การประยุกต์ใช้ไดจ์คสตร้าอัลกอริทึมกับเส้นทางจราจรเพื่อหาเส้นทางที่สั้นที่สุด

Traffic Route for the Shortest Path Travelling Using Dijkstra Algorithm

สุวรรณี อัศวกุลชัย
รองศาสตราจารย์ ดร. สาขาวิศวกรรมคอมพิวเตอร์และมัลติมีเดีย
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยหอการค้าไทย
E-mail: suwannee_ads@utcc.ac.th

บทคัดย่อ

การหาเส้นทางที่สั้นที่สุดกับเส้นทางจราจร เป็นวิธีการที่ช่วยในการลดค่าใช้จ่ายการเดินทางได้เป็นอย่างมาก ดังนั้นในการศึกษาครั้งนี้ มีวัตถุประสงค์ เพื่อพัฒนาวิธีการเลือกเส้นทางจราจรที่สั้นที่สุด สำหรับการเดินทางจากมหาวิทยาลัยหอการค้าไทยถึงปากซอยประชาสงเคราะห์ 23 ด้วยการใช้อัลกอริทึมของ Dijkstra เป็นพื้นฐาน จากการวิเคราะห์เพื่อให้ได้เส้นทางจราจรที่สั้นที่สุด และเพิ่มปัจจัยอื่นๆ ได้แก่ ทุกเส้นทางมีน้ำหนักเท่ากัน สามารถไปได้เพียง 2 เส้นทาง พบว่า มีเส้นทางจราจรที่เป็นไปได้ทั้งหมด 20 เส้นทาง และ ไดจ์คสตร้าอัลกอริทึมสามารถหาเส้นทางที่สั้นที่สุดเท่ากับ 253.25 เมตร ทั้งนี้จำนวนออปเจคที่เหมาะสมอย่างน้อย 1 ออปเจค ได้เส้นทางที่สั้นที่สุด คือ 235 ออปเจค สรุปผลได้ว่า การประยุกต์ไดจ์คสตร้าอัลกอริทึม สามารถหาเส้นทางที่สั้นที่สุดได้ถูกต้องร้อยละ 100

คำสำคัญ: ไดจ์คสตร้าอัลกอริทึม เส้นทางที่สั้นที่สุด มหาวิทยาลัยหอการค้าไทย

ABSTRACT

The shortest path of the traffic route is a way to reduce the cost of travel. Thus, this study aims to develop a methodology to apply the Dijkstra Algorithm for the shortest path from University of the Thai Chamber of Commerce to Soi Pracha-Uthid 23. Based on Dijkstra's algorithm, an additional factor related to the shortest path and others parameters i.e. all routes are equally weighted and the only two paths are open. The results from this study demonstrated that there are totally possible 20 routes. Under the Dijkstra Algorithm is applied for the shortest path that is 253.25 meters. In addition, the number of object is appropriate at least one object and the shortest route is 235 objects. It can be concluded that the accuracy of the Dijkstra Algorithm applied for the shortest path was 100 percent.

KEYWORDS: Dijkstra Algorithm, shortest path, University of the Thai Chamber of Commerce

บทนำ

ขั้นตอนวิธีในการทำงานเพื่อมาแก้ปัญหาใดปัญหาหนึ่ง ที่เรียกว่า อัลกอริธึมหรือลำดับขั้นตอน (Algorithm) (Denardo, 2003) ที่เหมาะสมมาประยุกต์ใช้ให้เกิดประโยชน์สูงสุด เช่น ลำดับขั้นตอนวิธีในการทำงานเพื่อแก้ปัญหาด้านการเดินรถที่ใช้สำหรับในการค้นหาเส้นทางที่สั้นที่สุด (Shortest Path) (Denardo, 2003) ที่เป็นที่นิยมใช้ วิธีการที่นิยมใช้ในการค้นหาเส้นทางที่สั้นที่สุด คือ Dijkstra's algorithm (Denardo, 2003; Kroger, 2005) ซึ่งปัญหาในการหาเส้นทางที่สั้นที่สุด ซึ่งต้องเดินทาง

