 16시 경(하교시간)에서의 예시이다.

<기존>



[그림 10] 매봉초등학교 → 천왕파출소

2020-06-23,09:00

출발: 서울 구로구 고척로21길 55
매봉초등학교

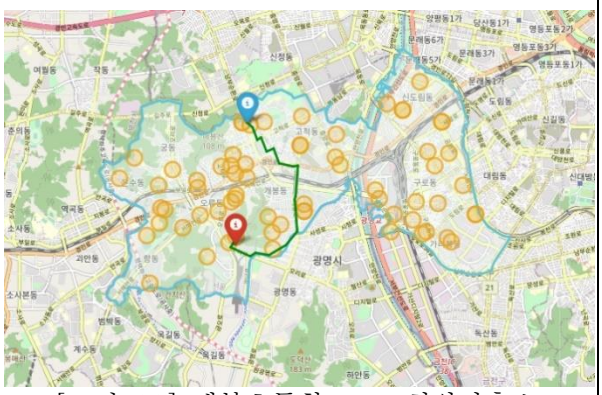
도착: 서울 구로구 오리로 1102-8
천왕파출소

$(x, y, z) = (0,0,0)$

목적함수 $z = 1011.0$

실제 걸리는 시간: 16분 51초

<스쿨존 고려>



[그림 11] 매봉초등학교 → 천왕파출소

2020-06-23,09:00

출발: 서울 구로구 고척로21길 55
매봉초등학교

도착: 서울 구로구 오리로 1102-8
천왕파출소

$(x, y, z) = (3,2,3)$

목적함수 $z = 38478.87236$

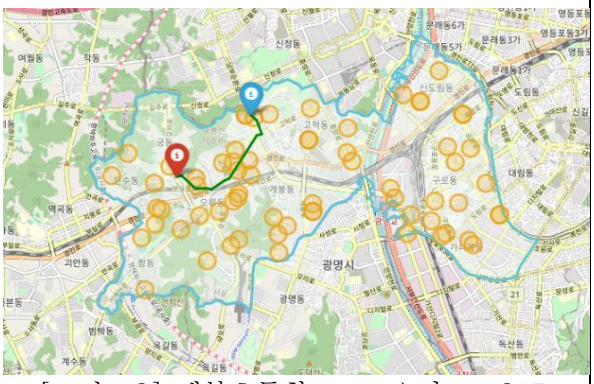
실제 걸리는 시간: 21분 40초

마찬가지로, 기존 네비게이션의 경로와 달리 스쿨존을 피해서 경로를 추천하는 것을 확인할 수 있다. 스쿨존 사고 위험율이 높은 시간 구역대인 9시 경(등교시간)에서의 예시이다.

