

TẠ TUẤN ANH

QUẢN LÝ HỆ THỐNG THÔNG TIN

KHÓA 22 - THÁNG 10/2017

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Tạ Tuấn Anh

**THIẾT KẾ THUẬT TOÁN DI TRUYỀN ỨNG DỤNG
TRONG BÀI TOÁN TỐI ƯU THU GOM
CHẤT THẢI RẮN ĐÔ THỊ**

LUẬN VĂN THẠC SĨ NGÀNH CÔNG NGHỆ THÔNG TIN

HÀ NỘI - 2017

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Tạ Tuấn Anh

**THIẾT KẾ THUẬT TOÁN DI TRUYỀN ỨNG DỤNG
TRONG BÀI TOÁN TỐI ƯU THU GOM
CHẤT THẢI RẮN ĐÔ THỊ**

Chuyên ngành: Quản Lý Hệ Thống Thông Tin

Mã số: 8480205

LUẬN VĂN THẠC SĨ NGÀNH CÔNG NGHỆ THÔNG TIN

NGƯỜI HƯỚNG DẪN KHOA HỌC: TS. LÊ HOÀNG SƠN

HÀ NỘI - 2017

Lời cam đoan

Tôi cam đoan kết quả của luận văn là chính tôi thực hiện, các số liệu thực nghiệm là theo đúng kết quả của chương trình. Nếu sai tôi xin chịu hoàn toàn chịu trách nhiệm.

LỜI CẢM ƠN

Trong suốt quá trình học tập và hoàn thiện đề tài này, em đã nhận được sự hướng dẫn, giúp đỡ quý báu của các thầy cô, anh chị và bạn bè. Với lòng kính trọng và biết ơn sâu sắc, em xin được bày tỏ lời cảm ơn chân thành tới:

Đầu tiên, em xin gửi lời cảm ơn chân thành nhất tới sự hướng dẫn tận tình của TS. Lê Hoàng Sơn. Trong suốt thời gian thực hiện đề tài, mặc dù thầy rất bận rộn trong công việc nhưng thầy vẫn giành rất nhiều thời gian và tâm huyết trong việc hướng dẫn em hoàn thiện đề tài này. Trong quá trình thực hiện đề tài, Thầy luôn định hướng, góp ý và sửa chữa những chỗ sai giúp em không bị lạc lối trong biển kiến thức mênh mông.

Em cũng xin được gửi lời cảm ơn đến các thầy cô trong khoa Công Nghệ Thông Tin, Trường Đại Học Công Nghệ đã dạy bảo, giúp đỡ, tạo điều kiện cho em trong thời gian em đã học tập tại trường.

Xin được gửi lời cảm ơn các thầy cô, các anh chị và các bạn trong Trung tâm Tính toán Hiệu năng cao, Trường Đại học Khoa Học Tự Nhiên đã giúp đỡ em trong suốt quá trình học tập và nghiên cứu tại trung tâm.

Cuối cùng, em xin gửi lời cảm ơn tới gia đình, anh chị và bạn bè đã giúp đỡ, cổ vũ, động viên trong công việc, học tập nói chung cũng như trong quá trình thực hiện đề tài này.

Xin chúc mọi người luôn mạnh khỏe, đạt được nhiều thành tích trong công tác, học tập và nghiên cứu khoa học.

Em xin chân thành cảm ơn!

Học viên

Tạ Tuấn Anh

DANH MỤC BẢNG BIỂU

STT	Tên hình, bảng biểu	Trang
Bảng 2.1	Thuật toán Dijkstra cổ điển	25
Bảng 2.2	Thuật toán Dijkstra cải tiến	26
Bảng 3.1	Sức chứa chất thải của mỗi xe	36
Bảng 3.2	Sức chứa chất thải ban đầu tại tất cả các node	37
Bảng 3.3	Ký hiệu và định nghĩa	38
Bảng 3.4	Mô hình cho bài toán thu gom chất thải	39
Bảng 3.5	Lượng chất thải mỗi node (kg)	40
Bảng 3.6	Sức chứa chất thải của mỗi xe (kg)	40
Bảng 3.7	Khoảng cách giữa các node (km)	41
Bảng 3.8	Kết quả hành trình thứ nhất	42
Bảng 3.9	Kết quả hành trình thứ 2	42
Bảng 3.10	Biểu tượng của các node trên ArcGIS	44
Bảng 3.11	Kết quả thực nghiệm	45

DANH MỤC HÌNH ẢNH

STT	Tên hình, bảng biểu	Trang
Hình 1.1	Ví dụ về các loại rác thải tại Sfax, Tunisia năm 2016	4
Hình 2.1	Sơ đồ thực hiện thuật giải di truyền	10
Hình 2.2	Cấu trúc nhiễm sắc thể	14
Hình 2.3	Sơ đồ thực hiện thuật toán	31
Hình 3.1	Bản đồ Tunisia	35
Hình 3.2	Ví dụ về hệ thống thu gom rác	40
Hình 3.3	Dữ liệu trong ArcGIS	43
Hình 3.4	Dữ liệu các khoảng cách thời gian giữa các node được tính thông qua chức năng Network Analyst trong phần mềm ArcGIS	44
Hình 3.5	Export dữ liệu các khoảng cách thời gian giữa các node ra file Excel	45
Hình 3.6	Kết quả tuyến đường của xe 1 trong phương pháp Dijkstra	46
Hình 3.7	Kết quả tuyến đường của xe 2 trong phương pháp Dijkstra	46
Hình 3.8	Kết quả tuyến đường của xe 3 trong phương pháp Dijkstra	47
Hình 3.9	Kết quả tuyến đường của xe 4 trong phương pháp Dijkstra	47

Hình 3.10	Kết quả tuyến đường của xe thứ nhất trong phương pháp GA	48
Hình 3.11	Kết quả tuyến đường của xe thứ hai trong phương pháp GA	49
Hình 3.12	Kết quả tuyến đường của xe thứ ba trong phương pháp GA	49
Hình 3.13	Kết quả tuyến đường của xe thứ tư trong phương pháp GA	50

DANH MỤC CÁC THUẬT NGỮ

Thuật ngữ	Viết tắt	Giải thích
Municipal Solid Waste	MSW	Chất thải rắn đô thị
Agricultural tractor		Máy kéo nông nghiệp
Dumper truck		Xe có thùng lật nghiêng để đổ chất thải
Compactor vehicles		Xe tải trọng lớn
Depot		Kho chứa các xe
Gather sites		Các địa điểm đổ chất thải trong đô thị
Transfer stations		Trung tâm thu thập chất thải/ Trạm trung chuyển chất thải
node		Các điểm trên bản đồ
Vehicle Routing	VR	Định tuyến xe
id		Số thứ tự
Vehicle Routing Problem	VRP	Bài toán định tuyến xe
Network Analyst	NA	Chức năng phân tích mạng trong ArcGIS
National Agency for Waste Management	NAGEd	Cơ quan Quốc Gia về Quản lý chất thải
Travelling Salesman Problem	TSP	Bài toán người đưa hàng
Fitness		Độ thích nghi
Phương pháp Ranking	Ranking	Phương pháp chọn lọc xếp hạng
Genetic Algorithm	GA	Thuật toán di truyền
Tournament		Phương pháp chọn lọc Tournament
Local Search		Tìm kiếm cục bộ
Edge Recombination		Kỹ thuật lai ghép cạnh

MỤC LỤC

LỜI CẢM ƠN	
DANH MỤC BẢNG BIỂU	
DANH MỤC HÌNH ẢNH	
DANH MỤC CÁC THUẬT NGỮ	
MỤC LỤC.....	
MỞ ĐẦU.....	1
CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN TỐI ƯU THU GOM CHẤT THẢI RẮN ĐÔ THỊ	4
1.1. Các loại chất thải đô thị và nhu cầu thu gom	4
1.2. Bài toán tối ưu thu gom chất thải rắn đô thị	5
1.3. Các nghiên cứu liên quan	6
1.4. Mục tiêu nghiên cứu	7
1.5. Tổng kết chương	8
CHƯƠNG 2: THIẾT KẾ THUẬT TOÁN DI TRUYỀN CHO BÀI TOÁN TỐI ƯU THU GOM CHẤT THẢI RẮN ĐÔ THỊ.....	9
2.1. Tổng quan về thuật toán di truyền	9
2.2. Thiết kế thuật toán di truyền cho bài toán thu gom chất thải tối ưu.....	14
2.2.1. Mã hóa cá thể	14
2.2.2. Hàm Fitness	15
2.2.3. Chọn lọc	16
2.2.4. Lai ghép	18
2.2.5. Đột biến	23
2.2.6. Tìm kiếm địa phương với thuật toán Dijkstra	24
2.2.7. Chi tiết thuật toán	28
2.3. So sánh Dijkstra và GA	32
2.4. Tổng kết chương	32
Chương 3 – ỨNG DỤNG THUẬT TOÁN DI TRUYỀN CHO BÀI TOÁN TỐI ƯU THU GOM CHẤT THẢI RẮN ĐÔ THỊ TẠI THÀNH PHỐ SFAX, TUNISIA.....	34
3.1. Giới thiệu về khu vực nghiên cứu	34

3.2.	Kịch bản thu gom chất thải rắn.....	35
3.3.	Mô tả dữ liệu thu thập và yêu cầu	36
3.4.	Mô hình thu gom chất thải rắn đô thị tại Sfax	37
3.5.	Môi trường thực nghiệm.....	43
3.6.	Kết quả thực nghiệm.....	45
3.7.	Đánh giá và so sánh	50
3.8.	Tổng kết chương	51
KẾT LUẬN		53
TÀI LIỆU THAM KHẢO.....		54
PHỤ LỤC.....		57

MỞ ĐẦU

Môi trường có tầm quan trọng đặc biệt đối với đời sống con người, đối với động thực vật và sự phát triển của nhân loại. Trong những năm gần đây, cùng với sự phát triển kinh tế - xã hội, các ngành sản xuất kinh doanh dịch vụ ở các đô thị và khu công nghiệp được mở rộng và phát triển nhanh chóng, một mặt đóng góp tích cực cho sự phát triển của quốc gia, mặt khác lượng chất thải rắn không hợp vệ sinh ngày càng nhiều, là nguồn gốc chính gây ô nhiễm môi trường. Từ đó đặt ra yêu cầu cấp bách cho chính quyền địa phương và người dân là phải có kế hoạch làm sạch, thu gom thường xuyên các loại chất thải rắn ở các khu nhà ở cũng như khu đô thị và khu công nghiệp. Đó là các loại chất thải sinh hoạt, thức ăn dư thừa, các loại chất thải đường phố. Khối lượng chất thải rắn trong các đô thị càng tăng do tác động của sự gia tăng dân số, phát triển kinh tế - xã hội và sự phát triển về trình độ và tính chất tiêu dùng trong đô thị. **Bài toán tối ưu thu gom chất thải rắn đô thị** có thể là tối ưu về lượng chất thải thu thập được, hoặc thời gian đi thu thập, hoặc là quãng đường đi thu thập là tối ưu nhất, hoặc là tối ưu chi phí vận chuyển dựa trên một kịch bản thu gom cụ thể.

Chất thải rắn nếu không được quản lý và xử lý nghiêm túc sẽ có khả năng gây suy thoái môi trường nghiêm trọng. Do đó, chất thải rắn đã trở thành vấn đề bức xúc đối với toàn xã hội và cần được quan tâm quản lý thu gom triệt để. Hiện nay, với khối lượng phát sinh lớn nhưng tỷ lệ thu gom còn hạn chế, chất thải rắn sinh ra chưa được thu gom và xử lý triệt để. Vì vậy, bài toán tối ưu thu gom chất thải rắn đô thị đang là bài toán khó với hầu hết các quốc gia trên thế giới. Nó mang nhiều ý nghĩa về mặt môi trường, phát triển cảnh quan và tiết kiệm kinh tế. Tại mỗi thành phố sẽ có các phương tiện vận chuyển chất thải, những bãi đỗ xe của các xe làm nhiệm vụ, các điểm đổ chất thải tập trung, các điểm trung chuyển chất thải và các bãi đổ chất thải lớn. Tùy vào yêu cầu về thời gian, phương tiện vận chuyển và tuyến đường đi của các xe mà mỗi một thành phố sẽ có những kịch bản riêng cho việc thu gom chất thải.

Bài toán tối ưu thu gom chất thải rắn đô thị thuộc lớp bài toán tối ưu tổ hợp nên nhóm các phương pháp tối ưu tiến hóa thường được sử dụng. **Thuật toán di truyền** là một nhánh của tối ưu tiến hóa nhằm tìm kiếm giải pháp thích hợp cho các bài toán tối ưu vận dụng các nguyên lý của tiến hóa như: di truyền, đột biến, chọn lọc tự nhiên, và trao đổi chéo. Thuật toán sử dụng ngôn ngữ máy tính để mô phỏng quá trình tiến hoá của một tập hợp những đại diện trừu tượng gọi là những nhiễm sắc thể. Tập hợp này sẽ phát triển theo hướng chọn lọc những giải pháp tốt hơn. Thuật toán di truyền giúp tìm ra giải pháp tối ưu nhất trong điều kiện thời gian và không gian cho phép. Thuật toán di truyền xét đến toàn bộ giải pháp, bằng cách xét trước một số giải pháp, sau đó loại bỏ những thành phần không thích hợp và chọn những thành phần thích nghi hơn để tạo sinh và biến hóa nhằm mục đích tạo ra nhiều giải pháp mới có hệ số thích nghi cao. Do đó, **luận văn sẽ áp dụng thuật toán di truyền cho bài toán tối ưu thu gom chất thải rắn đô thị.**

Để kiểm chứng tính hiệu quả của thuật toán, luận văn sẽ triển khai kiểm chứng trên kịch bản thực nghiệm tại thành phố Sfax, Tunisia. Tunisia là một quốc gia ở Bắc Phi với diện tích và dân số tương đối nhỏ so với các quốc gia khác trên thế giới. Cùng với sự phát triển của đất nước, các ngành sản xuất kinh doanh dịch vụ ở các đô thị và khu công nghiệp đã không ngừng làm phát sinh thêm lượng chất thải ở đây. Lượng chất thải bình quân tính trên đầu người là 0.815 kg/ngày ở khu vực đô thị. Sfax là thành phố lớn thứ hai và là một trong những thành phố có lượng chất thải bình quân theo đầu người lớn nhất ở Tunisia. Bài toán thu gom chất thải rắn đô thị là một trong những vấn đề quan trọng hàng đầu ở thành phố này.

Nội dung chính của luận văn gồm có ba chương:

Chương 1 – Giới thiệu bài toán tối ưu thu gom chất thải rắn đô thị.

Chương này giới thiệu về bài toán tối ưu thu gom chất thải rắn đô thị và các nghiên cứu liên quan.

Chương 2 –Thiết kế thuật toán di truyền cho bài toán tối ưu thu gom chất thải rắn đô thị

Nội dung của chương này tổng quan về thuật toán di truyền và thiết kế thuật toán di truyền để giải bài toán tối ưu thu gom chất thải rắn đô thị.

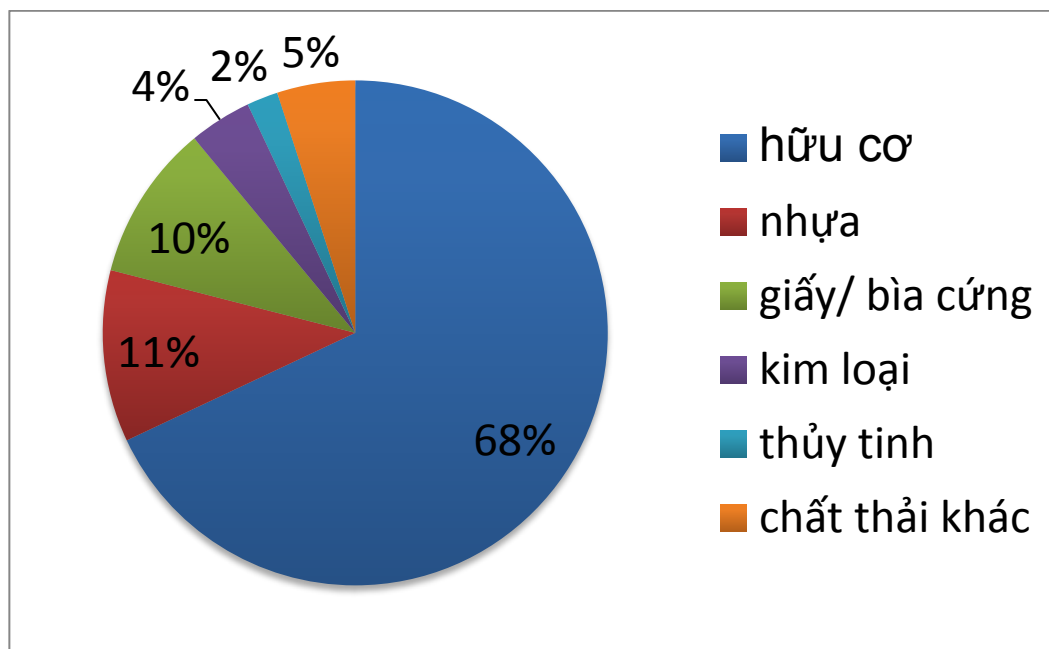
Chương 3 – Ứng dụng thuật toán di truyền trong bài toán tối ưu thu gom chất thải rắn đô thị tại thành phố Sfax, Tunisia

CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN TỐI ƯU THU GOM CHẤT THẢI RẮN ĐÔ THỊ

1.1. Các loại chất thải đô thị và nhu cầu thu gom

Trong những năm gần đây, cùng với sự phát triển kinh tế - xã hội, các ngành sản xuất kinh doanh dịch vụ ở các đô thị và khu công nghiệp được mở rộng và phát triển nhanh chóng, một mặt đóng góp tích cực cho sự phát triển của quốc gia, mặt khác lượng rác thải, chất thải thải ra ngoài môi trường ngày càng nhiều và ảnh hưởng rất lớn đến môi trường xung quanh là nguồn gốc chính gây ô nhiễm môi trường. Từ đó đặt ra yêu cầu cấp bách cho chính quyền địa phương và người dân là phải có kế hoạch làm sạch, thu gom, vận chuyển, xử lý thường xuyên các loại chất thải rắn ở các khu nhà ở cũng như khu đô thị và khu công nghiệp. Đó là các loại chất thải sinh hoạt, thức ăn dư thừa, các loại chất thải đường phố.

Thành phần của chất thải bao gồm chất thải hữu cơ, nhựa dẻo, giấy/bìa cứng, kim loại, thủy tinh và chất thải khác. Khối lượng chất thải rắn đô thị rất lớn nhưng chỉ có 70% lượng chất thải được đem đi chôn lấp.



Hình 1.1: Ví dụ về các loại rác thải tại Sfax, Tunisia năm 2016

Chất thải rắn là một mối quan tâm mang tính cấp thiết tại bất kỳ đô thị nào trên thế giới. Chất thải rắn là một trong những yếu tố chính gây biến đổi khí hậu và sự nóng lên của toàn cầu [3, 4]. Nó không chỉ làm ô nhiễm môi trường mà còn gián tiếp ảnh hưởng đến ách tắc giao thông, tài chính ngân sách và chất lượng cuộc sống. Ngày nay, hầu hết các nước đang phát triển trên thế giới hiện đang trong quá trình đô thị hóa và công nghiệp hóa, dẫn đến việc gia tăng lượng chất thải. Chính vì vậy mà việc thu thập và xử lý chất thải rắn, đặc biệt là trong bối cảnh các nước đang phát triển thực sự là một yêu cầu cấp thiết để bảo vệ môi trường, chất lượng cuộc sống và tuổi thọ của con người.