ไปตามจำนวนโหนด (Node or City) ที่ได้กำหนดให้ครบทุกเมือง โดยที่ต้องไม่เดินทางซ้ำไปยังเมืองที่เคยเดินทางมาแล้ว (Pouly, and Kohlas, 2011) การเลือกเส้นทางจึงจำเป็นในการจัดเรียงลำดับก่อนหลังของโหนดหรือเมือง เพื่อให้ได้เส้นทางที่ระยะทางที่สั้นที่สุด ปัญหานี้มีความยุ่งยากในการลำดับการเลือก เนื่องจากจำนวนเมืองที่มีมาก ขณะเวลาที่จะใช้ในการแก้ปัญหาก็ยิ่งมากเป็นทวีคูณ อย่างไรก็ตามระยะทางไม่ใช่ปัจจัยเดียวที่ส่งผลต่อ Cost ในการเดินทาง และเส้นทางที่สั้นที่สุดอาจไม่ใช่เส้นทางที่มี Cost น้อยที่สุด

การหาเส้นทางที่สั้นที่สุด ต้องอาศัยหลักการทางคณิตศาสตร์ เช่น Branch and Bound Algorithm หรือ Integer Linear programming (Gondran and Minoux, 2008) Dynamic programming เป็นต้น ปัญหาการหาเส้นทางที่สั้นที่สุดอยู่ในประเภท NP-hard หรือ Non-polynomial-completeness ซึ่งปัญหาที่ยากต่อการหาค่าคำตอบ หรือผลลัพธ์ และใช้เวลาในการค้นหานาน จึงได้มีการคิดค้นวิธีการหาคำตอบที่มีรูปแบบไม่แน่นอน (Stochastic search หรือ Approximation optimization algorithms) ได้แก่ ซิมมูเลเทดแอนนีลลิ่ง (Simulated annealing: SA) ทาบูเสิร์จ (Taboo search: TS) นิวรอลเน็ทเวิร์ค (Neural network: NN) จีเนติคอัลกอริทึม (Genetic algorithms: GA) พาร์ทิเคิลสวอร์มออพติไมเซชั่น (Particle swarm optimization: PSO) และแอนท์โคโลนีออพติไมเซชั่น (Ant colony optimization: ACO) วิธีการเหล่านี้ จัดการกับปัญหาที่มีความซับซ้อนโดยเฉพาะ (Pettie, 2004; Schrijver, 2004)

ระบบแผนที่ทั้ง Google Maps Here Maps Apple Maps หรืออื่นๆ ไม่ได้ใช้อัลกอริทึมเดียว อาจมีการปรับแล้วปรับอีก เพื่อเพิ่มประสิทธิภาพสำหรับแข่งขันในตลาด Apple Maps คนใช้ยิ่งมากก็ยิ่งมีประสิทธิภาพมากขึ้น แสดงว่า อัลกอริทึมของ Apple Maps อาจทำงานบนพื้นฐานวิธีระบบมดก็เป็นได้ ส่วน Google Maps ที่ใช้ค้นหาเส้นทาง บางทีก็ได้เส้นทางเดียว (Dijkstra's Algorithm) บางครั้งก็มีเส้นทางสำรองให้เลือก (ระบบมด) การผสมผสานทั้งสองอัลกอริทึมก็เป็นได้ และบางแผนที่ให้เลือกกรณีต้องการเส้นทางสำรอง แปลว่า ถ้าไม่คำนวณด้วย Dijkstra's Algorithm แล้วได้เส้นทางที่ดีที่สุด อาจเลือกเส้นทางสำรองอาจได้เส้นทางที่ไม่ได้ดีที่สุดเลยก็เป็นได้ (Sniedovich, 2006; Thorup, 2004; Williams, 2014)

วัตถุประสงค์

การประยุกต์ใช้ไดจ์คสตร้าอัลกอริทึม เพื่อหาเส้นทางที่สั้นที่สุด โดยทดสอบระยะทางจากมหาวิทยาลัยหอการค้าไทย ถึง ปากซอยประชาสงเคราะห์ 23

ประโยชน์ที่ได้รับ

สามารถประยุกต์ใช้ไดจ์คสตร้าอัลกอริทึมเพื่อหาเส้นทางที่สั้นที่สุด ในเส้นทางอื่นๆ เพื่อประหยัดเวลา พลังงานเชื้อเพลิง และรักษาสิ่งแวดล้อมอีกด้วย