b. 시간 구역, 사용자 설정에 따른 결과

i) x에 차이를 둔 경우

<낮은 스쿨존 가중치>



[그림 12] 매봉초등학교 → 오리로 1285

2020-06-27, 01:00

출발: 서울 구로구 고척로21길 55
매봉초등학교

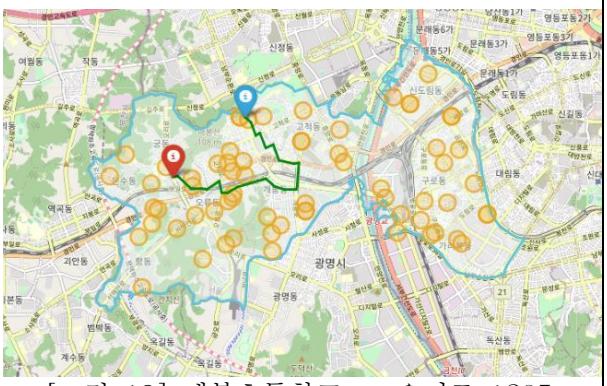
도착: 서울 구로구 오리로 1285

$(x, y, z) = (1,0,3)$

목적함수 $z = 2108.96063$

실제 걸리는 시간: 18분 58초

<높은 스쿨존 가중치>



[그림 13] 매봉초등학교 → 오리로 1285

2020-06-27, 01:00

출발: 서울 구로구 고척로21길 55
매봉초등학교

도착: 서울 구로구 오리로 1285

$(x, y, z) = (3,0,3)$

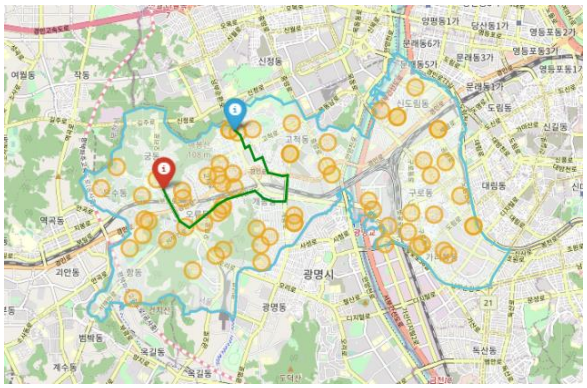
목적함수 $z = 7258.671730000002$

실제 걸리는 시간: 30분 31초

높은 스쿨존 가중치를 부여하면 시간만을 고려한 최적의 경로가 아니기에 걸리는 시간은 늘어나지만, 경로 상 스쿨존이 적게 포함됨을 확인할 수 있다.

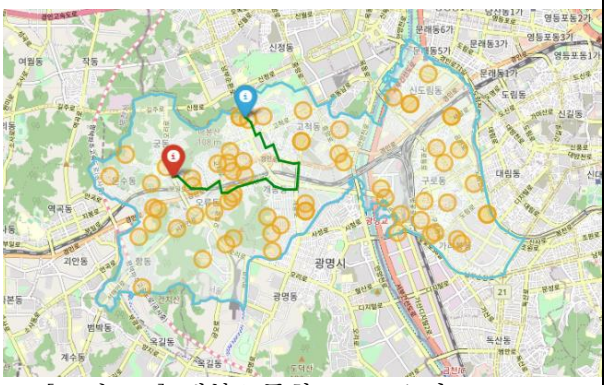
ii) t 에 차이를 둔 경우

<table>
<tr>
<td><스쿨존 사고율이 높은 시간대>

 
[그림 14] 매봉초등학교 → 오리로 1285
2020-06-26, 16:11
출발: 서울 구로구 고척로21길 55
매봉초등학교
도착: 서울 구로구 오리로 1285

$(x, y, z) = (3,0,3)$

목적함수 $z = 41862.020079999995$
실제 걸리는 시간: 27분 11초</td>
<td>< 스쿨존 사고율이 낮은 시간대>

 
[그림 15] 매봉초등학교 → 오리로 1285
2020-06-27, 01:00
출발: 서울 구로구 고척로21길 55
매봉초등학교
도착: 서울 구로구 오리로 1285

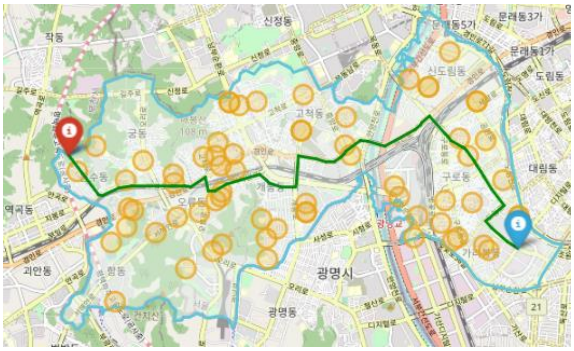
$(x, y, z) = (3,0,3)$

목적함수 $z = 7258.671730000002$
실제 걸리는 시간: 30분 31초</td>
</tr>
</table>

스쿨존 사고율이 높은 시간대의 경우, 경로 상에 스쿨존이 더욱 적게 포함됨을 확인할 수 있다. 학생들의 이동이 적어 사고율이 낮은 새벽 1시보다 하교시간인 16시에 스쿨존을 더욱 기피하는 예시이다.