Chất thải rắn nếu không được quản lý và xử lý nghiêm túc sẽ có khả năng gây suy thoái môi trường nghiêm trọng dẫn tới nhiều hệ lụy. Do đó, nhu cầu thu gom chất thải rắn đã trở thành vấn đề bức xúc đối với toàn xã hội và cần được quan tâm quản lý thu gom triệt để. Nhu cầu thu gom chất thải rắn thì cấp bách cực kỳ tuy nhiên khối lượng chất thải rắn phát sinh lớn và tỷ lệ thu gom còn hạn chế nên chất thải rắn sinh ra chưa được thu gom và xử lý triệt để. Vì vậy, bài toán tối ưu thu gom chất thải rắn đô thị đang là bài toán khó với hầu hết các quốc gia trên thế giới.

1.2. Bài toán tối ưu thu gom chất thải rắn đô thị

Tối ưu thu gom chất thải rắn đô thị mang nhiều ý nghĩa về mặt môi trường, phát triển cảnh quan và tiết kiệm kinh tế.

Tại mỗi thành phố sẽ có các phương tiện vận chuyển chất thải, những bãi đỗ xe của các xe làm nhiệm vụ, các điểm đổ chất thải tập trung, các điểm trung chuyển chất thải và các bãi đổ chất thải lớn. Tùy vào yêu cầu về thời gian, phương tiện vận chuyển và tuyến đường đi của các xe mà mỗi một thành phố sẽ có những kịch bản riêng cho việc thu gom chất thải. Bài toán tối ưu có thể là tối ưu về lượng chất thải thu thập được, hoặc thời gian đi thu thập, hoặc là quãng đường đi thu thập là tối ưu nhất, hoặc là tối ưu chi phí vận chuyển. Từ kịch bản cụ thể, xây dựng mô hình cho bài toán để tìm ra các phương pháp giải quyết.

1.3. Các nghiên cứu liên quan

Vấn đề tối ưu thu gom MSW có thể được mô tả bởi mô hình định tuyến xe (VR) với các ràng buộc cơ bản như sức chứa và ràng buộc mạng lưới định tuyến xe. Nghiên cứu trong [12] và [20] cho thấy sự khác nhau giữa các tuyến đường dân cư và các tuyến đường thương mại. Giải pháp point – to- point là chấp nhận được cho các tuyến đường thương mại nhưng các tuyến đường dân cư đòi hỏi phải sử dụng giải pháp định tuyến phù hợp. Các nghiên cứu trong [6], [7], [10], [11], [21] cũng đồng quan điểm trên.

Bài toán thu gom chất thải cũng có thể được xây dựng như Node Routing Problem (NRP), tức là các xe phải đi qua một số các điểm [6], [13], [21], [22]. Tuy nhiên các phương pháp giải là rất rộng rãi và không có phương pháp hoàn hảo nào để giải quyết vấn đề về định tuyến xe.

Một số tác giả sử dụng phương pháp giải chính xác cho mô hình thu gom chất thải như [7] sử dụng lý thuyết đồ thị và các công cụ lập trình toán học. Tác giả đã đề xuất phương pháp giảm thiểu khoảng cách đi thăm cho mỗi xe và giảm thiểu tổng công việc cho các xe. Tác giả [10] sử dụng phương pháp Heuristic để chia hệ thống quản lý chất thải thành ba cấp độ. Mỗi cấp độ sử dụng bộ sưu tập riêng biệt hoặc chiến lược vận chuyển để thu gom và vận chuyển chất thải. Mỗi giai đoạn đã được tối ưu hóa. Tác giả đã đề xuất thuật toán Heuristic để giảm thiểu độ dài quãng đường vận chuyển và có một mục tiêu chính là xác định tối ưu việc thu gom và tuyến đường vận chuyển, để giảm chi phí vận chuyển trong hệ thống quản lý chất thải. Phương pháp này có thể giảm hơn 30% tổng quãng đường vận chuyển.

Các tác giả [13] sử dụng phương pháp meta-Heuristic trong thu gom chất thải tại Đài Loan. Có hai bước để xác định khoảng cách đi thu thập. Bước đầu tiên là tối ưu kế hoạch thu thập tại các điểm bao gồm tất cả các khu vực dân cư và bước thứ hai là áp dụng thuật toán Heuristics ACO để giải quyết tối thiểu các xe sử dụng và khoảng cách tối thiểu đi thu gom chất thải. Tác giả [20] cải thiện kết quả sử dụng hệ thống thông tin địa lý GIS. Nó được chứng minh là một công cụ mạnh mẽ với

khả năng cung cấp các thông tin không gian chi tiết và sử dụng hiệu quả các thuật toán định tuyến có sẵn như Dijkstra trong các phần mềm GIS cho việc tìm kiếm các giải pháp tối ưu. Một vài ví dụ được liệt kê như tác giả [20] sử dụng ArcGIS Network Analyst [27] để xác định tuyến đường tốt nhất cho việc thu thập chất thải đô thị. Tác giả [15] nghiên cứu quản lý chất thải đô thị ở Port Said, Ai Cập thông qua phần mềm MPL V4.2 [19]. Tác giả [18] dựa vào MapInfo [9] để tìm các tuyến đường tối ưu trong thành phố Trabzon, Thổ Nhĩ Kỳ. Tác giả [4] cho rằng ArcGIS có khả năng cập nhật và hiển thị các thông tin cần thiết.

ArcGIS sử dụng thuật toán định tuyến dựa trên Dijkstra cho việc tìm kiếm các giải pháp tối ưu, mục tiêu là giảm thiểu tổng khoảng cách thu thập, sử dụng 13 tuyến đường kết nối 13 phường và một trạm trung chuyển [22]. Kết quả của phương pháp đã tiết kiệm được đến 9.93% cho quãng đường. Tác giả [22] áp dụng ArcGIS để giải quyết vấn đề thu gom chất thải đô thị ở thành phố Đà Nẵng, Việt Nam. Tác giả trình bày một mô hình định tuyến xe mới để tối ưu hóa lượng chất thải thu được thông qua phương pháp lai mới giữa Chaotic Particle Swarm Optimization và ArcGIS để tạo ra giải pháp tối từ mô hình định tuyến xe ở Đà Nẵng.

1.4. Mục tiêu nghiên cứu

Để giải quyết bài toán tối ưu thu gom chất thải rắn đô thị, luận văn nghiên cứu tổng quan về thuật toán di truyền - là một thuật toán trong nhóm các thuật toán tối ưu tiến hóa nhằm tìm kiếm giải pháp thích hợp cho các bài toán tối ưu, từ đó ứng dụng xây dựng phương pháp tối ưu thời gian thu gom chất thải. Đó là thiết kế thuật toán di truyền cho bài toán tối ưu thu gom chất thải rắn.

Để kiểm chứng tính hiệu quả của thuật toán, luận văn sẽ triển khai ứng dụng dựa trên trên kịch bản, dữ liệu thu thập và yêu cầu tại thành phố Sfax, Tunisia - là thành phố lớn thứ hai và là một trong những thành phố có lượng rác thải bình quân theo đầu người lớn nhất ở Tunisia.

1.5. Tổng kết chương

Chương 1 đã trình bày bài toán tổng quan thu gom chất thải rắn. Có thể nhận thấy bài toán tối ưu thu gom chất thải rắn là một mối quan tâm mang tính cấp thiết tại bất kỳ đô thị nào trên thế giới. Nó mang nhiều ý nghĩa về mặt môi trường, phát triển cảnh quan và tiết kiệm kinh tế.

Để giải quyết khó khăn này, luận văn xây dựng phương pháp tối ưu thời gian thu gom chất thải. Đó là thiết kế thuật toán di truyền cho bài toán tối ưu thu gom chất thải rắn ở chương sau.

CHƯƠNG 2: THIẾT KẾ THUẬT TOÁN DI TRUYỀN CHO BÀI TOÁN TỐI ƯU THU GOM CHẤT THẢI RẮN ĐÔ THỊ

2.1. Tổng quan về thuật toán di truyền

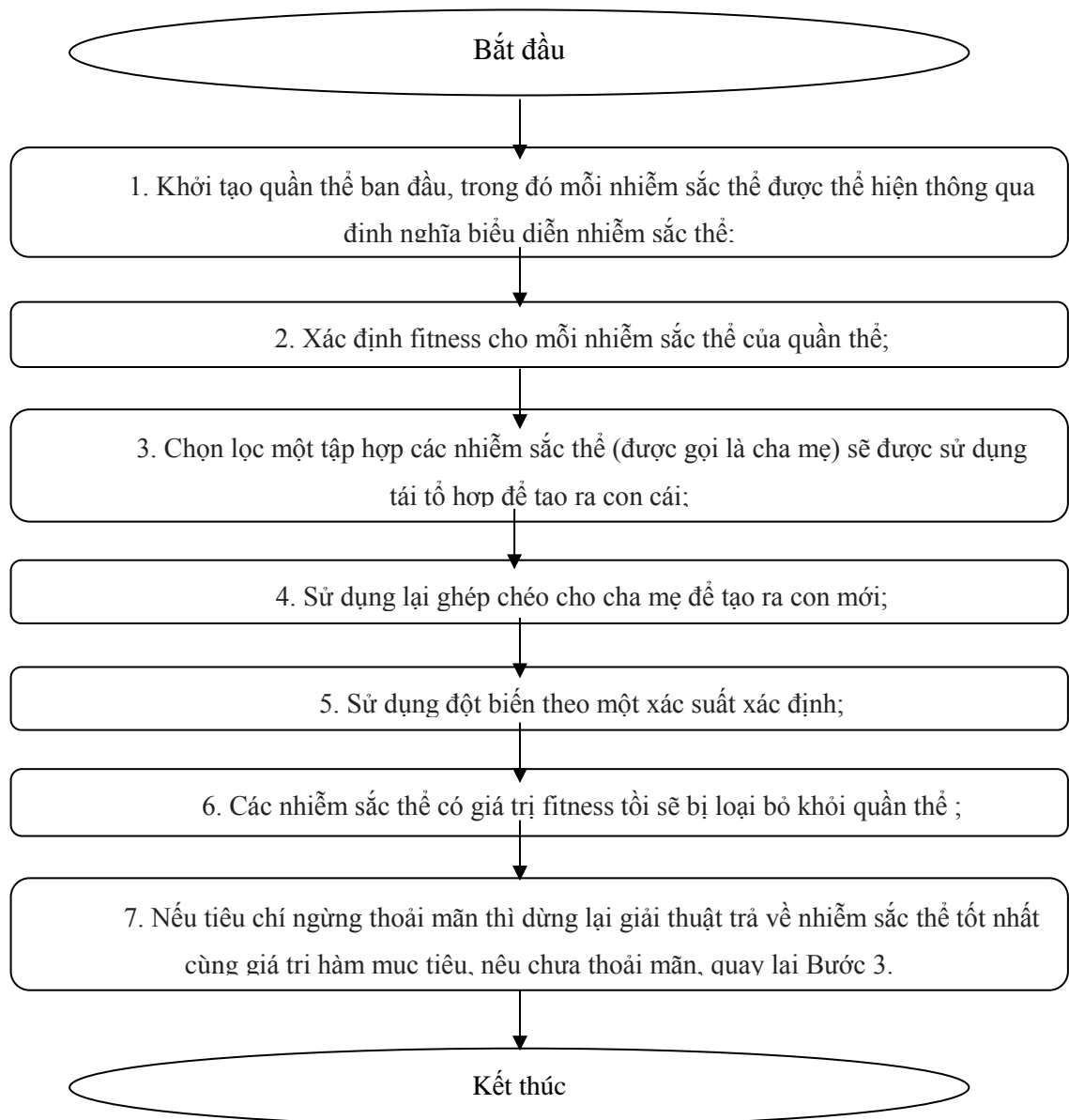
Hiện nay và trong tương lai, trí tuệ nhân tạo đã, đang và sẽ được nghiên cứu, phát triển rất mạnh mẽ và được ứng dụng rộng rãi. Đây là một mảng chuyên môn rất lớn trong khoa học máy tính, bao gồm nhiều lĩnh vực khác nhau. Một trong những lĩnh vực đó là kỹ thuật tính toán thông minh trong đó có Thuật toán di truyền đã đem lại những phương mới để giải bài toán mà nếu áp dụng những phương pháp truyền thống sẽ gặp nhiều khó khăn [1].

Thuật toán di truyền – chỉ cần nghe tên cũng có thể hiểu được rằng nó dựa trên việc quan sát quá trình tiến hóa trong tự nhiên. Các nguyên lý cơ bản của giải thuật di truyền được Holland công bố lần đầu tiên vào năm 1962. Sau đó các nền tảng toán học đầu tiên được công bố vào năm 1975 trong cuốn sách “Adaptation in Natural and Artificial System” cũng của Holland. Cũng có thể nói Holland là người đi tiên phong nghiên cứu trong lĩnh vực giải thuật di truyền cùng với Beglay [1].

Thuật toán di truyền được xây dựng dựa trên quy luật tiến hóa sinh học hay phát triển tự nhiên của một quần thể sống. Các cá thể trải qua một quá trình phát triển và sinh sản để tạo ra những cá thể mới cho thế hệ tiếp theo. Trong quá trình tăng trưởng và phát triển những cá thể xấu (theo một tiêu chuẩn nào đó hay còn gọi là độ thích nghi của nó trong môi trường) sẽ bị đào thải, ngược lại, những cá thể tốt sẽ được giữ lại (đây chính là quá trình chọn lọc) và được lai ghép (quá trình lai ghép) để tạo ra những cá thể mới cho thế hệ sau. Những cá thể mới được sinh ra mang những tính trạng của cá thể cha-mẹ (còn gọi là hiện tượng di truyền). Những cá thể được giữ lại có độ thích nghi khác nhau và quá trình lai ghép được thực hiện hoàn toàn ngẫu nhiên giữa các cá thể trong quần thể. Các cá thể được tạo ra trong quá trình lai ghép có thể sẽ xảy ra hiện tượng đột biến và tạo ra những cá thể khác với cá thể cha-mẹ. Cá thể này có thể tốt hơn hoặc xấu hơn cá thể cha-mẹ. Di truyền và đột biến là hai cơ chế có vai trò như nhau trong quá trình tiến hóa, mặc dù hiện

tượng đột biến xảy ra với xác suất nhỏ hơn nhiều so với xác suất của hiện tượng di truyền. Và quá trình lai ghép và chọn lọc là hai quá trình cơ bản xuyên suốt quá trình tiến hóa tự nhiên [2].

Sau đây là sơ đồ thực hiện thuật giải di truyền [26]:



Hình 2.1: Sơ đồ thực hiện thuật giải di truyền

Biểu diễn nhiễm sắc thể: Công việc đầu tiên khi thực hiện việc giải bài toán bằng thuật toán di truyền là chọn cách biểu diễn các cá thể còn gọi là nhiễm sắc thể. Đó là việc ánh xạ các tham số của bài toán lên một chuỗi có chiều dài xác định. Tùy theo từng bài toán cụ thể mà có nhưng cách biểu diễn khác nhau sao cho thích nghi. Trong đó có một số cách biểu diễn thông dụng là: biểu diễn theo dạng chuỗi ký tự, biểu diễn nhị phân, biểu diễn sử dụng hoán vị [26].

Quần thể và khởi tạo: Trong khởi tạo, một tập nhiễm sắc thể ban đầu được tạo ra, cũng gọi là quần thể ban đầu. Kích thước của quần thể hay số lượng nhiễm sắc thể rất quan trọng đối với thuật toán di truyền tổng quát, cần phải cân nhắc. Nếu chọn kích thước quá nhỏ có thể chỉ dẫn đến một kết quả tối ưu địa phương, không phát huy được hiệu quả của thuật toán, trong khi chọn kích thước lớn đưa ra một xác suất cao hơn tối ưu toàn cục sẽ được tìm thấy, tuy nhiên, thời gian tính toán tăng lên, chương trình chạy chậm lại. Kích thước quần thể tốt nên ở trong khoảng 20-30, tuy nhiên đôi khi kích thước 50 -100 vẫn được xem là tốt [26].

Đánh giá và chọn lọc tiến hóa: Toán tử chọn lọc được sử dụng để xác định nhiễm sắc thể sẽ được sử dụng cho thế hệ kế tiếp. Nhiều kỹ thuật khác nhau có thể được sử dụng trong toán tử chọn lọc, tuy nhiên, thường là một quá trình chọn lọc được mô phỏng, trong đó các nhiễm sắc thể "mạnh nhất" được sử dụng trong di truyền. Một trong những phương pháp chọn lọc được gọi là bánh xe Roulette, một bánh xe được chia thành các phần theo giá trị Fitness của các nhiễm sắc thể trong quần thể, ở đây những nhiễm sắc thể tốt hơn có phần bánh xe lớn hơn và những nhiễm sắc thể yếu sẽ nhận được một phần nhỏ của bánh xe. Vì vậy, xác suất được lựa chọn là tỷ lệ thuận với giá trị Fitness. Khi bánh xe quay, nhiễm sắc thể có độ thích nghi cao hơn sẽ có cơ hội được lựa chọn nhiều hơn, và ngược lại [26].

Phương pháp thứ hai để chọn lọc là phương pháp Ranking. Trong phương pháp Ranking, tất cả các nhiễm sắc thể trong quần thể được sắp xếp theo giá trị Fitness $f(x)$ để phân cấp, các nhiễm sắc thể với giá trị Fitness tốt hơn được xếp

hạng cao hơn. Trong TSP, thường là $f(x)$ xác định độ dài của toàn bộ đường đi, tuy nhiên, hàm này có thể bao gồm bổ sung các đặc điểm và các độ đo để giữ cho các nhiệm sắc thể trong quần thể. Nếu trong phương pháp bánh xe Roulette, giá trị Fitness được sử dụng khi gán một xác suất để được chọn, trong phương pháp Ranking các nhiệm sắc thể được chọn lọc theo hạng [26].

Một phương pháp khác để lựa chọn được gọi là chọn lọc Tournament. Phương pháp này sử dụng đặc điểm từ phương pháp Ranking nhưng có sự điều chỉnh, Lựa chọn Tournament đánh hạng chỉ một nhóm con các nhiệm sắc thể. Lúc đầu, chọn hai nhóm con từ quần thể. Mỗi nhóm con phải chứa ít nhất hai nhiệm sắc thể. Các nhiệm sắc thể được xếp hạng trong một nhóm như trong phương pháp chọn lọc Ranking. Các nhiệm sắc thể tốt nhất từ mỗi nhóm được lựa chọn để sinh sản, và các nhiệm sắc thể tồi nhất được chọn để loại bỏ khỏi quần thể. Để tạo ra một lứa con mới gồm l phần tử, trong mỗi lần lặp người ta chọn l nhóm con sau đó thực hiện chọn l cặp cha mẹ để sản sinh ra l phần tử con [5].