iii) b 에 차이를 둔 경우

<낮은 통행량 가중치>



[그림 16] 구로동 212-5 → 온수동 73-5

2020-06-27 13:30

출발: 서울특별시 구로구 구로동 212-5

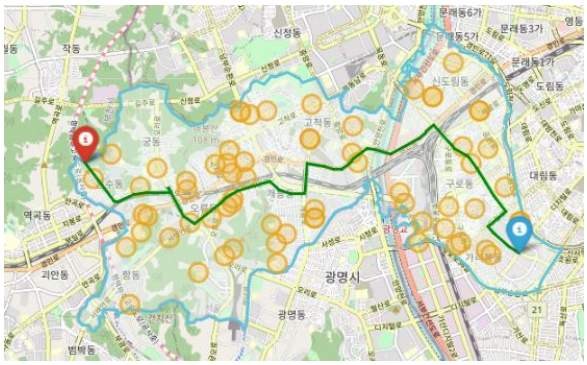
도착: 서울특별시 구로구 온수동 73-5

$(x, y, z) = (2,1,1)$

목적함수 $z = 15049.07663$

실제 걸리는 시간: 65분 39초

<높은 통행량 가중치>



[그림 17] 구로동 212-5 → 온수동 73-5

2020-06-27 13:30

출발: 서울특별시 구로구 구로동 212-5

도착: 서울특별시 구로구 온수동 73-5

$(x, y, z) = (2,3,1)$

목적함수 $z = 49320.35875$

실제 걸리는 시간: 64분 49초

높은 통행량 기피 가중치를 부여하면, 스쿨존을 더 많이 포함하는 경로이더라도 실제 걸리는 시간이 줄어드는 경로를 추천함을 확인할 수 있다. 경로 상은 [그림 17]이 더 돌아가는 것처럼 보이지만, 더 높은 b를 입력한 만큼 실제 소요시간은 줄어든 예시이다.

c. OPL 을 이용한 수리 모델과 알고리즘의 성능 비교

지금까지 알고리즘을 코딩으로 구현해 솔루션을 구했는데, 그 효용성을 알아보기 위해 '본론-2'에서 설정한 LP 모델을 이용하여 OPL 로 구현해보았다. (첨부파일 6 참고)

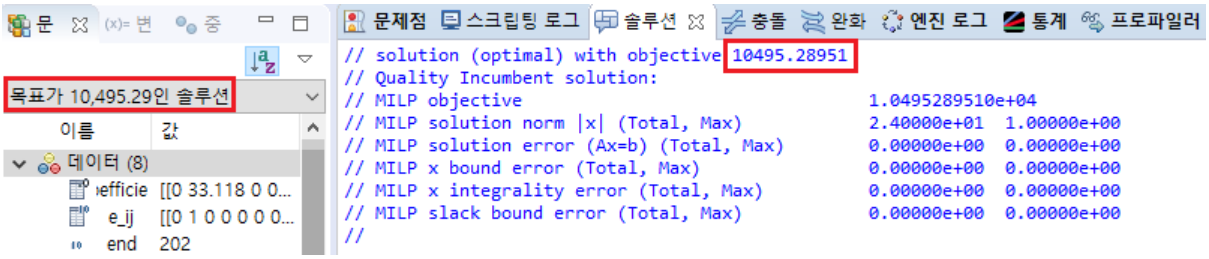
'우신중고교입구교차로'을 출발 지점으로, '구로소방서앞'을 도착 지점으로 하는 경로를 통해 비교하고자 한다.

$t = 0$ (평일, 구간 1), $x = 3, y = 1, z = 2$ 인 상황에서, OPL 을 이용한 수리 모델과 Dijkstra's algorithm 을 비교해보았다.

수리 모델과 알고리즘 각각에서 목적함수 $z = 10495.28$를 갖는,

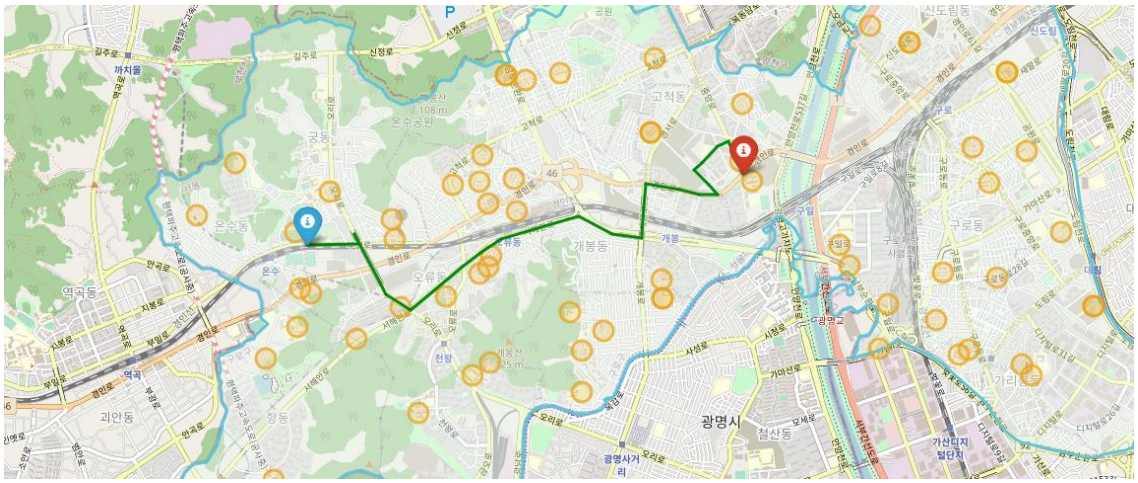
우신중고교입구교차로 -> 수궁동사무소입구삼거 -> 오류고가차도북측 -> 오류고가차도남측 -> 금강수목원아파트 -> 광덕사거리 -> 오류동역교차로 -> 미래빌라 -> 오류동역 2 번출구 -> 현대홈맨션 -> 성은빌라 -> 엑스포카센타 -> 개봉고가차도 -> 성한빌딩 -> 고척동국제외국어학원 -> 개봉사거리 -> 금석빌딩앞 -> 개봉동 179-46 -> 한마음아파트삼거리 -> 영동빌딩 -> 서울은행고척동지점 -> 강서빌딩 -> 서봉빌딩 -> 윈골프클럽 -> 구로소방서앞