Tái tổ hợp hay lai ghép: Một phần quan trọng của thuật toán di truyền là toán tử lai ghép. Toán tử chéo mô phỏng sự sinh sản giữa hai nhiệm sắc thể, ở đây con cái được tạo ra thừa hưởng một số đặc điểm từ các nhiệm sắc thể cha mẹ. Nhiều toán tử chéo và đột biến với các nhiệm sắc thể được mã hoá dưới dạng một chuỗi ký hiệu hoặc số. Một số toán tử chéo được sử dụng nhiều trong bài toán VRP và TSP như sau [16], [24] :

- Lai ghép một điểm cắt
- Lai ghép vòng
- Lai ghép sắp xếp
- Lai ghép đồng bộ
- Lai ghép kết hợp cạnh
- Lai ghép trộn

Đột biến là một giải pháp xử lý vấn đề tối ưu hóa địa phương và tăng khả năng tìm ra sự tối ưu toàn cục [14]. Trong toán tử đột biến, một nhiễm sắc thể mới được tạo ra từ giải pháp đơn lẻ được chọn bởi thay đổi một số đặc điểm trong nó. Trong thuật toán di truyền, các toán tử chéo và đột biến được áp dụng bởi xác suất xác định trước. Chúng ta có thể tìm thấy các giá trị cho những xác suất này được trong đề xuất của Srinivas và Patnaik (1994) [23], Hong và các cộng sự [14].

Duy trì sự đa dạng và áp lực chọn lọc: Hai yếu tố quan trọng của di truyền thuật toán là đa dạng quần thể và áp lực chọn lọc. Hai yếu tố này có liên quan: nếu áp lực chọn lọc đang gia tăng thì sự đa dạng dân số sẽ giảm và ngược lại. Áp lực chọn lọc là một công việc của toán tử lựa chọn. Áp lực chọn quá yếu có thể dẫn đến việc tìm kiếm không hiệu quả. Toán tử lựa chọn cũng như các toán tử khác ảnh hưởng đến tính đa dạng của quần thể. Làm sao để đạt được hiệu suất tốt trong khi duy trì tính đa dạng của quần thể càng lâu càng tốt. Đột biến rất quan trọng trong sự biến đổi của các nhiễm sắc thể, khi quần thể trở nên đồng nhất [23]. Sự đa dạng quần thể cũng có thể được duy trì bằng cách tăng kích thước quần thể hoặc do tỷ lệ đột biến lớn hơn, tuy nhiên, yếu tố hiệu suất nên được tính đến. Các kỹ thuật khác cũng được sử dụng. Cách tiếp cận phổ biến là tránh trùng lặp nhiễm sắc thể trong quần thể.

Tiêu chí dừng: Thuật toán di truyền là một quá trình ngẫu nhiên có thể chạy mãi mãi, nếu một tiêu chí dừng không được áp dụng. Vì vậy, để đảm bảo thuật toán di truyền sẽ kết thúc, người dùng thường phải định nghĩa tiêu chí dừng cho thuật toán. Một tiêu chí dừng đơn giản có thể là thời gian tính toán tối đa hoặc số lặp lặp lại tối đa, hoặc giá trị trung bình của độ thích nghi trên tất cả các nhiễm sắc thể của quần thể không thay đổi [26].

Có nhiều cách tiếp cận thuật toán di truyền có thể được tìm thấy trong một số tài liệu, một số trong đó bao gồm đa quần thể, di truyền động hoặc lai ghép với các phương pháp tiếp cận heuristic khác được biết đến [25]. Tuy nhiên, các nguyên tắc

chính của thuật toán di truyền vẫn giữ nguyên. Trong phần còn lại, chúng tôi sẽ trình bày thuật toán di truyền thiết kế cho bài toán thu gom chất thải rắn.

2.2. Thiết kế thuật toán di truyền cho bài toán thu gom chất thải tối ưu

2.2.1. Mã hóa cá thể

Hành trình của các xe như sau: các xe bắt đầu ở Depot đi lấy chất thải ở các Gather sites và đổ chất thải tại Transfer stations đến khi nào không còn Gather sites nào có chất thải thì quay trở về Depot. Như vậy điểm bắt đầu của hành trình là Depot và kết thúc hành trình cũng là Depot nhưng trước khi quay trở về Depot các xe phải đi đến các Transfer stations để đổ chất thải. Ký hiệu id của các node như sau:

‘1’: id của Depot.

‘2’: id của Transfer stations thứ nhất.

‘3’: id của Transfer stations thứ hai.

‘4’, ‘ N^+ ’: các id của Gather sites với số lượng $= N^+ - 4$.

Trong thuật toán di truyền mỗi mỗi nhiễm sắc thể của quần thể tương ứng là một lời giải của bài toán. Từ kịch bản trên một hành trình đi lấy hết chất thải ở tất cả Gather sites của 4 xe sẽ có cấu trúc như sau :



Hình 2.2: Cấu trúc nhiễm sắc thể

Ví dụ của một nhiễm sắc thể như sau:

{ 11: [1.0, 18.0, 42.0, 38.0, 5.0, 7.0, 9.0, 13.0, 17.0, 20.0, 21.0, 34.0, 33.0, 32.0, 24.0, 37.0, 8.0, 30.0, 31.0, 3.0], 12: [1.0, 19.0, 39.0, 15.0, 14.0, 11.0, 2.0,

1.0], 21: [3.0, 26.0, 36.0, 25.0, 28.0, 27.0, 23.0, 41.0, 40.0, 3.0, 1.0], 14: [1.0, 35.0, 4.0, 16.0, 12.0, 2.0, 1.0], 13: [1.0, 22.0, 29.0, 10.0, 6.0, 2.0, 1.0] }

Trong đó:

- 11: hành trình thứ nhất của xe thứ nhất;
- 12: hành trình thứ nhất của xe thứ hai;
- 13: hành trình thứ nhất của xe thứ ba;
- 14: hành trình thứ nhất của xe thứ tư;
- 21: hành trình thứ hai của xe thứ nhất.

2.2.2. Hàm Fitness

Độ thích nghi là khả năng thích nghi của mỗi nhiễm sắc thể (giải pháp) đối với môi trường (bài toán). Việc xây dựng độ thích nghi là một bước quan trọng trong thuật toán di truyền. Để đánh giá được độ thích nghi của các nhiễm sắc thể thuật toán di truyền sử dụng một hàm đo gọi là hàm Fitness.

Hàm lượng giá hay hàm thích nghi là hàm dùng để đánh giá độ tốt của một lời giải hoặc nhiễm sắc thể. Hàm Fitness nhận vào một tham số là xâu mã hóa của một nhiễm sắc thể và trả ra một số thực. Tùy theo giá trị của số thực này mà ta biết độ tốt của cá thể đó (chẳng hạn với bài toán tìm cực đại thì giá trị 12 trả ra càng lớn thì cá thể càng tốt, và ngược lại, với bài toán tìm cực tiểu thì giá trị trả ra càng nhỏ thì cá thể càng tốt).

Bài toán đặt ra là tối ưu thời gian thu gom chất thải tại thành phố A. Vì vậy sẽ có hàm Fitness sẽ là tổng thời gian đi thu gom của cả 4 xe trong hành trình đi tới các Gather sites của từng xe và được tính như sau:

$$\min \sum_{k \in V} \sum_{i, j \in \mathbb{Z}, N^+} t_{ij}(k) x_j^i(k)$$

Trong đó: $t_{ij}(k)$ là thời gian đi từ node i đến node j của xe k.

$x_j^i(k)$ Đánh dấu cung giữa hai node i và j:

$x_j^i(k)=1$ nếu xe k đi qua node i và node j.

$x_j^i(k)=0$ nếu xe k không đi qua node i và j.

2.2.3. Chọn lọc

Quá trình chọn lọc những nhiễm sắc thể từ quần thể hiện tại để tạo ra thế hệ sau của nó. Trong quá trình này diễn ra sự đào thải những nhiễm sắc thể xấu chỉ giữ lại những nhiễm sắc thể tốt. Những nhiễm sắc thể có độ thích nghi lớn hơn hoặc bằng với độ thích nghi tiêu chuẩn sẽ được giữ lại và độ thích nghi của các nhiễm sắc thể trong quần thể sẽ hoàn thiện hơn sau nhiều thế hệ. Để cho đơn giản chúng ta thường sắp xếp độ thích nghi của các cá thể theo thứ tự nhiễm sắc thể. Các bước cụ thể được mô tả như sau:

- Tính giá trị Fitness của từng nhiễm sắc thể trong quần thể hiện hành
- Sắp xếp nhiễm sắc thể trong quần thể theo thứ tự giá trị Fitness giảm dần
- Loại bỏ 50% nhiễm sắc thể ở cuối dãy. Giữ lại 50% nhiễm sắc thể có giá trị Fitness tốt nhất.

Cách chọn lọc này thích nghi với quy luật tự nhiên, di truyền các tính trạng tốt cho các thế hệ về sau.

Ví dụ:

Quần thể khởi tạo 4 nhiễm sắc thể và sắp xếp theo giá trị fitness của mỗi nhiễm sắc thể như sau:

Trip 1: { 11: [1.0, 18.0, 17.0, 20.0, 21.0, 16.0, 4.0, 25.0, 41.0, 40.0, 29.0, 42.0, 38.0, 39.0, 35.0, 34.0, 33.0, 31.0, 30.0, 3.0], 12: [1.0, 19.0, 15.0, 14.0, 11.0, 10.0, 3.0, 1.0], 21: [3.0, 24.0, 28.0, 27.0, 26.0, 32.0, 37.0, 36.0, 22.0, 3.0, 1.0], 14: [1.0, 8.0, 13.0,

6.0, 23.0, 3.0, 1.0], 13: [1.0, 5.0, 12.0, 7.0, 9.0, 3.0, 1.0]}, Fitness1 = 10.9174242429

Trip 2: {11: [1.0, 20.0, 33.0, 21.0, 7.0, 6.0, 4.0, 39.0, 40.0, 22.0, 18.0, 10.0, 32.0, 23.0, 11.0, 35.0, 5.0, 9.0, 15.0, 2.0], 12: [1.0, 8.0, 37.0, 36.0, 14.0, 24.0, 3.0, 1.0], 21: [2.0, 28.0, 30.0, 29.0, 27.0, 31.0, 34.0, 38.0, 26.0, 3.0, 1.0], 14: [1.0, 25.0, 17.0, 41.0, 13.0, 3.0, 1.0], 13: [1.0, 42.0, 16.0, 19.0, 12.0, 2.0, 1.0]}, Fitness2 = 11.2744515173

Trip 3: {11: [1.0, 17.0, 33.0, 34.0, 27.0, 42.0, 21.0, 41.0, 23.0, 18.0, 13.0, 19.0, 38.0, 39.0, 28.0, 7.0, 6.0, 8.0, 20.0, 3.0], 12: [1.0, 5.0, 22.0, 30.0, 24.0, 36.0, 2.0, 1.0], 21: [3.0, 26.0, 31.0, 37.0, 10.0, 35.0, 14.0, 15.0, 4.0, 3.0, 1.0], 14: [1.0, 29.0, 9.0, 25.0, 40.0, 2.0, 1.0], 13: [1.0, 16.0, 12.0, 32.0, 11.0, 3.0, 1.0]}, Fitness3 = 11.2777541689

Trip 4: {11: [1.0, 39.0, 18.0, 37.0, 17.0, 30.0, 23.0, 12.0, 38.0, 35.0, 9.0, 13.0, 15.0, 34.0, 19.0, 31.0, 20.0, 26.0, 10.0, 2.0], 12: [1.0, 24.0, 41.0, 6.0, 29.0, 22.0, 3.0, 1.0], 21: [2.0, 36.0, 8.0, 32.0, 27.0, 25.0, 4.0, 40.0, 33.0, 3.0, 1.0], 14: [1.0, 14.0, 21.0, 16.0, 42.0, 2.0, 1.0], 13: [1.0, 7.0, 11.0, 28.0, 5.0, 2.0, 1.0]}, Fitness4 = 11.409784405

Sau đó chọn 50% nhiễm sắc thể tốt ở đầu dãy ta được 50% quần thể mới:

Trip 1: {11: [1.0, 18.0, 17.0, 20.0, 21.0, 16.0, 4.0, 25.0, 41.0, 40.0, 29.0, 42.0, 38.0, 39.0, 35.0, 34.0, 33.0, 31.0, 30.0, 3.0], 12: [1.0, 19.0, 15.0, 14.0, 11.0, 10.0, 3.0, 1.0], 21: [3.0, 24.0, 28.0, 27.0, 26.0, 32.0, 37.0, 36.0, 22.0, 3.0, 1.0], 14: [1.0, 8.0, 13.0, 6.0, 23.0, 3.0, 1.0], 13: [1.0, 5.0, 12.0, 7.0, 9.0, 3.0, 1.0]}, Fitness1 = 10.9174242429

Trip 2: {11: [1.0, 20.0, 33.0, 21.0, 7.0, 6.0, 4.0, 39.0, 40.0, 22.0, 18.0, 10.0, 32.0, 23.0, 11.0, 35.0, 5.0, 9.0, 15.0, 2.0], 12: [1.0, 8.0, 37.0, 36.0, 14.0, 24.0, 3.0, 1.0],

21: [2.0, 28.0, 30.0, 29.0, 27.0, 31.0, 34.0, 38.0, 26.0, 3.0, 1.0], 14: [1.0, 25.0, 17.0, 41.0, 13.0, 3.0, 1.0], 13: [1.0, 42.0, 16.0, 19.0, 12.0, 2.0, 1.0]}, Fitness2 = 11.2744515173

2.2.4. Lai ghép

Toán tử lai ghép được thực hiện để tìm ra không gian giải pháp mới bằng cách kết hợp của chuỗi giữa các cặp cha mẹ được lựa chọn. Trong toán tử này, mỗi phân đoạn của con cái được lựa chọn một cách ngẫu nhiên với xác suất xác định giữa các phân đoạn tương ứng của cha mẹ.

Phép lai ghép được sử dụng trong bài toán của chúng ta là kỹ thuật chéo cạnh Edge Recombination [27]

Kỹ thuật như sau:

1. X = Node đầu tiên ngẫu nhiên của cha hoặc mẹ.

2. Lặp cho tới khi độ dài của nhiễm sắc thể CHILD đầy, bắt đầu lặp :

- Thêm X tới CHILD

- Xóa X khỏi các danh sách hàng xóm

Nếu danh sách hàng xóm của X rỗng:

- Z = node ngẫu nhiên mà chưa có trong CHILD

nếu không:

- Xác định hàng xóm trong các hàng xóm của X , sao cho hàng xóm này

có ít hàng xóm nhất (*)

- Nếu có nhiều hơn 1 hàng xóm như (*), chọn ngẫu nhiên một

- Z = node được chọn

$X = Z$

Ví dụ với cặp cha mẹ:

parent1 {11: [1.0, 20.0, 33.0, 21.0, 7.0, 6.0, 4.0, 39.0, 40.0, 22.0, 18.0, 10.0, 32.0, 23.0, 11.0, 35.0, 5.0, 9.0, 15.0, 2.0], 12: [1.0, 8.0, 37.0, 36.0, 14.0, 24.0, 3.0, 1.0], 21: [2.0, 28.0, 30.0, 29.0, 27.0, 31.0, 34.0, 38.0, 26.0, 3.0, 1.0], 14: [1.0, 25.0, 17.0, 41.0, 13.0, 3.0, 1.0], 13: [1.0, 42.0, 16.0, 19.0, 12.0, 2.0, 1.0]}

parent2 {11: [1.0, 18.0, 17.0, 20.0, 21.0, 16.0, 4.0, 25.0, 41.0, 40.0, 29.0, 42.0, 38.0, 39.0, 35.0, 34.0, 33.0, 31.0, 30.0, 3.0], 12: [1.0, 19.0, 15.0, 14.0, 11.0, 10.0, 3.0, 1.0], 21: [3.0, 24.0, 28.0, 27.0, 26.0, 32.0, 37.0, 36.0, 22.0, 3.0, 1.0], 14: [1.0, 8.0, 13.0, 6.0, 23.0, 3.0, 1.0], 13: [1.0, 5.0, 12.0, 7.0, 9.0, 3.0, 1.0]}

Để tiện phân biệt các Transfer Station và Depot của mỗi xe ta mã hóa lại như sau:

parent1: [11d, 20.0, 33.0, 21.0, 7.0, 6.0, 4.0, 39.0, 40.0, 22.0, 18.0, 10.0, 32.0, 23.0, 11.0, 35.0, 5.0, 9.0, 15.0, 11ts_2.0, 12d, 8.0, 37.0, 36.0, 14.0, 24.0, 12ts_3.0, 12d2, 13d, 42.0, 16.0, 19.0, 12.0, 13ts_2.0, 13d2, 14d, 25.0, 17.0, 41.0, 13.0, 14ts_3.0, 14d2, 21ts_2.0, 28.0, 30.0, 29.0, 27.0, 31.0, 34.0, 38.0, 26.0, 21ts2_3.0, 21d]

parent2: [11d, 18.0, 17.0, 20.0, 21.0, 16.0, 4.0, 25.0, 41.0, 40.0, 29.0, 42.0, 38.0, 39.0, 35.0, 34.0, 33.0, 31.0, 30.0, 11ts_3.0, 12d, 19.0, 15.0, 14.0, 11.0, 10.0, 12ts_3.0, 12d2, 13d, 5.0, 12.0, 7.0, 9.0, 13ts_3.0, 13d2, 14d, 8.0, 13.0, 6.0, 23.0, 14ts_3.0, 14d2, 21ts_3.0, 24.0, 28.0, 27.0, 26.0, 32.0, 37.0, 36.0, 22.0, 21ts2_3.0, 21d]

Tập danh sách hàng xóm của mỗi nút:

11d: 21d, 20.0, 18.0

4.0: 6.0, 39.0, 16.0, 25.0

5.0: 35.0, 9.0, 13d, 12.0

6.0: 7.0, 4.0, 13.0, 23.0

7.0: 21.0, 6.0, 12.0, 9.0

8.0: 12d, 37.0, 14d, 13.0

9.0: 5.0, 15.0, 7.0, 13ts_3.0

10.0: 18.0, 32.0, 11.0, 12ts_3.0

11.0: 23.0, 35.0, 14.0, 10.0

11ts_2.0: 15.0, 12d, 30.0

12.0: 19.0, 13ts_2.0, 5.0, 7.0

12d: 11ts_2.0, 8.0, 11ts_3.0, 19.0

12d2: 12ts_3.0, 13d

12ts_3.0: 24.0, 12d2, 10.0

13.0: 41.0, 14ts_3.0, 8.0, 6.0

13d: 12d2, 42.0, 5.0

13d2: 13ts_2.0, 14d, 13ts_3.0

13ts_2.0: 12.0, 13d2, 9.0

14.0: 36.0, 24.0, 15.0, 11.0

14d: 13d2, 25.0, 8.0

14d2: 14ts_3.0, 21ts_2.0, 21ts_3.0

14ts_3.0: 13.0, 14d2, 23.0

15.0: 9.0, 11ts_2.0, 19.0, 14.0

16.0: 42.0, 19.0, 21.0, 4.0

17.0: 25.0, 41.0, 18.0, 20.0

18.0: 22.0, 10.0, 11d, 17.0

19.0: 16.0, 12.0, 12d, 15.0

20.0: 11d, 33.0, 17.0, 21.0

21.0: 33.0, 7.0, 20.0, 16.0

21d: 21ts2_3.0, 11d

21ts2_3.0: 26.0, 21d, 22.0

21ts_2.0: 14d2, 28.0, 24.0

22.0: 40.0, 18.0, 36.0, 21ts2_3.0

23.0: 32.0, 11.0, 6.0, 14ts_3.0

24.0: 14.0, 12ts_3.0, 21ts_3.0, 28.0

25.0: 14d, 17.0, 4.0, 41.0

26.0: 38.0, 21ts2_3.0, 27.0, 32.0

27.0: 29.0, 31.0, 28.0, 26.0

28.0: 21ts_2.0, 30.0, 24.0, 27.0

29.0: 30.0, 27.0, 40.0, 42.0

30.0: 28.0, 29.0, 31.0, 11ts_3.0

31.0: 27.0, 34.0, 33.0, 30.0

32.0: 10.0, 23.0, 26.0, 37.0

33.0: 20.0, 21.0, 34.0, 31.0

34.0: 31.0, 38.0, 35.0, 33.0

35.0: 11.0, 5.0, 39.0, 34.0

36.0: 37.0, 14.0, 22.0

37.0: 8.0, 36.0, 32.0

38.0: 34.0, 26.0, 42.0, 39.0

39.0: 4.0, 40.0, 38.0, 35.0

40.0: 39.0, 22.0, 41.0, 29.0

41.0: 17.0, 13.0, 25.0, 40.0

42.0: 13d, 16.0, 29.0, 38.0

Đầu tiên, chọn node đầu tiên từ parent ngẫu nhiên, giả sử parent1 được node 11d

CHILD: 11d

Tiếp theo, sau khi chọn 11d, loại 11d khỏi tất cả danh sách hàng xóm, tiếp theo ta chọn B vì trong các hàng xóm của 11d thấy 20.0 có số lượng hàng xóm ít nhất

CHILD: 11d 20.0

Tiếp theo, sau khi loại 20.0 khỏi tất cả danh sách hàng xóm, thấy 21.0 có 4 hàng xóm, còn 33.0 và 17.0 cả 2 chỉ có 3 hàng xóm, vì vậy chọn ngẫu nhiên 1 trong 2:

CHILD: 11d 20.0 17.0

Tiếp theo, sau khi chọn 17.0 và loại 17.0 khỏi tất cả danh sách hàng xóm, thấy 25.0, 41.0, 18.0 cả 3 đều có 3 hàng xóm nên chọn ngẫu nhiên .