의 경로를 제공받음을 확인할 수 있다.



[그림 18] OPL 솔루션

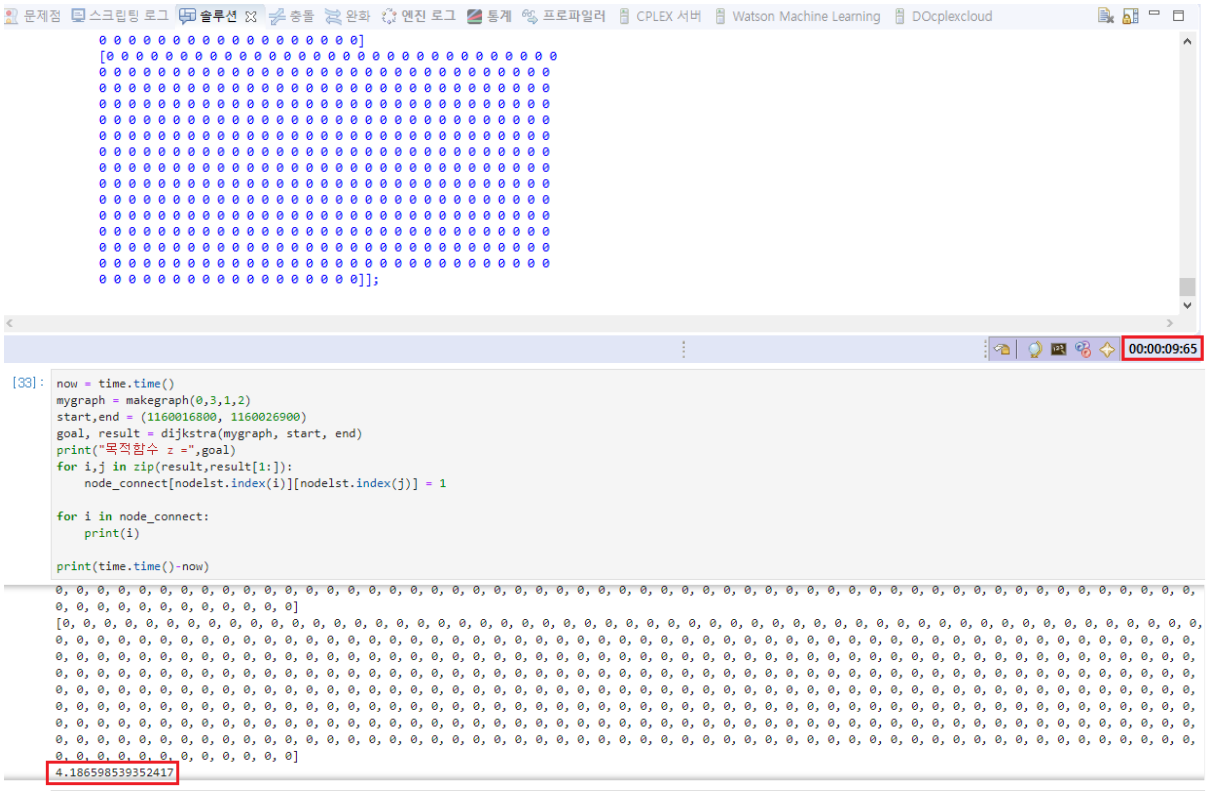
목적함수 z = 10495.289509999999
실제 걸리는 시간: 18분 45초



[그림 19] Dijkstra's algorithm 의 결과

둘은 동일한 목적함수로, 동일한 경로를 제공한다. 그러나, 둘의 computation time 을 비교해보면 유의미한 차이를 발견할 수 있다.