CHILD: 11d 20.0 17.0 25.0

Làm tương tự, cuối cùng ta được

CHILD: {11: [1.0, 20.0, 17.0, 25.0, 41.0, 13.0, 6.0, 4.0, 39.0, 38.0, 34.0, 33.0, 31.0, 30.0, 29.0, 27.0, 26.0, 32.0, 23.0, 2.0], 12: [1.0, 8.0, 37.0, 36.0, 14.0, 11.0, 3.0, 1.0], 21: [2.0, 28.0, 24.0, 19.0, 15.0, 9.0, 5.0, 12.0, 35.0, 3.0, 1.0], 14: [1.0, 40.0, 22.0, 18.0, 10.0, 3.0, 1.0], 13: [1.0, 42.0, 16.0, 21.0, 7.0, 2.0, 1.0]}

2.2.5. Đột biến

Toán tử đột biến sử dụng để ngăn chặn chuỗi hội tụ sớm và khám phá ra không gian giải pháp mới. Tuy nhiên, không giống như lai ghép, sự đột biến thường được dùng bằng cách sửa đổi gen trong phạm vi một nhiễm sắc thể.

Kỹ thuật đột biến được sử dụng là kỹ thuật hoán vị: Chọn 2 vị trí bất kỳ rồi hoán đổi giá trị của nút đó. Ví dụ:

Ví dụ :

Trước đột biến:

Child {11: [1.0, 20.0, 17.0, 25.0, 41.0, 13.0, 6.0, 4.0, 39.0, 38.0, 34.0, 33.0, 31.0, 30.0, 29.0, 27.0, 26.0, 32.0, 23.0, 2.0], 12: [1.0, 8.0, 37.0, 36.0, 14.0, 11.0, 3.0, 1.0], 21: [2.0, 28.0, 24.0, 19.0, 15.0, 9.0, 5.0, 12.0, 35.0, 3.0, 1.0], 14: [1.0, 40.0, 22.0, 18.0, 10.0, 3.0, 1.0], 13: [1.0, 42.0, 16.0, 21.0, 7.0, 2.0, 1.0]}

Chọn vị trí của đoạn là 3 và 8.

Sau đột biến:

Child: { 11: [1.0, 20.0, 4.0, 25.0, 41.0, 13.0, 6.0, 17.0, 39.0, 38.0, 34.0, 33.0, 31.0, 30.0, 29.0, 27.0, 26.0, 32.0, 23.0, 2.0], 12: [1.0, 8.0, 37.0, 19.0, 14.0, 11.0, 3.0, 1.0], 21: [2.0, 28.0, 24.0, 36.0, 15.0, 9.0, 5.0, 12.0, 35.0, 3.0, 1.0], 14: [1.0, 40.0, 22.0, 18.0, 10.0, 3.0, 1.0], 13: [1.0, 42.0, 16.0, 21.0, 7.0, 2.0, 1.0] }

2.2.6. Tìm kiếm địa phương với thuật toán Dijkstra

Thuật toán Dijkstra [d], mang tên của nhà khoa học máy tính người Hà Lan Edsger Dijkstra, là một thuật toán giải quyết bài toán đường đi ngắn nhất nguồn đơn trong một đồ thị có hướng không có cạnh mang trọng số âm.

Bài toán: Cho đơn đồ thị liên thông, có trọng số $G = (V, E)$. Tìm khoảng cách $d(u_0, v)$ từ một đỉnh u_0 cho trước đến một đỉnh v bất kỳ của G và tìm đường đi ngắn nhất từ u_0 đến v .

Phương pháp của thuật toán Dijkstra là: xác định tuần tự đỉnh có khoảng cách đến u_0 từ nhỏ đến lớn.

Trước tiên, đỉnh có khoảng cách đến u_0 nhỏ nhất chính là u_0 , với $d(u_0, u_0) = 0$. Trong các đỉnh $v \neq u_0$, tìm đỉnh có khoảng cách k_1 đến u_0 là nhỏ nhất. Đỉnh này phải là một trong các đỉnh kề với u_0 . Giả sử đó là u_1 :

$$d(u_0, u_1) = k_1.$$

Trong các đỉnh $v \neq u_0$ và $v \neq u_1$, tìm đỉnh có khoảng cách k_2 đến u_0 là nhỏ nhất. Đỉnh này phải là một trong các đỉnh kề với u_0 hoặc với u_1 . Giả sử đó là u_2 :
 $d(u_0, u_2) = k_2$.

Tiếp tục như trên, cho đến bao giờ tìm được khoảng cách từ u_0 đến mọi đỉnh v của G . Nếu $V = \{u_0, u_1, \dots, u_n\}$ thì:

$$0 = d(u_0, u_0) < d(u_0, u_1) < d(u_0, u_2) < \dots < d(u_0, u_n)$$

Bảng 2.1: Thuật toán Dijkstra cổ điển

Thuật toán Dijkstra cổ điển:

Input: $G=(V,E)$ là đơn đồ thị liên thông, $\{G$ có các đỉnh $a=u_0, u_1, \dots, u_n=z$ và trọng số $m(u_i, u_j)$, với $m(u_i, u_j) = \infty$ nếu (u_i, u_j) không là một cạnh trong $G\}$

Output: Đường đi ngắn nhất từ đỉnh u_0 tới đỉnh z bất kỳ.

def Route() :

 for $i:= 1$ to n

$L(u_i):= \infty$

$L(a):= 0$

$S:= V \setminus \{a\}$

$u:= a$

 while $S \neq \emptyset$

 begin

 for tất cả các đỉnh v thuộc S

 if $L(u) + m(u,v) < L(v)$ then $L(v) := L(u) + m(u,v)$

$u:=$ đỉnh thuộc S có nhãn $L(u)$ nhỏ nhất

$\{L(u):$ độ dài đường đi ngắn nhất từ a đến $u\}$

$S:= S \setminus \{u\}$

 end while

end def

Tuy nhiên nếu sử dụng thuật toán Dijkstra cổ điển vào mô hình bài toán sẽ không giải quyết được hết các ràng buộc và điều kiện trong mô hình. Vì vậy, thuật toán Dijkstra yêu cầu phải cải tiến.

Bảng 2.2: Thuật toán Dijkstra cải tiến

Thuật toán Dijkstra cải tiến:

Input: Sức chứa chất thải ở các Gather sites: Z

Danh sách id của các node: N

Danh sách id của các xe: V

Sức chứa của các xe: C

Output: Hành trình đi của các xe và tổng thời gian và khoảng cách.

del Route():

Khởi tạo các xe bắt đầu ở Depot và lượng chất thải hiện tại của các xe ban đầu bằng 0, $Q(i) = 0$, $i = \overline{1, K}$

while ($\sum Z(i) > 0$):

For i in V :

while ($((C(i) - Q(i)) \geq 0.4 \ \& \ !(Z(i) = 0))$):

a = tìm node của xe i

Math = false

Neighbor(a) # **function 1**

For $b = 1$ to $N(a)$ do

Constraint (a, b, i) # **function 2**

if(true) then

Node(i) = b

$Q(i) = Q[i][b]$

Math(b)

Break

Math = true

```

end if

end for

if(Math = false)
Node(i) = Transfer station;
break
end if
end while

Print(result) # function 3

end for

end while
print hành trình xe
function 1
#tìm danh sách các Gather sites N(a) kề với node hiện tại theo thứ tự từ gần nhất
đến xa nhất.
def Neighbor(a1, gather):
    N(a) = { }
    For i in V:
        if (ia!= a):
            Distance(i,a)
            N(a) = N(a) ∪ {i}
        #Sắp xếp theo thứ tự khoảng cách tăng dần
        Sort(N(a))
function 2
def Constraint (a,b,i): # giải quyết các ràng buộc trong mô hình
    #Lượng chất thải hiện tại của xe I sau khi rời khỏi node a: Q[i][a]

    # nhãn của các node: L

    # sức chứa của Gather sites: G

    while (Q[i][a] != C(i) | (C(i)- Q[i][a]) > Z(b)):

        Q[i][b] = Q[i][a] + Z(b)

        if ((L(b) < L(a) + distance[a][b]) & (Q[i][b] <= C(i)) & ((Q[i][b] -
        Q[i][a]) <= Z(b))):

            return true

        else:

```

```

                                return false
                        return false
function 3
def Print(result):
    for i in len(Math):
        route += node[i] + "→"
    return result

```

2.2.7. Chi tiết thuật toán

Sau đây trình bày thuật toán di truyền ứng dụng cho bài toán tối ưu thu gom chất thải với dữ liệu ở thành phố A. Thuật toán như sau:

Input: Sức chứa chất thải ở các Gather sites: Z

Danh sách id của các node: N

Danh sách id của các xe: V

Sức chứa của các xe: C

Bản đồ vị trí của tất cả các node

Số lượng nhiễm sắc thể trong quần thể ban đầu (P)

Số lần lặp tối đa

Output: Hành trình đi của các xe và tổng thời gian và khoảng cách.

1, Khởi tạo ngẫu nhiên P nhiễm sắc thể, trong đó mỗi nhiễm sắc thể được sinh ra bởi thuật toán Dijkstra cải tiến Chromosome [i] = Route()

2.a Vòng lặp xử lý

3. Với mỗi nhiễm sắc thể trong quần thể $i = 1$ đến p

4. Tính toán giá trị Fitness của nhiễm sắc thể đó

a: Tính thời gian và khoảng cách đi được của tất cả các xe trong hành trình theo công thức (A1) – (A3)

b: Tính giá trị Fitness của nhiễm sắc thể i theo giá trị hàm mục tiêu được cho bởi (A0)

c: Cập nhật pBest và gBest theo quy tắc sau:

Nếu $pBest[i] < Fitness(i)$ thì $pBest[i] = Fitness(i)$,

Nếu $gBest < pBest[i]$ thì $gBest = pBest[i]$.

2.b Kết thúc vòng lặp

5.a: Vòng lặp bắt đầu từ $i = 1$ đến $P/2$

Giữ lại 50% best chromosomes Best_Fit (1, $P/2$)

5.b. Vòng lặp kết thúc

6.a: Vòng lặp bắt đầu $i = P/2 + 1$ đến $3P/4$

Giữ ngẫu nhiên 25% các nhiễm sắc thể trong quần thể ban đầu, nhiễm sắc thể $[i]$
 $= Route()$

6.b. Vòng lặp kết thúc

7: Vòng lặp bắt đầu $i = 3P/4 + 1$ tới P

Tạo ra 25% nhiễm sắc thể cuối cùng từ các nhiễm sắc thể tốt nhất ở thế hệ cha mẹ bằng cách sử dụng phương pháp lai ghép Edge Recombination

Chromosome $[i] = Edge_Recombination(Random(Chromosome [1, $P/2$]),
 $Random(Chromosome [1, $P/2$]))$$

8: Đột biến nhiễm sắc thể bằng cách hoán vị.

Mutation (Chromosome $[i]$)

9: Kiểm tra nếu nhiễm sắc thể không thỏa mãn các ràng buộc của mô hình thì ta thay thế nhiễm sắc thể đó bằng nhiễm sắc thể mới bằng thuật toán Dijkstra cải tiến

If (Constraints satisfy (Chromosome [i]) = 0)

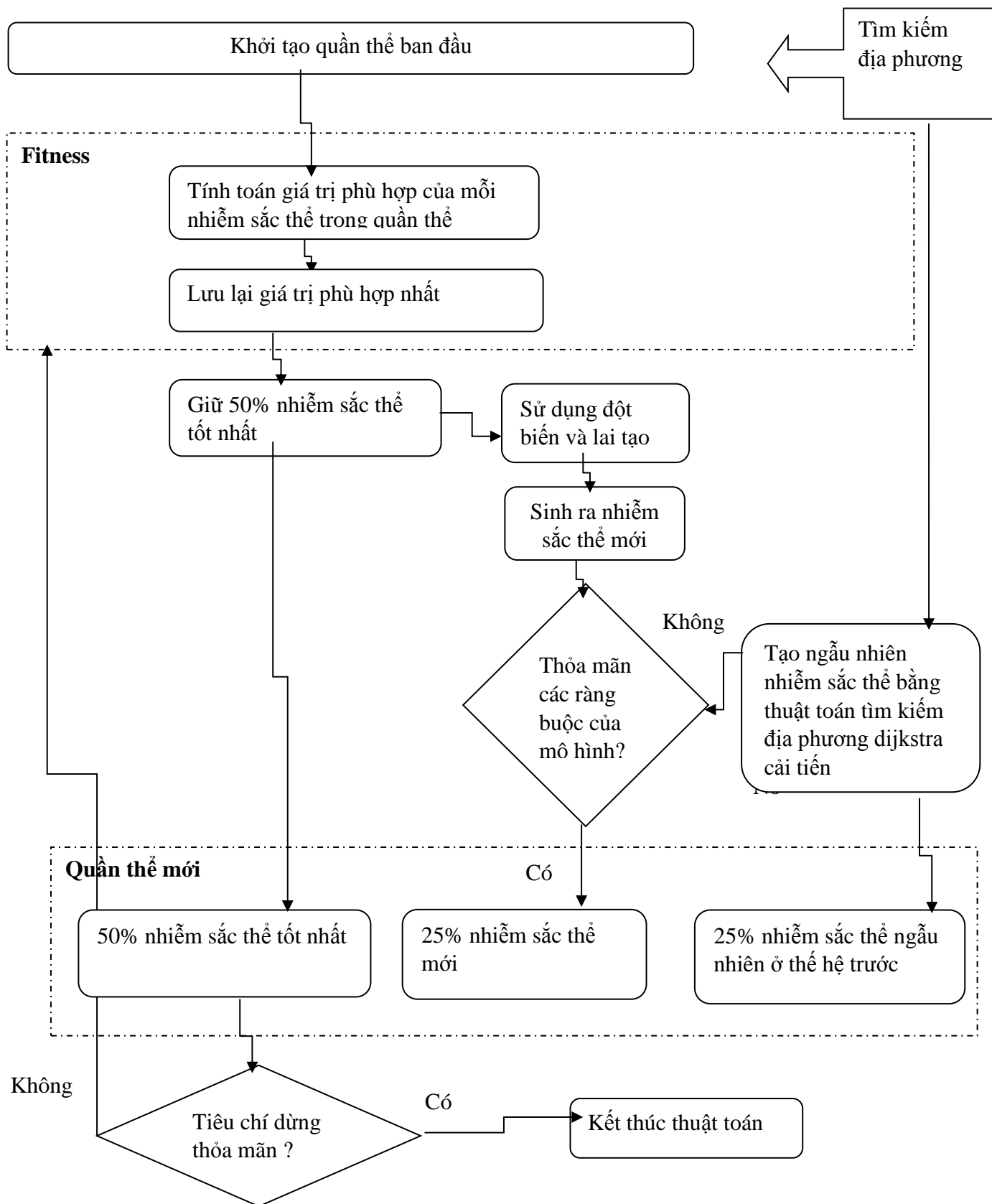
Chromosome [i] = Route ()

End if

7b. Vòng lặp kết thúc

Trở lại 2a.

Thoát khi điều kiện dừng thỏa mãn.



Hình 2.3: Sơ đồ thực hiện thuật toán

2.3. So sánh Dijkstra và GA

Thuật toán định tuyến Dijkstra cũng thông dụng để sử dụng tìm kiếm các giải pháp tối ưu như trong phần mềm ArcGIS để định tuyến đường và mục tiêu là giảm thiểu tổng khoảng cách thu thập. Trong luận văn thuật toán Dijkstra được sử dụng với 2 mục đích. Mục đích thứ nhất là để dùng để tạo ra các giải pháp cho thuật toán di truyền cho bài toán của chúng ta, mục đích thứ hai là dùng để đánh giá, so sánh với kết quả với thuật toán được di truyền.

Khởi tạo quần thể ban đầu đặc biệt quan trọng với thuật toán di truyền. Việc khởi tạo quần thể, tạo ra các nhiễm sắc thể - *hành trình ngẫu nhiên* đối với kịch bản và ràng buộc của thành phố A việc là tương đối phức tạp. Thuật toán Dijkstra cải tiến ở mục 2.2.6 trình bày bên trên giải quyết kịch bản và ràng buộc của bài toán ở thành phố này. Do đó, trong bước khởi tạo quần thể chúng tôi đã sử dụng và sửa đổi thuật toán Dijkstra cải tiến một chút mục đích để tạo ra các hành trình ngẫu nhiên và khởi tạo tập nhiễm sắc thể cho quần thể trong thuật toán di truyền của chúng ta. Việc này được xử lý như sau: Thay vì chọn Gather site có trọng số từ Gather site trước đó là nhỏ nhất theo tư tưởng của thuật toán Dijkstra cổ điển, việc này sẽ được thay thế bằng cách chọn ngẫu nhiên Gather site bất kỳ trong tập hàng xóm của Gather site trước đó, việc này được thực hiện cho tất cả các node trong hành trình đi. Việc thay đổi một chút như vậy thôi là được một nhiễm sắc thể có hành trình ngẫu nhiên và thỏa mãn yêu cầu bài toán.

Với mục đích giúp đánh thuật toán di truyền được đề xuất. Bởi vì thuật toán Dijkstra là thuật toán tìm kiếm địa phương, do vậy chúng ta hi vọng thuật toán di truyền kỳ vọng sẽ tìm ra được các kết quả tốt hơn so với phương pháp dùng thuật toán Dijkstra. Việc này sẽ được thực nghiệm và sáng tỏ ở chương 3.