OPL 을 이용한 수리 모델 솔루션을 구하기까지는 9.65 초가, 다익스트라 알고리즘을 이용해 솔루션을 구하기까지는 4.19 초가 소요되었음을 확인할 수 있는데, 이는 결과는 같지만 알고리즘의 효용성을 보여준다고 할 수 있다.



[그림 4] OPL 과 다익스트라 알고리즘의 솔루션 구현 시간 차이

Ⅲ.결론

1. 프로젝트 의의

먼저 이번 프로젝트의 의의에 대해 생각해보았다.
첫째, 시간적인 요소만 고려하였던 기존의 네비게이션과 달리 사용자에게 '어린이 보호구역 위험도'의 요소를 고려한 경로를 제시할 수 있게 하였으며 사용자는 목적지까지의 경로에 관해 교통량, 사고 위험도 등 각 요소에 대한 선호도를 표현하여 사용자 맞춤형으로 최적화된 경로를 이용할 수 있다는 의의가 있다.

둘째, 프로젝트에서 설정했던 목표에 부합하는 결과를 도출한 후 분석을 통해 결과에 담긴 함의를 살펴보고자 하였다. 실제 네비게이션과 같은 활용을 고민하여 알고리즘 결과의 시각화 구현을 시도하였고, 기존 네비게이션과의 비교 분석으로 알고리즘 결과의 성능 및 유 의미성에 대한 분석 등이 이루어졌다.

셋째, 이번 프로젝트 내에서는 구로구에 한정하여 모델을 개발했지만, 추후 서울특별시 전체, 혹은 데이터가 존재한다면 전국 단위까지 현재의 모델을 더 확장할 수 있겠다.

2. 프로젝트 한계

이번 프로젝트의 한계는 다음과 같다.
첫째, parameter 설정에 있어 시간적 제약 및 방대한 데이터 양으로 인해 현실에 적용될 모든 parameter 변수들을 완벽하게 구현하지는 못하였다.
우선 모델에서 고려하지 못한 parameter 들이 있었다. 모델에서는 스쿨존 위험도를 세 개의 기준만으로 평가하였다. 그러나 보다 정확한 경로 설정을 위해서는 개별 학교 규모, 교통신호기, 과속단속카메라 설치 개수 등 다른 요소들 또한 고려해야 할 것이다.
더 나아가 모델에서 구현하고자 했으나 목표한 단계만큼 구체화하지는 못한 parameter 들도 있었다. 교통량을 하나의 parameter 로 선정했지만 도심 내 주도로가 아닌 단거리 경로에 대한 교통량 데이터 수집에 어려움이 있었다. 그래서 주행속도 데이터를 수집해 교통량을 예측하는 방법을 채택하게 되었다. 그러나 주행속도 데이터도 그 양이 적었으며, 단순히 역수를 취하는 방식으로 parameter 를 반영하였기에 데이터에 대한 신뢰성을 보장하지 못했다. 따라서 후속연구에서는 보다 정교한 방식의 데이터 가공이 필요할 것이다.
또한 시간대별 교통사고량도 요일별, 시간대별 구간으로 나누어 데이터를 적용할 수 있었지만 월별 데이터는 수집하지 못했다. 스쿨존 위험도를 논할 때 방학과 학기중을 구분하는 것도 중요한 요인으로 작용할 것이라 예상되었으나 모델에 실제로 구현하지는 못하게 되었다.

둘째, 모델을 구현하는 과정에서 약간의 오차 및 한계가 발생했다. 위도와 경도를 기반으로 지구가 구라고 가정하고 구면좌표에서의 두 점 사이의 거리를 계산하였지만, 실제로는 살짝 찌그러진 구에서의 좌표 값이기 때문에 최대 0.5%의 오차가 발생할 수 있다고 한다. 또한 표준 노드 데이터를 사용하긴 하였으나 골목길같이 세세한 작은 지도 데이터를 모두 반영하지는 못하였다. 따라서 이를 근사하기 위해 네이버 길 찾기의 데이터를 활용하였으며 실제 도로주행에서의 신호등에 따른 차이는 고려하지 못했다는 아쉬움이 있다.