2.4. Tổng kết chương

Chương này đã trình bày tổng quan lý thuyết về thuật toán di truyền từ đó thiết kế thuật toán di truyền cho bài toán tối ưu thu gom chất thải rắn qua việc: trình bày cách mã hóa bài toán, xây dựng hàm Fitness, chọn lựa kỹ thuật khởi tạo quần thể, chọn lọc di truyền, lai ghép di truyền, đột biến di truyền. Cùng việc trình bày

thuật toán Dijkstra, vai trò của thuật toán Dijkstra trong việc thiết kế và so sánh với thuật toán di truyền. Chương tiếp theo sẽ trình bày kết quả thực nghiệm triển khai tại thành phố Sfax, Tunisia.

Chương 3 – ỨNG DỤNG THUẬT TOÁN DI TRUYỀN CHO BÀI TOÁN TỐI ƯU THU GOM CHẤT THẢI RẮN ĐÔ THỊ TẠI THÀNH PHỐ SFAX, TUNISIA

3.1. Giới thiệu về khu vực nghiên cứu

Tunisia, tên chính thức Cộng hòa Tunisia, là một quốc gia ở Bắc Phi. Nước này giáp với Algérie ở phía tây, Libya ở phía đông nam, và Biển Địa Trung Hải ở phía bắc và phía đông. Tunisia có diện tích 165,000 km² với dân số ước tính chỉ hơn 10.3 triệu người [15]. Tên nước xuất phát từ tên thủ đô Tunis nằm ở phía đông bắc. Tunisia có 10.778 hộ gia đình và lượng chất thải thải ra là 2.423 triệu tấn (2012). Lượng chất thải bình quân tính trên đầu người là 0.815 kg/ngày ở khu vực đô thị. Sự tăng trưởng lượng chất thải là 2.5 % mỗi năm.

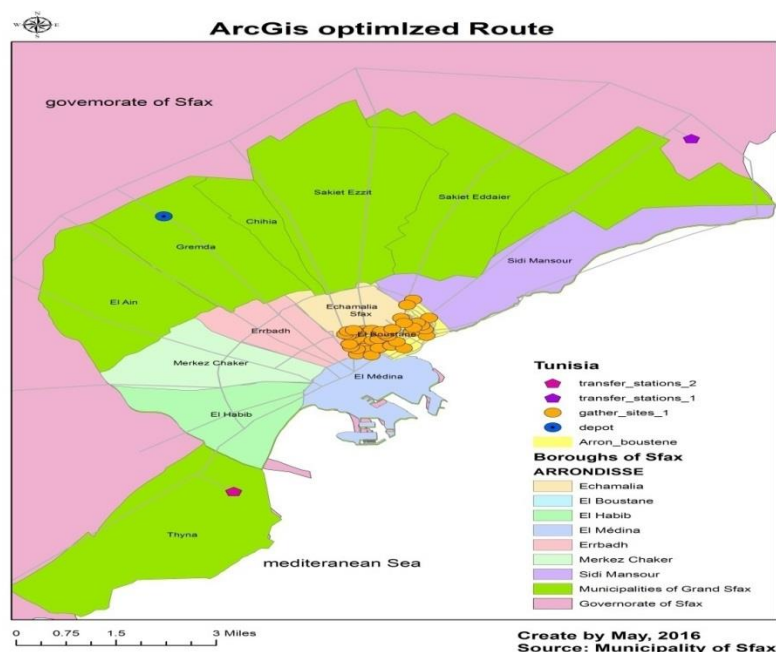
Sfax là thành phố có lượng dân cư đông thứ hai và là một trong những thành phố ô nhiễm nhất ở Tunisia [15]. Lượng chất thải trung bình tính theo đầu người mỗi ngày là 0.702 kg/hab/day. Việc quản lý chất thải và phân chia nhiệm vụ quản lý chất thải bao gồm hai giai đoạn. Ban đầu các xe đi thu gom chất thải từ các nơi chứa chất thải trong đô thị được quản lý chung bởi Sfax và lượng chất thải thu gom được sẽ được chở tới các trạm trung chuyển. Giai đoạn thứ hai là chất thải được vận chuyển từ trạm trung chuyển đến bãi chất thải do ANGED quản lý. Đề tài này tập trung vào giai đoạn đầu tiên. Sfax được chia làm bảy quận. Mỗi quận có một trợ lý giám đốc (chịu trách nhiệm) và những người lao động. Đề tài này nghiên cứu và giải quyết bài toán tối ưu chất thải rắn ở quận 'elboustene' có bình quân lượng chất thải theo đầu người là 0.944kg, lớn hơn cả lượng chất thải bình quân đầu người của thành phố. Diện tích của quận 'elboustene' là 315 Hectares, bao gồm 17446 hộ, 217 cơ sở công nghiệp và nhiều tổ chức về tài chính, y tế và các tổ chức công cộng. Chất thải rắn ở quận 'elboustene' được thu thập bởi khu vực tư nhân.

3.2. Kịch bản thu gom chất thải rắn

Chất thải rắn đô thị ở Sfax xuất bao gồm hai loại. Loại thứ nhất là chất thải sinh hoạt từ nhà ở, khách sạn, chất thải thải đường phố. Loại thứ hai là chất thải thải đồng dạng từ chợ, văn phòng, nhà hàng, bệnh viện, nhà tù, trại lính, nhà kho, chuồng trại chăn nuôi. Các nơi này được gọi là các Gather site.

Kịch bản thu gom chất thải rắn ở Sfax liên quan đến việc sử dụng các loại xe không đồng nhất. Một số xe bán tự động như Agricultural tractor, Dumper truck và xe tự động như Compactor vehicles. Có 4 xe sẽ thực hiện nhiệm vụ đi thu gom chất thải, trong đó có 2 xe Agricultural tractor, 1 xe Dumper truck và 1 Compactor vehicles. Mỗi loại xe khác nhau về sức chứa chất thải. Xe Agricultural tractor có sức chứa là 1.6 tấn. Xe Dumper truck là 2.3 tấn, trong khi đó Compactor vehicles sức chứa lên tới 7.4 tấn. Sức chứa chất thải ở mỗi Gather sites là 0.4 tấn và thời gian làm dịch vụ ở mỗi Gather sites là 15 phút. Các xe kết thúc hành trình thu gom chất thải của mình khi tất cả chất thải ở các Gather sites được lấy đi hết.

Tại Sfax có 1 Depot chứa các xe đi làm nhiệm vụ, 39 Gather sites và 2 Transfer stations.



Hình 3.1: Bản đồ Tunisia

Các xe bắt đầu quá trình đi thu gom chất thải bắt đầu từ Depot. Sau đó sẽ đi đến các Gather sites để lấy chất thải đến khi nào lượng chất thải được lấy bằng với sức chứa của các xe hoặc sức chứa còn lại trên xe nhỏ hơn lượng chất thải tại Gather sites thì xe sẽ đi đến các Transfer stations để đổ chất thải rồi quay lại lấy chất thải ở các Gather sites còn lại. Cứ lặp lại quá trình như vậy cho mỗi xe đến khi nào không còn chất thải ở Gather sites thì xe sẽ quay trở lại Depot và kết thúc hành trình. Lượng chất thải ở mỗi Gather sites không được lấy một phần mà phải được lấy đi hết trong một lần.

Thời gian làm việc của mỗi xe phụ thuộc vào từng quận. Đối với quận ‘elboustene’ ở Sfax thì ca làm việc kéo dài từ 6 AM đến 1 PM.

3.3. Mô tả dữ liệu thu thập và yêu cầu

Thứ tự của các xe là cố định và sức chứa chất thải của mỗi xe là một hằng số. Lượng chất thải hiện tại của xe là một số biến thiên. Ban đầu, khi các xe xuất phát ở Depot, lượng chất thải hiện tại của mỗi xe sau khi rời khỏi Depot bằng 0. Nếu lượng chất thải hiện tại trên xe mà bằng với sức chứa của xe hoặc sức chứa còn lại trên xe nhỏ hơn lượng chất thải tại Gather sites thì xe sẽ đi đến Transfer stations gần nhất để đổ chất thải. Và lượng chất thải hiện tại của xe sau khi rời khỏi Transfer stations lại bằng 0.

Bảng 3.1: Sức chứa chất thải của mỗi xe

Số thứ tự của xe	Tên xe	Sức chứa chất thải
1	Compactor vehicle	7.4 tấn
2	Dumper truck	2.3 tấn
3	Agricultural tractor 1	1.6 tấn
4	Agricultural tractor 2	1.6 tấn

Số lượng các node là cố định. Sức chứa chất thải tại mỗi Gather sites ban đầu đều là hằng số. Trong quá trình đi thu gom chất thải sức chứa tại mỗi Gather sites sẽ bằng 0 nếu Gather sites đó được xe đến lấy chất thải. Lượng chất thải tại Depot và

Transfer stations không sử dụng trong quá trình tính toán và được thiết lập ban đầu bằng 0. Sức chứa tại các Gather sites ban đầu đều bằng nhau và bằng 0.4 tấn.

Bảng 3.2: Sức chứa chất thải ban đầu tại tất cả các node

Tên node	Sức chứa chất thải	Số lượng
Depot	0	1
Transfer station 1	0	1
Transfer station 2	0	1

3.4. Mô hình thu gom chất thải rắn đô thị tại Sfax

Dựa trên những điểm nổi bật và được thúc đẩy bởi những ý tưởng của [19] và [21], một mô hình VR giải quyết vấn đề cho bài toán của quận ‘elboustene’ được xây dựng với các giả định sau:

1. Chỉ thu gom chất thải ở quận ‘Elboustene’
2. Khoảng cách giữa các node và thời gian đi giữa các node là biết.
3. Số lượng các node và vị trí của chúng trên bản đồ là cố định.
4. Thời gian làm việc là cố định. Các xe chỉ làm trong ca ngày.
5. Thời gian load và unload của mỗi xe là bằng nhau và tải toàn phần.
6. Sức chứa chất thải của mỗi loại xe không bằng nhau.
7. Tất cả các xe cần thu gom hết chất thải trước khi quay trở về Depot, vì vậy thời gian thu gom chất thải là mục tiêu chính.

Các ký hiệu và định nghĩa:

Bảng 3.3: Ký hiệu và định nghĩa

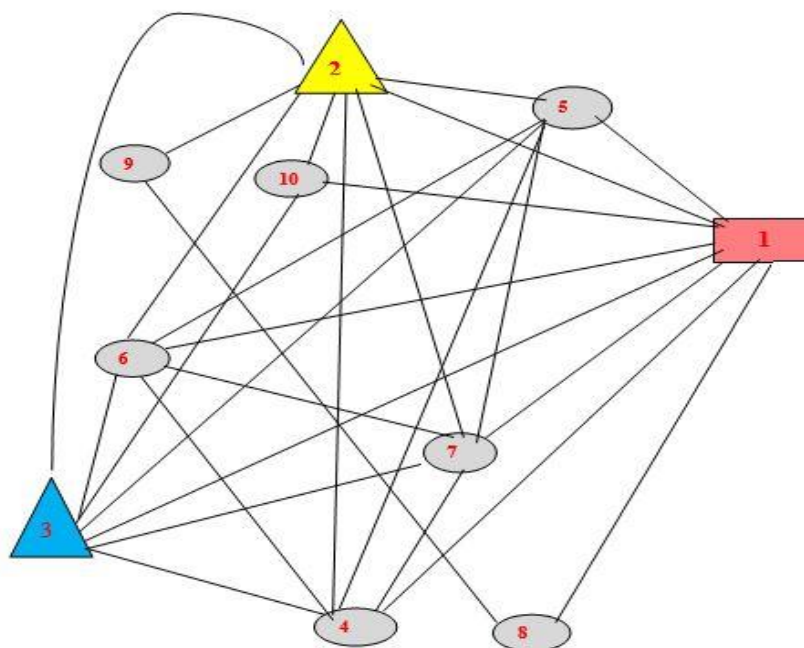
Ký hiệu	Định nghĩa và giải thích
$\bar{N} = \{1, 2, 3, 4, \dots, N^+\}$	Danh sách id của các node ‘1’: id của Depot. ‘2’: id của Transfer stations thứ nhất. ‘3’: id của Transfer stations thứ hai. ‘4’, ‘ N^+ ’: các id của Gather sites với số lượng $= N^+ - 4$.
$Z = \{Z_1, Z_2, Z_3, Z_4, \dots, Z_{N^+}\}$	Sức chứa chất thải của tất cả các node : $Z_1 = 0$: sức chứa chất thải ở Depot. Z_2 : sức chứa của Transfer stations thứ nhất. Z_3 : sức chứa của Transfer stations thứ hai. Z_4, \dots, Z_{N^+} : sức chứa của các Gather site.
$V = \{1, 2, 3, 4\}$	Danh sách các id của các xe: ‘1’: id của xe thứ nhất. ‘2’: id của xe thứ hai. ‘3’: id của xe thứ ba. ‘4’: id của xe thứ tư.
$C = \{C_1, C_2, C_3, C_4\}$	Sức chứa chất thải của các xe, trong đó: C_1 : sức chứa của thứ nhất. C_2 : sức chứa của xe thứ hai. C_3, C_4 : sức chứa của xe thứ ba, thứ 4
$Q = \{Q_k^i\}$	k: id của xe, i: id của node. Q_k^i : lượng chất thải hiện tại của xe k sau khi rời khỏi node i.
$x_j^i(k)$	Đánh dấu cung giữa hai node i và j: $x_j^i(k)=1$ nếu xe k đi qua node i và node j. $x_j^i(k)=0$ nếu xe k không đi qua node i và j.
$t_{ij}(k)$	Thời gian đi từ node i đến node j của xe k.

Từ những ký hiệu và định nghĩa trên, mô hình VRP được xây dựng như sau:

Bảng 3.4: Mô hình cho bài toán thu gom chất thải

Hàm mục tiêu (A ₀)		$\min \sum_{k \in V} \sum_{i,j \in 2, N^+} t_{ij}(k) x_j^i(k)$	Tối thiểu thời gian đi thu gom chất thải.
Ràng buộc	A ₁	$\sum_{j=4}^{N^+} x_j^1(k) = 1 \quad (\forall k \in V)$	Các xe bắt đầu cuộc hành trình ở Depot.
	A ₂	$\sum_{i=2}^3 x_1^i(k) = 1 \quad (\forall k \in V)$	Hành trình của các xe kết thúc ở Depot.
	A ₃	$\sum_{i=1,2,3} \sum_{j \in \bar{N}} Q_k^i x_j^i(k) = 0 \quad (\forall k \in V)$	Lượng chất thải hiện tại của xe sau khi rời khỏi Depot, Transfer stations bằng 0.
	A ₄	$\sum_{i,j \in \bar{N}} Q_k^i x_j^i(k) \leq C_k \quad (\forall k \in V)$	Tổng lượng chất thải hiện tại của xe k sau khi rời khỏi node i phải bé hơn hoặc bằng sức chứa của xe k.
	A ₅	$(Q_k^j - Q_k^i) x_j^i(k) = Z_j \quad (\forall k \in V, \forall i, j \in \bar{N})$	Lượng chất thải lấy đi ở node j phải bằng sức chứa của node j.

Ví dụ minh họa: Một hệ thống thu gom chất thải gồm bộ (N^+, Z, V, Q) như **hình 3.2** bao gồm 1 Depot (id : 1), 2 Transfers station (id : 2) và (id : 3) , 7 Gather site (id từ 4 tới 10). Trọng số chất thải của mỗi node trong Bảng 3.5. Hệ thống có 4 xe đi thu gom chất thải bao gồm 2 Agricultural tractor (id: Ag1 và Ag2), 1 Dumper truck (id : D) và 1 Compactor vehicles (id : C). Sức chứa của mỗi xe trong Bảng 3.6. Khoảng cách các node trong Bảng 3.7.



Hình 3.2 : Ví dụ về hệ thống thu gom rác

Bảng 3.5: Lượng chất thải mỗi node (kg)

\tilde{N}	1	2	3	4	5	6	7	8	9	10
R	0	60000*	25000*	735	814	730	828	868	618	1020

*: Sức chứa chất thải tại Transfer station

Bảng 3.6: Sức chứa chất thải của mỗi xe (kg)

V	Ag1	Ag2	D	C
C	800	800	1000	1500

Bảng 3.7: Khoảng cách giữa các node (km)

N^+	1	2	4	5	6	3	7	8	9	10
1	0	60	67	150	40	160	60	25		34
2	60	0	55	75	203	20	30		21	22
4	67	55	0	40	79	82	52			
5	150	75	40	0	79	102	84			
6	40	203	79	79	0	45	44			
3	160	20	82	102	45	0	60			
7	60	30	52	84	44	60	0			
8	25							0	66	
9		21						66	0	
10	34	22				19				

Tất cả các xe xuất phát ở Depot, kết quả hành trình thứ nhất của các xe đến các node như Bảng 3.8. Kết quả này đã thoả mãn ràng buộc trong (A_1) .

Bước đầu tiên là tìm các Gather sites lân cận của mỗi node theo thứ tự gần tới xa nhất. Bước thứ 2 là xác định các ràng buộc về sức chứa chất thải của các xe trong hành trình thu thập chất thải tại các node từ đó quyết định được các Gather sites mà mỗi xe sẽ đi qua, dựa vào khoảng cách tới các Transfer stations quyết định về Transfer station nào để đổ chất thải sau đó về Depot.

Compactor vehicles bắt đầu hành trình từ Depot (theo ràng buộc A_1) với đang chứa lượng chất thải bằng 0 (theo ràng buộc A_3), sau đó đi tới node 8 ($Q^j = 828$) và node 9 ($Q^j = 618$). Compactor vehicles đang chứa 1446 kg chất và giá trị này nhỏ hơn sức chứa của xe đó (theo ràng buộc A_4), nhưng nó không thể chứa thêm rác có ở node 7 hoặc 10 bởi vì nếu thêm thì sức chứa hiện tại sẽ lớn hơn sức chứa của xe (theo ràng buộc A_5). Compactor vehicles sẽ về Transfer station 2 để đổ rác. Tiếp theo Compactor vehicles sẽ bắt đầu từ Transfer station đi tới node 10 có khối lượng rác 1020 kg để thu rác sau đó lại về Transfer station để đổ rác lần nữa. Tương tự xe Dumper truck lại cũng tiếp tục từ Transfer station đi tới node 7 có khối lượng rác 828 kg để thu rác sau đó lại về Transfer station để đổ rác lần nữa và kết thúc hành trình (Bảng 3.9).

Bảng 3.8: Kết quả hành trình thứ nhất

V	Ag2	D	Ag1		C	C	
node	4	5	6	7	8	9	10
Q^j	735	814	730	828	828	618	1020
Trạng thái	Đầy	Đầy	Đầy	828	Đầy	Đầy	1020

Bảng 3.9: Kết quả hành trình thứ 2

V				D			C
node	4	5	6	7	8	9	10
Q^j	0	0	0	828	0	0	1020
Trạng thái	Đầy	Đầy	Đầy	Đầy	Đầy	Đầy	Đầy

Sau khi thu thập hết rác ở tất cả các node và đổ rác ở Transfer station, các xe sẽ trở về Depot (theo ràng buộc A_2).

Ta được hành trình như sau:

Xe 1: 1 -> 6 -> 3 -> 1

Xe 2: 1 -> 4 -> 3 -> 1

Xe 3: 1 -> 5 -> 2 -> 7 -> 1

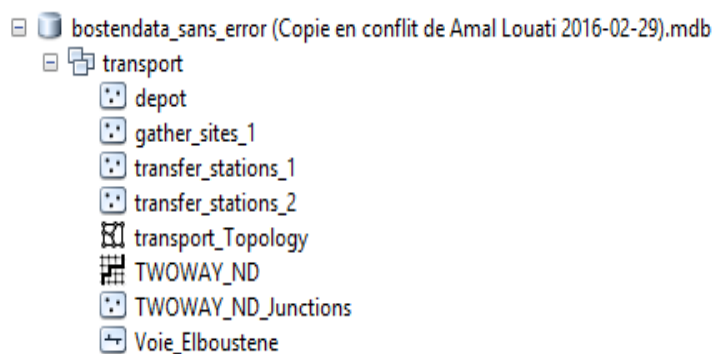
Xe 4: 1 -> 8 -> 9 -> 2 -> 10 -> 1

3.5. Môi trường thực nghiệm

Các thuật toán được cài đặt trên Python và thử nghiệm trên một máy tính với cấu hình: Intel® Core™ i5-7400 CPU@ 3.00GHz; 3000Mhz, 4 Core(s), 4 Logical Processor(s), RAM 8GB. Trong Thuật toán di truyền, số lượng tối đa các bước lặp là 1.000, kích thước quần thể là 50 số lần lặp là 1000. Thí nghiệm được tiến hành trên các số liệu tọa độ địa lý thực tế của thành phố Sfax, trong đó bao gồm 1 kho, 2 trạm chuyển giao, 1 bãi chất thải và 39 nơi đổ chất thải tập trung. Có 4 xe vận chuyển, trong đó có 1 Compactor vehicles vehicle, 1 xe Dumper truck và 2 xe Agricultural tractor.

Việc đo đạc và thể hiện kết quả trực quan lên bản đồ cho người dùng qua Phần mềm ArcGis là phần mềm ứng dụng công nghệ hệ thống thông tin địa lý của Viện nghiên cứu hệ thống môi trường [5], đó là một hệ thống phần mềm cung cấp một giải pháp tổng thể về hệ thống thông tin địa lý, bao gồm nhiều modul khác nhau. Trong đề tài này sử dụng modul Network Analyst trong ArcGis được sử dụng để thao tác tạo dữ liệu, sau đó là biên tập và thành lập bản đồ, cũng như là để phân tích bản đồ.





Dữ liệu tọa độ các node trên bản đồ được lưu trong cơ sở dữ liệu geodatabase mô tả mô hình dữ liệu đối tượng GIS. Mô tả như sau:



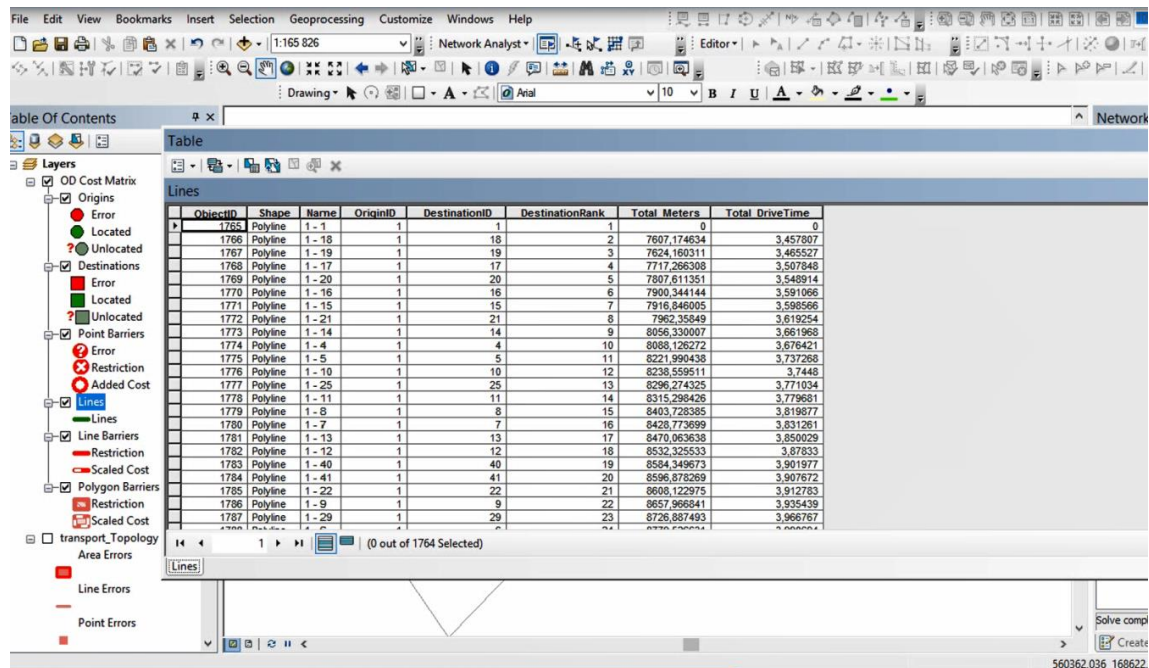
Hình 3.3: Dữ liệu trong ArcGIS

Trong đó, Depot, gather_sites_1, transfer_stations_1, transfer_stations_2 là dữ liệu địa lý của các Depot, Gather sites, Transfer station 1 và Transfer station 2 tương ứng. TWOWAY_ND, TWOWAY_ND_Junctions, Voie_Elboustene là bộ dữ liệu mạng về các tuyến đường, các điểm topo ở Sfax.

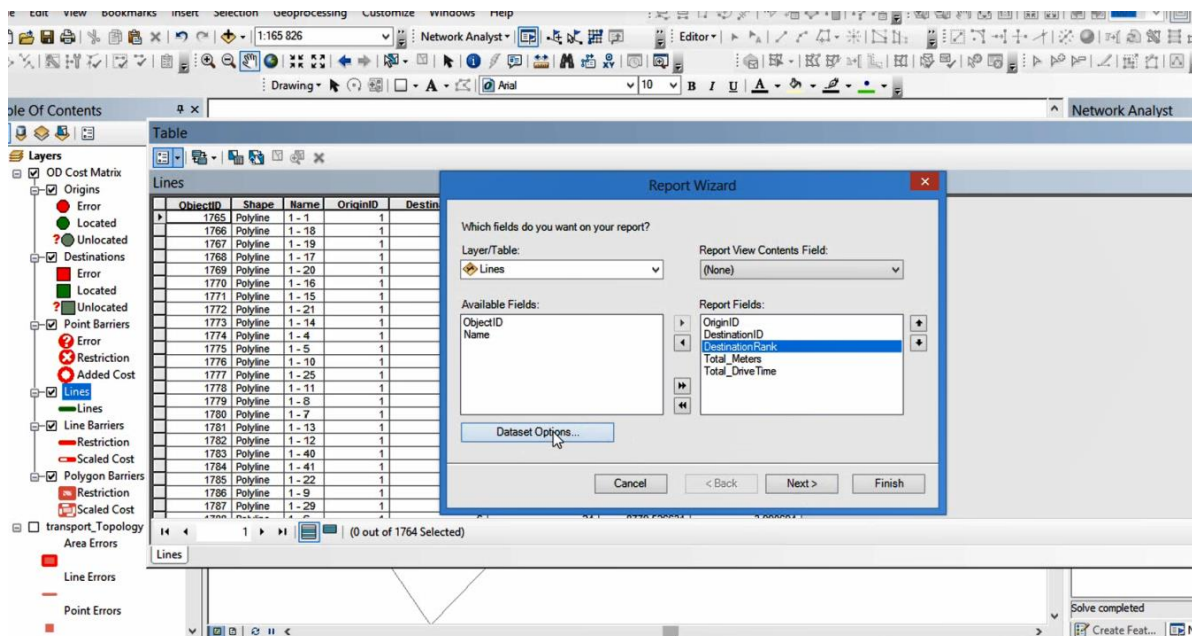
Bảng 3.10: Biểu tượng của các node trên ArcGIS

Tên node	Biểu tượng
Depot	
Transfer station 1	
Transfer station 2	
Gather sites	

Từ dữ liệu địa lý của các node, khoảng cách và thời gian đi giữa hai node bất kỳ được tính dựa vào chức năng Network Analyst trong ArcGIS



Hình 3.4: Dữ liệu các khoảng cách thời gian giữa các node được tính thông qua chức năng Network Analyst trong phần mềm ArcGIS



Hình 3.5: Export dữ liệu các khoảng cách thời gian giữa các node ra file Excel

3.6. Kết quả thực nghiệm

Bảng 3.11: Kết quả thực nghiệm

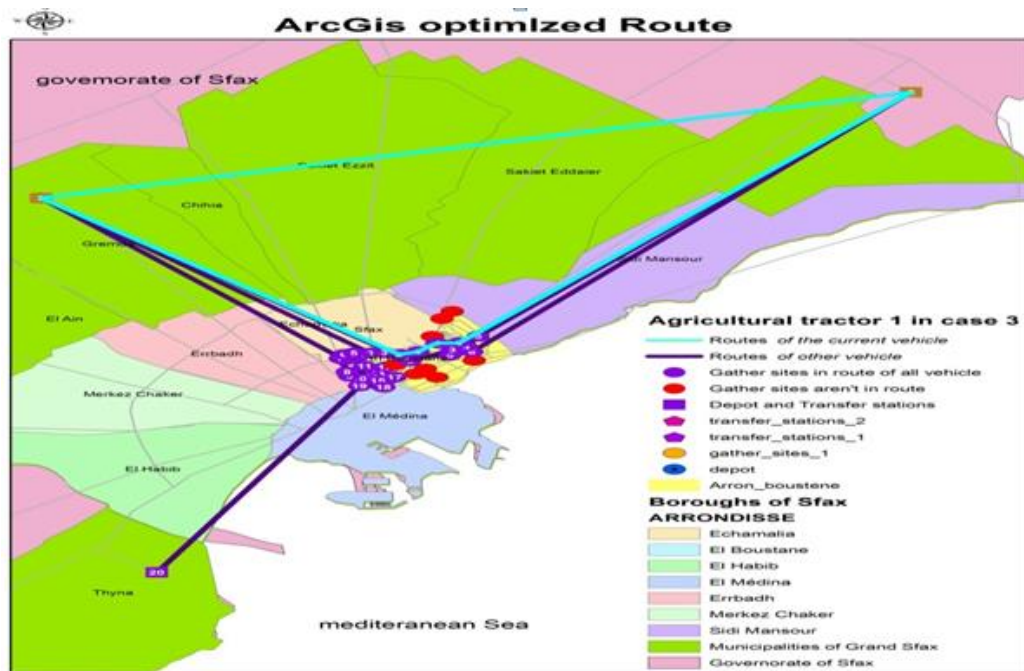
Tiêu chí	Tuyến đường thực tế (Sfax - 2016)	Thuật toán Dijkstra cải tiến	Thuật toán di truyền
Tổng thời gian (h)	14.100000000	10.9174242429	10.9154286219
Tổng khoảng cách (km)	166.7500000	154.100000062	153.836578094

Hành trình các xe trong Phương pháp **Thuật toán Dijkstra** như sau:

Trip:

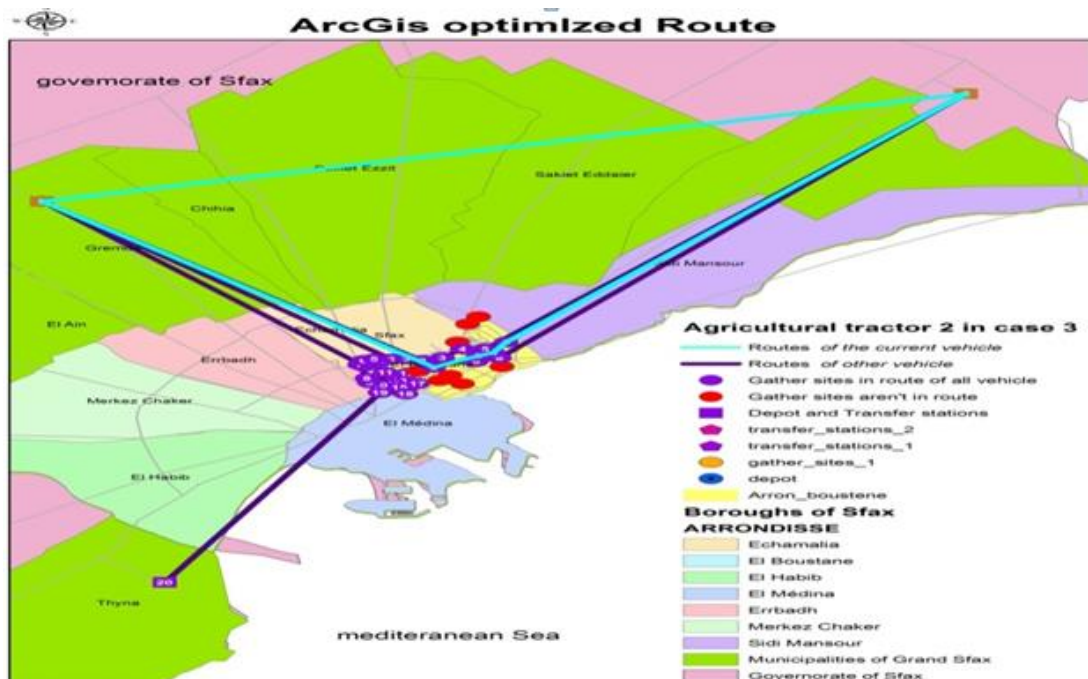
{11: [1.0, 18.0, 17.0, 20.0, 21.0, 16.0, 4.0, 25.0, 41.0, 40.0, 29.0, 42.0, 38.0, 39.0, 35.0, 34.0, 33.0, 31.0, 30.0, 3.0], 12: [1.0, 19.0, 15.0, 14.0, 11.0, 10.0, 3.0, 1.0], 21: [3.0, 24.0, 28.0, 27.0, 26.0, 32.0, 37.0, 36.0, 22.0, 3.0, 1.0], 14: [1.0, 8.0, 13.0, 6.0, 23.0, 3.0, 1.0], 13: [1.0, 5.0, 12.0, 7.0, 9.0, 3.0, 1.0]}

- Agricultural tractor 1



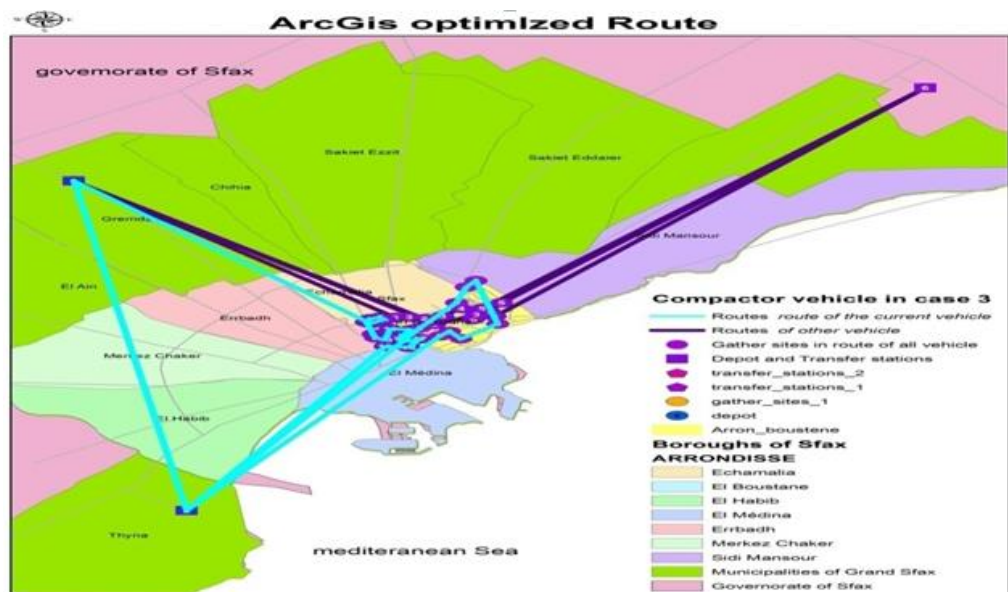
Hình 3.6: Kết quả tuyến đường của xe 1 trong phương pháp Dijkstra

- Agricultural tractor 2



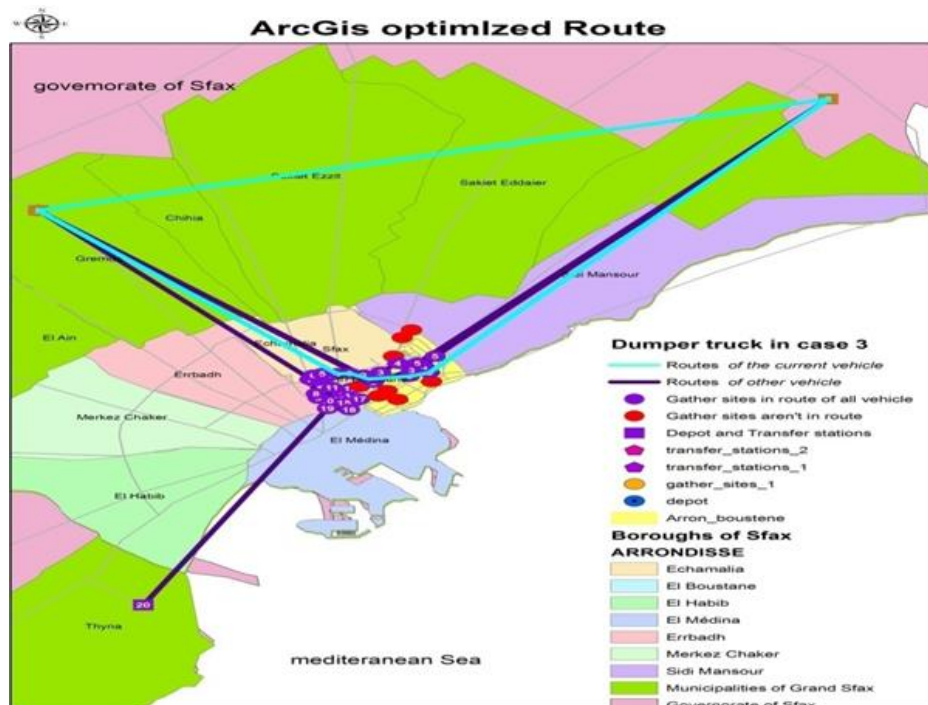
Hình 3.7: Kết quả tuyến đường của xe 2 trong phương pháp Dijkstra