셋째, Dijkstra's algorithm 을 모델을 구현할 알고리즘으로 선정한 것이 적합하였는지에 대해 다른 알고리즘과 비교하는 등 종합적인 근거를 제시하는 부분이 부족했다고 판단된다. 강의에서 배운 내용을 바탕으로 조원들 간 논의를 통해 Dijkstra's algorithm 을 선정하였으나 추후 연구를 발전시킨다면 보다 다양한 알고리즘을 평가하여 가장 적합한 형태의 알고리즘을 선정하는 절차를 고민해볼 필요가 있다고 여겨진다.

추가로, Folium을 이용한 지도 데이터 시각화 과정에서, 어린이보호구역을 300m 반경의 주황색 원으로 구현했다. 그러나, 이 반경은 초기 지도에서는 비율에 맞게 구현되지만 지도를 확대/축소할 경우 비율이 왜곡된다. 이 현상을 코딩을 통해 해결하지 못하였다. 이에 본 보고서에서 제시한 모든 예시는 초기 지도의 왜곡없는 모습을 사용했음을 밝힌다.

Ⅳ.참고문헌

[신문]

1. 2020.04.01, 모터그래프, 신화섭, 아틀란 내비, '스쿨존 회피' 업데이트 후 다운로드 6 배 급증
2. 2019.12.11, 한국경제, 김명일, 여론에 떠밀려 두 달 만에 처리 "민식이법 자해공갈단 나올 판"
3. 2020.04.30, 머니투데이, 유동주, 민식이 아버지도 잘못 알고 있는 '민식이법'
4. 2020.04.29, 경기일보, 장영준, [와글와글 커뮤니티] 민식이법 이용해 협박?..."신고 안 하겠다" 합의금 요구
5. 2019.12.19, 연합뉴스, 이은중, [2019 사건 그후] ⑤ '민식이법' 제정 부른 아산 스쿨존 교통사고
6. 2020.01.29, TGN (땡큐 굿 뉴스), 임규수, "구로구, 어린이 교통사고 예방 위해 발 벗고 나섰다"

[학위논문]

1. 백재진, 교통량 예측 기반 최적경로 탐색엔진의 개발 및 구현, 중앙대학교 대학원, 컴퓨터공학과 휴먼인터페이스 전공 석사 학위논문, 2008.08

[단행본]

1. 홍상연, 정재훈, 2018, 도로교통사고 예방 위한 위험도 평가기법 모색, 서울연구원
2. 김준기. 교통사고에 안전한 국토 구현. 경기도: 국토연구원, 2014.
3. (No.2015-02)주요 사회경제지표와 교통지표간 상관관계. 경기도 교통 DB 센터, 2015

[인터넷 사이트]

1. 서울 열린데이터 광장 데이터셋
   https://data.seoul.go.kr/dataList/datasetList.do
2. 공공데이터포털
   https://www.data.go.kr/tcs/dss/selectDataSetList.do?keyword=
3. 교통사고상세통계
   http://taas.koroad.or.kr/web/shp/sbm/initUnityAnalsSys.do?menuId=WEB_KMP_OVT_UAS
4. 어린이 교통사고 통계 // 전체 교통사고 통계 (서울시)

https://data.seoul.go.kr/dataList/570/S/2/datasetView.do
https://data.seoul.go.kr/dataList/322/S/2/datasetView.do

5. 어린이보호구역
https://www.google.com/maps/d/u/0/viewer?mid=1dp1gE0XPSaXdX1cJXXxgnR2KdoRJtUU_Q&ll=37.57822414437999%2C126.9675696616819&z=13

6. 표준노드링크 관리시스템
http://nodelink.its.go.kr/Nodelink.aspx
http://nodelink.its.go.kr/intro/intro05.aspx

7. 좌표 체계 변환 관련 사이트
https://www.osgeo.kr/17
(중부원점 좌표계-국토지리정보원의 표준, WGS84 표준 좌표계-네이버지도 사용)

8. 파이썬 코딩을 위해 참조한 사이트
https://github.com/GodMoonGoodman/NaverMap-Car-minimum-distance
https://stackoverrun.com/ko/q/11234571
https://www.fun-coding.org/Chapter20-shortest-live.html