- Compactor vehicle



Hình 3.8: Kết quả tuyến đường của xe thứ 3 trong phương pháp Dijkstra

- Dumper truck

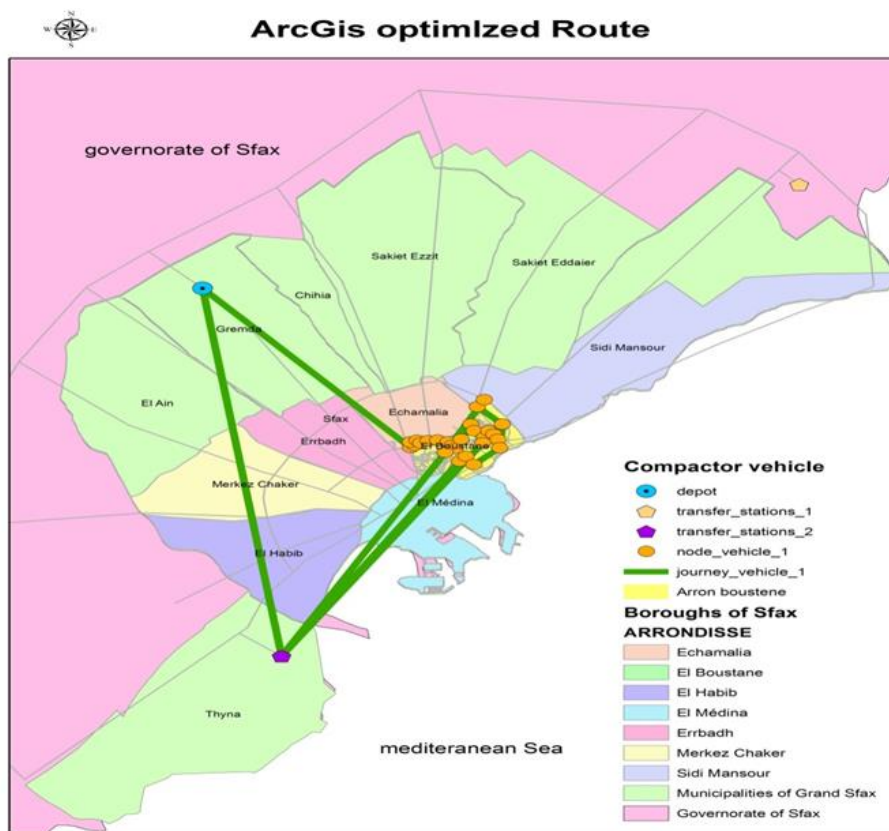


Hình 3.9: Kết quả tuyến đường của xe 4 trong phương pháp Dijkstra

Hành trình các xe trong phương pháp dùng **Thuật toán di truyền** như sau :

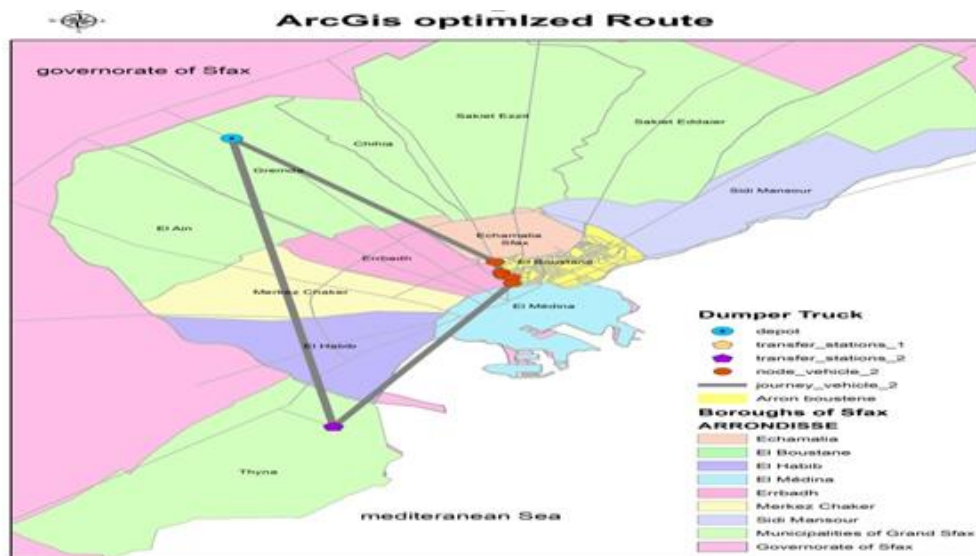
Trip {11: [1.0, 18.0, 17.0, 20.0, 21.0, 16.0, 4.0, 25.0, 41.0, 40.0, 29.0, 42.0, 38.0, 39.0, 35.0, 34.0, 33.0, 31.0, 30.0, 3.0], 12: [1.0, 19.0, 15.0, 14.0, 11.0, 10.0, 3.0, 1.0], 21: [3.0, 24.0, 28.0, 27.0, 26.0, 32.0, 37.0, 36.0, 22.0, 3.0, 1.0], 14: [1.0, 13.0, 9.0, 6.0, 23.0, 3.0, 1.0], 13: [1.0, 5.0, 12.0, 7.0, 8.0, 3.0, 1.0]}

- **Compactor vehicle**



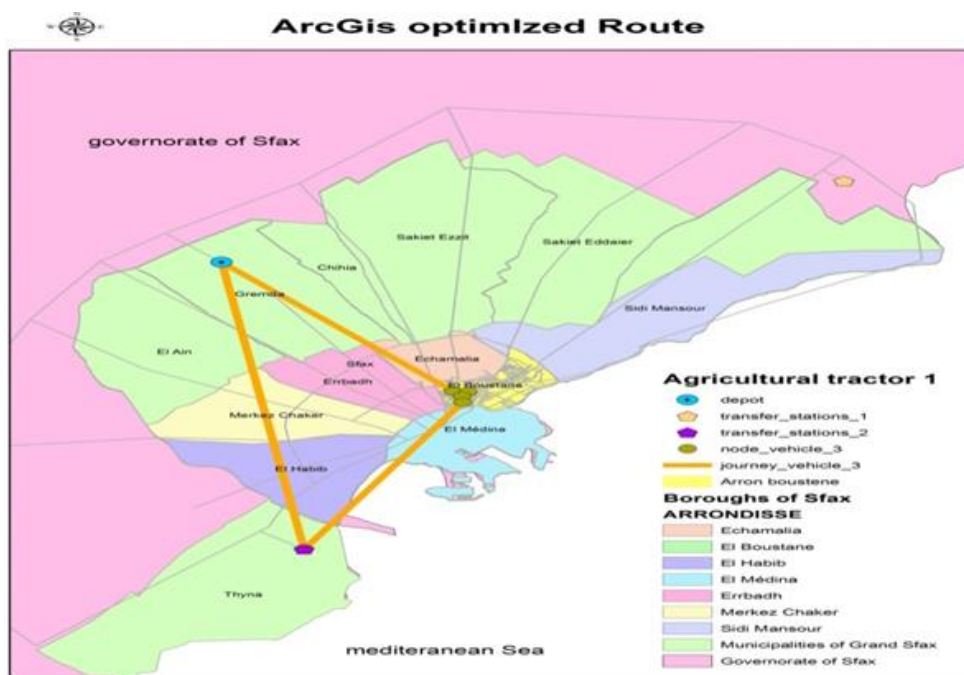
Hình 3.10: Kết quả tuyến đường của xe thứ nhất trong phương pháp GA

- Dumper truck



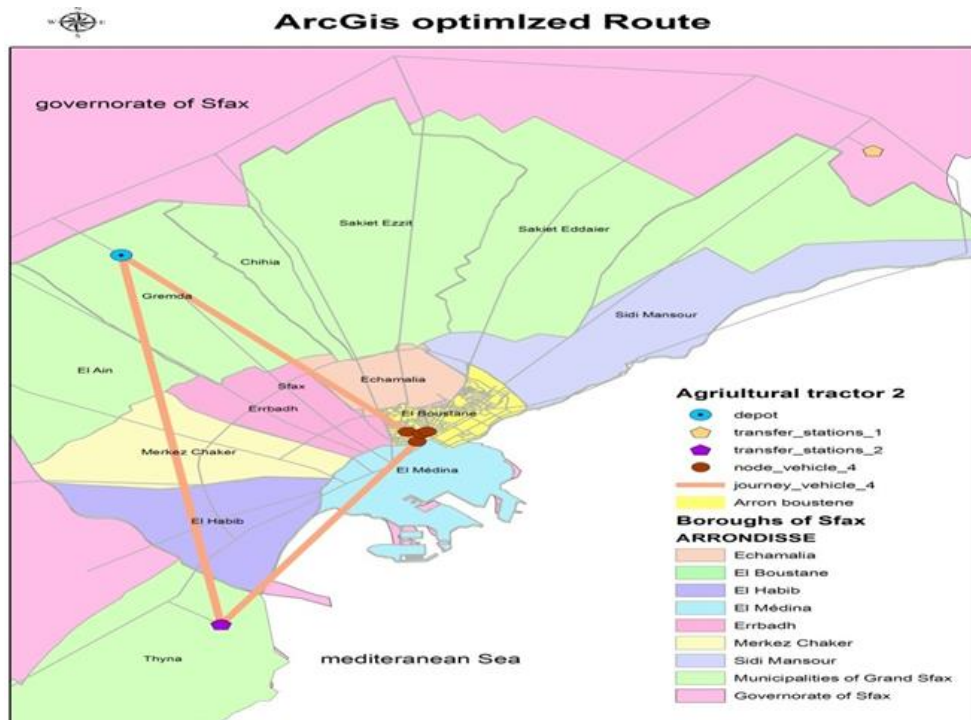
Hình 3.11: Kết quả tuyến đường của xe thứ hai trong phương pháp GA

- Agricultural tractor 1



Hình 3.12: Kết quả tuyến đường của xe thứ ba trong phương pháp GA

- Agricultural tractor 2



Hình 3.13: Kết quả tuyến đường của xe thứ tư trong phương pháp GA

3.7. Đánh giá và so sánh

Từ kết quả thực nghiệm trong bảng 3.2 đã cho thấy tổng thời gian vận chuyển chất thải và khoảng cách đi lại trong phương pháp dùng thuật toán Dijkstra cải tiến tốt hơn tổng thời gian vận chuyển chất thải và khoảng cách đi lại tuyến đường thực tế đang được áp dụng tại thành phố Sfax năm 2016, và tổng thời gian và khoảng cách trong phương pháp dùng thuật toán di truyền tốt hơn thuật toán Dijkstra. Điều này cho rõ ràng thấy dùng thuật toán di truyền vào bài toán này sẽ cải thiện tốt hơn so với hành trình thu gom chất thải được áp dụng tại thành phố Sfax.

Thuật toán dijktra cải tiến dựa vào việc chọn thành phố tiếp theo để đi đến có thời gian di chuyển ngắn nhất từ thành phố hiện tại trên hành trình của một xe. Bắt đầu, từ điểm depot số 1, xe đầu tiên chạy thẳng ngay đến thành phố có thời gian di chuyển ngắn nhất là thành phố số 18. Tiếp theo, thành phố số 2 được chọn để làm điểm di chuyển tiếp vì có thời gian di chuyển ngắn nhất từ thành phố số 18. Cứ tiếp

diễn theo quy luật trên, ta hình thành một lịch trình đi cho các xe để thu gom rác thải.

Việc sử dụng thuật toán Dijkstra làm điểm khởi tạo và sử dụng toán tử lai ghép có yếu tố di truyền theo cạnh đã làm cho hành trình của thuật toán di truyền có rất nhiều điểm chung với dijktra.

Với việc dùng khởi hành trình được tạo ra bởi thuật toán Dijkstra cải tiến làm gen khởi tạo, thuật toán di truyền đã giảm thời gian chạy thuật toán đi rất nhiều khi thuật toán chạy đạt đến điểm hội tụ. Điều này lý giải rõ ràng kết quả tại sao mà thuật toán di truyền tốt hơn thuật toán Dijkstra cải tiến cả về thời gian và khoảng cách. Phương pháp dùng thuật toán Dijkstra cải tiến và phương pháp di truyền được thiết kế đều đã giải quyết được các ràng buộc trong kịch bản thu gom chất thải.

Tuy nhiên, so sánh quãng thời gian của thuật toán di truyền (10.915 giờ) và thuật toán Dijkstra cải tiến (10.917 giờ) và quãng đường của thuật toán di truyền (154.1 km) và thuật toán Dijkstra cải tiến (153.8 km) đã thực nghiệm, tuy tốt hơn nhưng khoảng chênh lệch là khá nhỏ. Điều này gợi ý cho rằng có thể thuật toán có thể thay đổi các kỹ thuật khác của các thành phần trong thuật toán di truyền như chọn một cách lai ghép khác, cách đột biến khác có thể phù hợp hơn với bài toán và thu được kết quả tốt hơn đáng kể.

Từ các kết quả thu được ở trên, để giải quyết bài toán tối ưu thời gian thu gom chất thải, ta nên sử dụng phương pháp thuật toán di truyền để giải quyết triệt để các yêu cầu của bài toán, từ đó cho kết quả tốt hơn.

3.8. Tổng kết chương

Nội dung chương 3 là kết quả thực nghiệm của hai phương pháp dùng thuật toán di truyền và Dijkstra cải tiến tại thành phố Sfax, tunisia. Kết quả của hai phương pháp là khác nhau. Kết quả thực nghiệm cho thấy rõ hơn việc áp dụng thuật toán di truyền vào vào toán thu gom chất thải tại thành phố Sfax cái thiện thời gian

và khoảng cách đi đáng kể. Tuy nhiên, thay đổi cách tiếp cận khác của các toán tử dùng trong thuật toán di truyền có thể có những kết quả tốt hơn cho bài toán.

KẾT LUẬN

Bài toán thu gom chất thải rắn đô thị đang là vấn đề cấp thiết trong thực tế, bài toán mang nhiều ý nghĩa về mặt môi trường, phát triển cảnh quan và tiết kiệm kinh tế. Bài toán tối ưu thu gom chất thải rắn đô thị thuộc lớp bài toán tối ưu tổ hợp nên nhóm các phương pháp tối ưu tiến hóa thường được sử dụng. Xuất phát từ ý nghĩa thực tế đó, luận văn đã nghiên cứu và trình bày một số nội dung chính như sau:

- Tìm hiểu được tổng quan về bài toán tối ưu thu gom chất thải rắn đô thị, và kiến thức tổng quan thuật toán di truyền.
- Xây dựng thuật toán di truyền cho bài toán tối ưu thu gom chất thải rắn
- Triển khai thực nghiệm bài toán tại thành phố Sfax, Tunisia.

Dựa trên những kết quả bước đầu đã đạt được, trong tương lai, đề tài có thể được phát triển theo các hướng như sau:

- Tiếp tục cải tiến các phương pháp.
- Xây dựng một phương pháp mới dựa trên một thuật toán khác để đạt được hiệu quả thu gom chất thải tối ưu hơn nữa.

TÀI LIỆU THAM KHẢO

Tiếng Việt:

1. Huỳnh Quang Đệ (2015), *Đánh giá hiệu quả của giải thuật di truyền giải bài toán cây khung truyền thống tối ưu với các kỹ thuật mã hóa cây*, Luận văn thạc sĩ khoa học ngành Công nghệ thông tin, TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI.

2. Lại Thị Nhung (2011), *Thiết kế chuỗi cung ứng bằng giải thuật di truyền*, Luận văn thạc sĩ khoa học ngành Bảo đảm toán học cho máy tính & các hệ thống tính toán, Đại học Khoa Học Tự Nhiên – Đại học Quốc Gia Hà Nội.

Tiếng Anh:

3. Apaydin, O., Gonullu, M. T. (2008). Emission control with route optimization in solid waste collection process: A case study. *Sadhana*, 33(2), 71–82.

4. Ai, T. J., Kachitvichyanukul, V. (2009). A particle swarm optimisation for vehicle routing problem with time windows. *International Journal of Operational Research*, 6(4), 519-537

5. Alvarenga, G. B., de Abreu Silva, R. M., & Sampaio, R. M. (2004). A hybrid algorithm for the vehicle routing problem with window. *INFOCOMP Journal of Computer Science*, 4(2), 9-16.

6. Beliën, J., Boeck, L. De, Ackere, J. Van, 2012. Municipal Solid Waste Collection and Management Problems : *A Literature Review 1655*, 1–25.

7. Bonomo, F., Durán, G., Larumbe, F., Marengo, J., 2012. A method for optimizing waste collection using mathematical programming: a Buenos Aires case study. *Waste Manag. Res.* 30, 311–24. doi:10.1177/0734242X11402870

8. Chalkias, C., Lasaridi, K. (2009). A GIS based model for the optimisation of municipal solid waste collection: the case study of Nikea, Athens, Greece. *WSEAS Transactions on Environment and Development*, 5(10), 640-650.

9. Daniel, L., Loree, P., Whitener, A. (2002). *Inside MapInfo professional: the friendly user guide to MapInfo professional*. Cengage Learning.
10. Das, S., Bhattacharyya, B.K., 2015. Optimization of municipal solid waste collection and transportation routes. *WASTE Manag.* doi:10.1016/j.wasman.2015.06.033.
11. Faccio, M., Persona, A., Zanin, G., 2011. Waste collection multi objective model with real time traceability data. *Waste Manag.* 31, 2391–2405. doi:10.1016/j.wasman.2011.07.005.
12. Han, H., 2015. Waste Collection Vehicle Routing Problem : A Literature Review 27, 1–12. doi:10.7307/ptt.v27i4.1616.
13. Huang, S., Lin, P., 2015. Vehicle routing – scheduling for municipal waste collection system under the “ Keep Trash off the Ground ” policy \$. *Omega* 55, 24–37. doi:10.1016/j.omega.2015.02.004.
14. HONG, Tzung-Pei, et al. Evolution of appropriate crossover and mutation operators in a genetic process. *Applied Intelligence*, 2002, 16.1: 7-17.
15. Ing, H.K., Analytique, C., Service, I., Atmosph, P., 2007. *Surveillance de la Qualité de l ' Air en Tunisie* Ingénieur Principal en Chimie Analytique Réseau National de Surveillance de la Qualité de l ' Air.
16. JIH, Wan-rong; HSU, Y. A family competition genetic algorithm for the pickup and delivery problems with time window. *Bulletin of the College of Engineering*, 2004, 90: 121-130.
17. Jozefowicz, N., 2008. *Multi-objective vehicle routing problems* 189, 293–309. doi:10.1016/j.ejor.2007.05.055
18. Karadimas, N. V., Kolokathi, M., Defteraious, G., Loumos, V., 2007. Ant Colony System vs ArcGIS Network Analyst: *The Case of Municipal Solid Waste Collection*. 5th WSEAS Int. Conf. Environ. Ecosyst. Dev. 128–134.

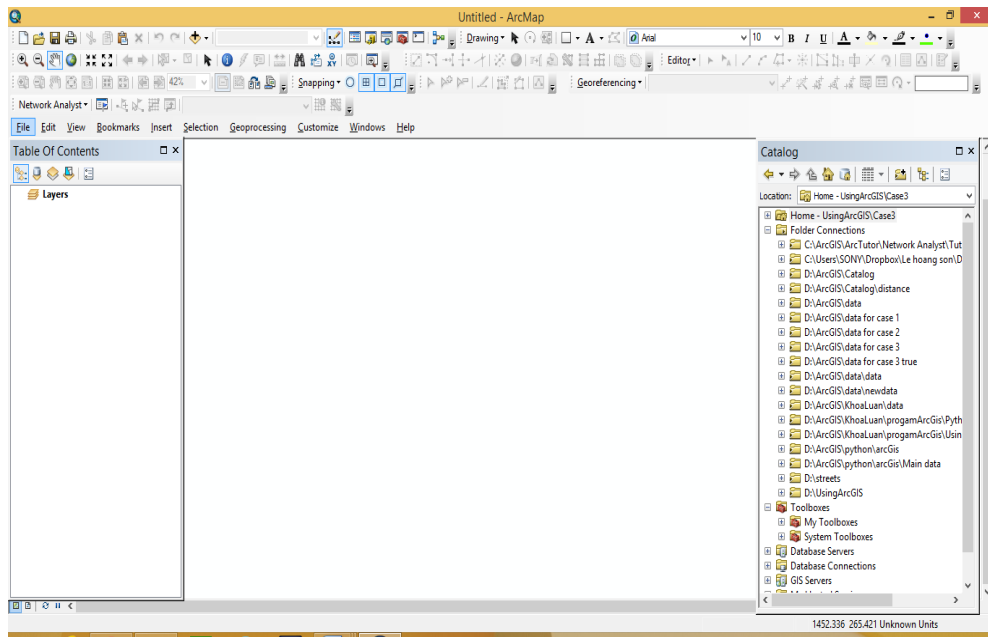
19. Rosen, L. (2005). *Open source licensing*. Prentice Hall.
20. Sahoo, S., Kim, S., Kim, B.-I., Kraas, B., Popov Jr., A., 2005. *Routing optimization for Waste Management*. Interfaces (Providence). 35, 24–36. doi:10.1287/inte.1040.0109.
21. Santos, L., Coutinho-Rodrigues, J., Antunes, C.H., 2011. *A web spatial decision support system for vehicle routing using Google Maps*. Decis. Support Syst. 51, 1–9. doi:10.1016/j.dss.2010.11.008.
22. Son, L.H., 2014. Optimizing Municipal Solid Waste collection using Chaotic Particle Swarm Optimization in GIS based environments: *A case study at Danang city, Vietnam*. Expert Syst. Appl. 41, 8062–8074. doi:10.1016/j.eswa.2014.07.020.
23. SRINIVAS, Mandavilli; PATNAIK, Lalit M. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 1994, 24.4: 656-667.
24. TAN, Kay Chen, et al. Heuristic methods for vehicle routing problem with time windows. *Artificial intelligence in Engineering*, 2001, 15.3: 281-295.
25. YEUN, Liong Choong, et al. Vehicle routing problem: models and solutions. *Journal of Quality Measurement and Analysis*, 2008, 4.1: 205-218.
26. VAIRA, Gintaras; KURASOVA, Olga. Genetic algorithms and VRP: the behaviour of a crossover operator. *Baltic Journal of Modern Computing*, 2013, 1.3-4: 161-185.
27. <http://www.rubicite.com/Tutorials/GeneticAlgorithms/CrossoverOperators/EdgeRecombinationCrossoverOperator.aspx>

PHỤ LỤC


CODE PYTHON CHO THUẬT TOÁN

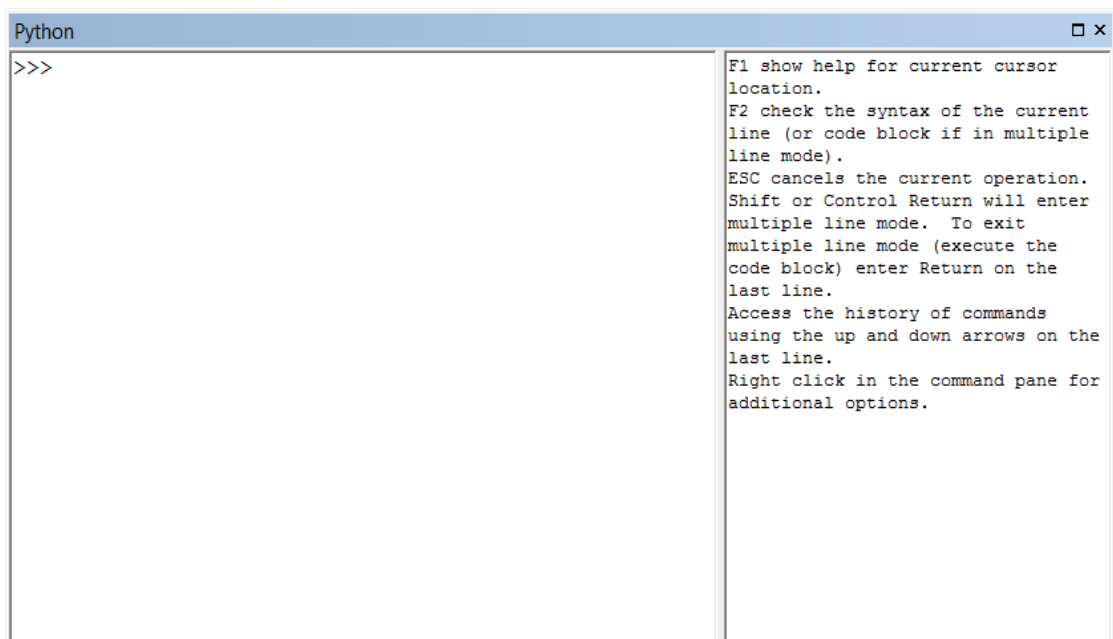
1. Mở ArcGIS

Click ArcMap để mở chương trình trong ArcGIS. Giao diện như sau:



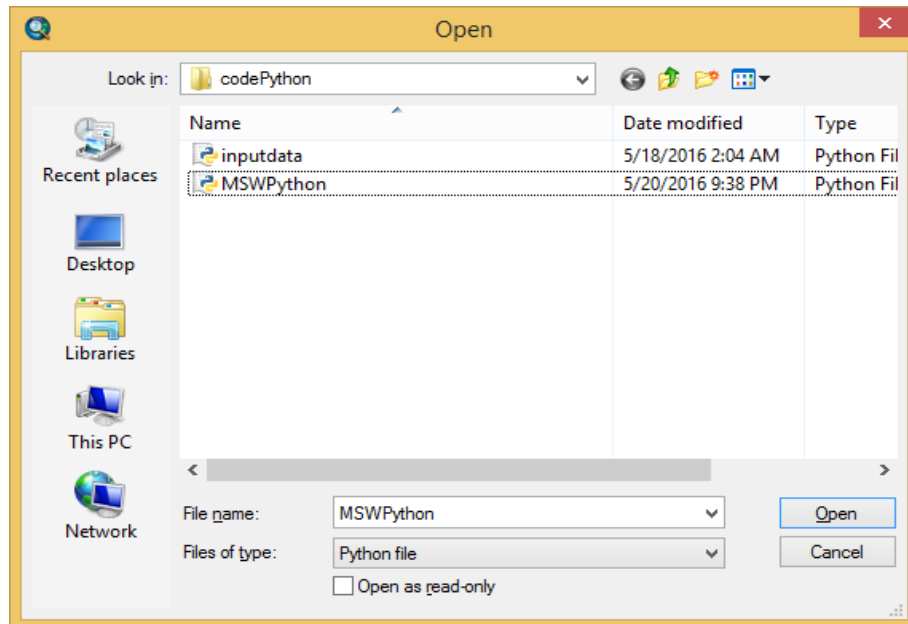
2. Mở Python trong ArcGIS

Click nút  trên thanh công cụ để mở Python, giao diện như sau:



3. Load file trong Python

Click chuột phải trong giao diện Python và chọn file cần load lên.



4. Đưa dữ liệu vào arcGIS

Tạo file mxd trong đường dẫn D:\ArcGIS\LuanVanGA\progamArcGis\Python

Tên của file này là TunisiaPython.mxd

Dữ liệu để trong đường dẫn: D:\ArcGIS\LuanVanGA\data

```
import arcpy
try:
    mxd =
arcpy.mapping.MapDocument(r"D:\ArcGIS\LuanVanGA\progamArcGis\Python\TunisiaPython.mxd")
    df = arcpy.mapping.ListDataFrames(mxd, "Layers")[0]

    newlayer1 =
arcpy.mapping.Layer(r"D:\ArcGIS\LuanVanGA\data\bostendata_sans.mdb\transport\Boustene")
    newlayer2 =
arcpy.mapping.Layer(r"D:\ArcGIS\LuanVanGA\data\ADMIN_LIMITS_UTM_Car\Commu_SF_UTM_Car.shp")
```

```

newlayer3 =
arcpy.mapping.Layer(r"D:\ArcGIS\LuanVanGA\data\ADMIN_LIMITS_UTM_Car\Arrond_SF_UTM_Car.shp")
newlayer4 =
arcpy.mapping.Layer(r"D:\ArcGIS\LuanVanGA\data\Arron_boustene.shp"
)
newlayer5 =
arcpy.mapping.Layer(r"D:\ArcGIS\LuanVanGA\data\bostendata_sans.mdb
\transport\transfer_stations_2")
newlayer6 =
arcpy.mapping.Layer(r"D:\ArcGIS\LuanVanGA\data\bostendata_sans.mdb
\transport\transfer_stations_1")
newlayer7 =
arcpy.mapping.Layer(r"D:\ArcGIS\LuanVanGA\data\bostendata_sans.mdb
\transport\Depot")

arcpy.mapping.AddLayer(df, newlayer1, "TOP")
arcpy.mapping.AddLayer(df, newlayer2, "TOP")
arcpy.mapping.AddLayer(df, newlayer3, "TOP")
arcpy.mapping.AddLayer(df, newlayer4, "TOP")
arcpy.mapping.AddLayer(df, newlayer5, "TOP")
arcpy.mapping.AddLayer(df, newlayer6, "TOP")
arcpy.mapping.AddLayer(df, newlayer7, "TOP")

arcpy.RefreshActiveView()
arcpy.RefreshTOC()
mxd.save()
except Exception as ex:
print ex.args[0]

arcpy.management.MakeFeatureLayer(r"D:\ArcGIS\LuanVanGA\data\bost
endata_sans.mdb\transport\Boustene","Governorate of Sfax")

arcpy.management.MakeFeatureLayer(r"D:\ArcGIS\LuanVanGA\data\AD
MIN_LIMITS_UTM_Car\Commu_SF_UTM_Car.shp","Municipalities of
Grand Sfax")

arcpy.management.MakeFeatureLayer(r"D:\ArcGIS\LuanVanGA\data\AD
MIN_LIMITS_UTM_Car\Arrond_SF_UTM_Car.shp","Boroughs of
Sfax")

arcpy.management.MakeFeatureLayer(r"D:\ArcGIS\LuanVanGA\data\Arr
on_boustene.shp","Arron boustene")

```

```

arcpy.management.MakeFeatureLayer(r"D:\ArcGIS\LuanVanGA\data\bost
endata_sans.mdb\transport\transfer_stations_2", "transfer_stations_2")

arcpy.management.MakeFeatureLayer(r"D:\ArcGIS\LuanVanGA\data\bost
endata_sans.mdb\transport\transfer_stations_1", "transfer_stations_1")

arcpy.management.MakeFeatureLayer(r"D:\ArcGIS\LuanVanGA\data\bost
endata_sans.mdb\transport\Depot", "Depot")
mxd.save()

```

5. Đọc dữ liệu

File MainData.xlsx đặt trong đường dẫn: D:/ArcGIS/LuanVanGA/data

```

#read file
file_capacity = "D:/ArcGIS/LuanVanGA/data/MainData.xlsx"
wb = xlrd.open_workbook(file_capacity)
sheet = wb.sheet_by_index(0)
# capacity of the vehicle
C = [[sheet.cell_value(r, c) for c in range(sheet.ncols)] for
r in range(sheet.nrows)]
# set a dictionary include: the nodes : [capacity of the nodes, time service,
coordinates X, coordinates Y]
allNode = { } # dictionary of all node
sheet1 = wb.sheet_by_index(1)
data = [[sheet1.cell_value(r, c) for c in range(sheet1.ncols)] for r
in range(sheet1.nrows)]
for rows in range(sheet1.nrows):
    allNode[data[rows][0]] = [data[rows][2], data[rows][3], data[rows][4],
data[rows][5]]
del allNode[data[0][0]]
# find dictionary of all Gather sites
ZNode = allNode.copy()
allKey = allNode.keys() # list key of all node
for i in range(3):
    del ZNode[allKey[i]]
ZKey = ZNode.keys() # list key of Gather sites
# read the distance and time between the node
sheet2 = wb.sheet_by_index(2)
data2 = [[sheet2.cell_value(r, c) for c in range(sheet2.ncols)] for r
in range(sheet2.nrows)]

```

$V = \{1, 2, 3, 4\}$

Biến data dùng để đọc sức chứa chất thải, thời gian làm dịch vụ, tọa độ X, tọa độ Y của tất cả các node.

Biến data2 để đọc khoảng cách và thời gian giữa hai node bất kỳ

Biến C để đọc sức chứa của các xe.

6. Tìm hàng xóm cho node hiện tại

```
# Find Gather sites neighbors of node a in the order of closest to farthest
def Neighbor(a1, gather):
    N = [] # list Gather sites neighbors of node a in the order of closest to farthest
    distance = []
    for i in ZKey:
        if(a1 != i):
            # find the distance between node i and node a
            for rows in range(sheet2.nrows):
                if ((a1 == data2[rows][0]) & (i == data2[rows][1])):
                    distance.append(data2[rows][2])
            break
        distance = sorted(distance)
        for j in distance:
            for rows in range(sheet2.nrows):
                if((j == data2[rows][2]) & (a1 == data2[rows][0])):
                    N.append(data2[rows][1])
            break
    return N
#*****
```

7. Thuật toán Dijkstra dựa vào kịch bản thu gom chất thải

```
# Main algorithm

def Route():

    sumZ = 0 # total current waste of all the Gather sites

    for j in ZKey:
```

```

sumZ += ZNode[j][0]

Node = [[allKey[0]] : allKey[0]] : allKey[0]] : allKey[0]] #Initialize
the vehicle start at Depot

Distance = [0, 0, 0, 0]

Time = [0, 0, 0, 0]

Q = [0, 0, 0, 0]

label1 = { } #old label

label2 = { } # new label

la = 0

lb = 0

# find the first lable list for all nodes

neighborNode = Neighbor(a1=allKey[0],gather=ZKey)

for c in neighborNode:

for rows in range(sheet2.nrows):

if((allKey[0] == data2[rows][0])&(c == data2[rows][1])):

    label1[c] = data2[rows][2]

# constraints in model

while((sumZ > 0)&(len(ZKey) != 0)):

for i in V:

    a = Node[i-1][len(Node[i-1])-1]

while((((C[i][2]-Q[i-1])>0.4) | (math.fabs(C[i][2]-Q[i-1] - 0.4) <=
0.01))&(sumZ > 0)& (len(ZKey) != 0):

    neighbor = Neighbor(a1=a,gather=ZKey)

```



```

        # calculate new lable for all nodes

for d in neighbor:

    for rows in range(sheet2.nrows):

        if((a == data2[rows][0]) & (d == data2[rows][1])):

            label2[d] = la + data2[rows][2]

# start for constraints

for b in neighbor:

    u = C[i][2] - Q[i-1]

    if(u > ZNode[b][0]):

        Q[i-1] += ZNode[b][0]

    if(math.fabs(u - ZNode[b][0]) <= 0.01):

        Q[i-1] += ZNode[b][0]

    if((label2[b] <= label1[b]) & (Q[i-1] <= C[i][2]) & ((u > ZNode[b][0]) |
    (math.fabs(u - ZNode[b][0]) <= 0.01))):

        for rows in range(sheet2.nrows):

            if ((a == data2[rows][0]) & (b == data2[rows][1])):

                Distance[i-1] += data2[rows][5]

                Time[i-1] += (data2[rows][7] + ZNode[b][1])

        break

    sumZ -= ZNode[b][0]

    ZNode[b][0] = 0

    Node[i-1].append(b)

    ZKey.remove(b)

```

```

        la = label2[b]

        del label2[b]

        label1 = label2

        a = b

        break

if((C[i][2]-Q[i-1]) < 0.4): # the vehicle will go TS to unload waste

    disTS1 = 0

    disTS2 = 0

    timeTS1 = 0

    timeTS2 = 0

    for rows in range(sheet2.nrows):

if((a == data2[rows][0]) & (allKey[1] == data2[rows][1]]):

        disTS1 = data2[rows][5]

        timeTS1 = data2[rows][7]

if((a == data2[rows][0]) & (allKey[2] == data2[rows][1]]):

        disTS2 = data2[rows][5]

        timeTS2 = data2[rows][7]

        # Check the vehicle will go TS1 or TS2

if(disTS1 < disTS2):

        Node[i-1].append(allKey[1])

        Distance[i-1] += disTS1

        Time[i-1] += timeTS1

```

```

        Q[i-1] = 0

    else:

        Node[i-1].append(allKey[2])

        Distance[i-1] += disTS2

        Time[i-1] += timeTS2

        Q[i-1] = 0

    # if total current = 0, check if current node isn't TS, the vehicle will go
    # TS then come back Depot. If current node is TS, vehicle will come back
    # Depot.

    for i in V:

        distance1 = 0

        distance2 = 0

        time1 = 0

        time2 = 0

        distance01 = 0

        distance02 = 0

        time01 = 0

        time02 = 0

        currentNode = Node[i-1][len(Node[i-1])-1]

    for rows in range(sheet2.nrows):

        if ((currentNode == data2[rows][0]) & (allKey[1] == data2[rows][1])):

            distance1 = data2[rows][5]

            time1 = data2[rows][7]

```

```

        if ((currentNode == data2[rows][0]) & (allKey[2] ==
data2[rows][1])):

            distance2 = data2[rows][5]

            time2 = data2[rows][7]

        if ((allKey[0] == data2[rows][0]) & (allKey[1] == data2[rows][1])):

            distance01 = data2[rows][5]

            time01 = data2[rows][7]

        if ((allKey[0] == data2[rows][0]) & (allKey[2] == data2[rows][1])):

            distance02 = data2[rows][5]

            time02 = data2[rows][7]

        if((currentNode != allKey[1]) & (currentNode != allKey[2])):

            if(distance1 < distance2):

                Node[i-1].append(allKey[1])

                Node[i-1].append(allKey[0])

                Distance[i-1] += (distance1 + distance01)

                Time[i-1] += (time1 + time01)

            else:

                Node[i-1].append(allKey[2])

                Node[i-1].append(allKey[0])

                Distance[i-1] += (distance2 + distance02)

                Time[i-1] += (time2 + time02)

        elif(currentNode == allKey[1]):

            Node[i-1].append(allKey[0])

```

```

        Distance[i-1] += distance01

        Time[i-1] += time01

    else:

        Node[i-1].append(allKey[0])

        Distance[i-1] += distance02

        Time[i-1] += time02

    print"Route of ",C[i][1],": "

    for j in Node[i-1]:

        print j,"-->"

    print"Total Distance: ",Distance[i-1]

    print "Total Time: ", Time[i-1]

#print result

    totalDistance = 0

    totalTime = 0

    for i in Distance:

        totalDistance += i

    for i in Time:

        totalTime += i

    print"Total Distance of all the vehicle:",totalDistance
    print"Total Time of all the vehicle:",totalTime

```

8. Đưa kết quả của mỗi xe lên ArcGIS

Node[0]: Danh sách các node trong hành trình của Compactor vehicles vehicle

Node[1]: Danh sách các node trong hành trình của xe Dumper truck

Node[2]: Danh sách các node trong hành trình của xe Agricultural tractor 1

Node[4]: Danh sách các node trong hành trình của xe Agricultural tractor 2

Sử dụng hàm PointsToLine để nối hai điểm với nhau

Thay đổi chỉ số của Node[], tên của out_feat_class và tên của hành trình của xe để hiển thị kết quả lên ArcGIS.

```
#add result to arcGis
out_feat_class = 'node_vehicle_1.shp'
workspace = r'D:\ArcGIS\LuanVanGA\progamArcGis\Python'

os.chdir(workspace)
gp = arcgisscripting.create(10.2)
gp.toolbox = 'management'
gp.workspace = workspace

#Create feature class
gp.CreateFeatureclass(workspace, out_feat_class, 'POINT')

# Get insert cursor
rows = gp.InsertCursor(out_feat_class)
feat = rows.NewRow()

# Set geometry
for t in Node[0]:
    ptObj = gp.CreateObject('Point')
    ptObj.x = allNode[t][2]
    ptObj.y = allNode[t][3]
    feat.Shape = ptObj

# Add to feature class
rows.InsertRow(feat)

arcpy.PointsToLine_management("node_vehicle_1.shp",r"D:\ArcGIS\LuanVanGA\data\bostendata_sans.mdb\transport\journey_vehicle_1")
```