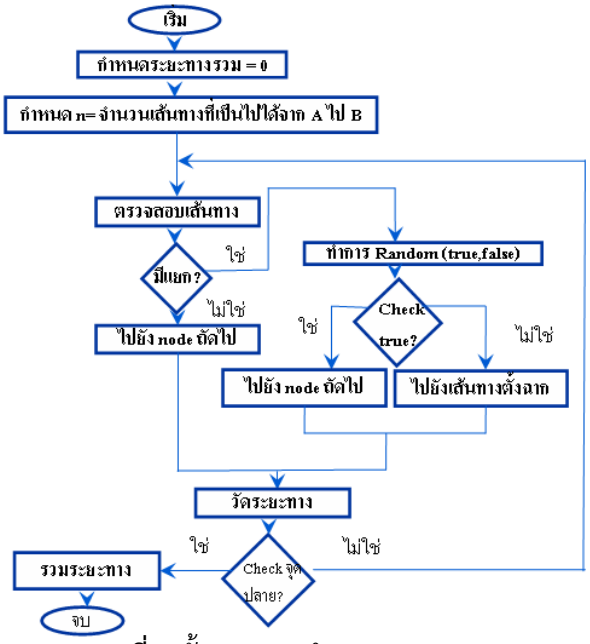
วิธีการดำเนินการวิจัย

1. การออกแบบแบบจำลอง

แบ่งเป็น 3 ส่วน ได้แก่ การประยุกต์ใช้ Dijkstra Algorithm การคำนวณระยะทาง และการกำหนดเส้นทางที่เป็นไปได้ทั้งหมด รวมถึงข้อกำหนดทิศทางของเส้นทางที่ทำการจำลอง

1.1 การประยุกต์ใช้ Dijkstra Algorithm

การคำนวณหาเส้นทางเป็นการประยุกต์จาก Shortest path ในส่วนของการเคลื่อนที่ระหว่าง node นำไปสู่ผลลัพธ์ในการแสดงผล และ Algorithm มีการทำงานแสดงในรูปที่ 1

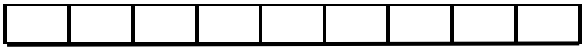


รูปที่ 1 ขั้นตอนการทำงานของ Algorithm

จากรูปที่ 1 แสดงขั้นตอนการทำงานโดยเริ่มแรกจะทำการกำหนดค่าเริ่มต้นจากต้นทางไปยังปลายทางให้มีระยะทางรวมเท่ากับ 0 จากนั้นกำหนด n หรือตัว Object ให้เท่ากับจำนวนของเส้นทางที่เป็นไปได้จาก A ไป B เมื่อตั้งค่าแล้วจะเริ่มทำได้ตรวจสอบเส้นทางตั้งแต่ node เริ่มแรกโดยการตรวจสอบว่าเป็นแยกหรือไม่ ถ้าไม่จะทำการไปยัง node ถัดไป แต่ถ้าเป็นแยกจะทำการสุ่ม true กับ false เมื่อทำการสุ่มจะทำการตรวจสอบว่าที่ทำการสุ่มนั้นได้ true หรือไม่ ถ้าเป็น true จะไปยัง node ถัดไป ถ้าไม่ใช่ก็คือการสุ่มแล้วได้ false จะไปยังเส้นที่ตั้งฉาก ไม่ว่าจะเจอแยกหรือไม่ เมื่อไปยัง node ถัดไปแล้วจะทำการวัดระยะทาง หลังจากวัดระยะทางแล้วจะทำการตรวจสอบว่า node ที่ไปนั้นเป็นจุดปลายทางหรือไม่ ถ้าไม่ใช่จะทำการวนซ้ำไปขั้นตอนการตรวจสอบเส้นทาง และจะทำการวนไปเรื่อยๆ จนกว่าถึงจุดปลายทาง จากนั้นรวมระยะทางทั้งหมดจากต้นทางไปยังไปปลายทางแล้วทำการเทียบระยะทางว่าระยะทางทั้งหมดที่ทำการวัดระยะทางในแต่ละเส้นทางนั้น เส้นทางไหนมีระยะทางสั้นที่สุดเมื่อได้ระยะทางที่สั้นที่สุดแล้วนั้นจะทำการแสดงผล

1.2 การคำนวณระยะทาง

วิธีการคำนวณระยะทางที่ใช้ใน Algorithm โดยการคำนวณระยะทางที่ได้จากการคำนวณจุดพิกเซลที่แมพเข้ากับเส้นทาง แสดงในรูปที่ 2



รูปที่ 2 แมพพิกเซลบนเส้นทาง

จากรูปที่ 2 ด้านบนแสดงการแมพพิกเซลบนเส้นทาง โดยการคำนวณพิกเซลนั้นจะนับจากจำนวนช่องพิกเซลที่แมพบนเส้นทางที่ต้องการหาระยะทาง จากภาพจะได้พิกเซลทั้งหมด 9 พิกเซล โดยการคำนวณนั้นจะคำนวณระยะทางในแต่ละช่วงที่ทำการเคลื่อนที่จากจุดหนึ่งไปยังจุดหนึ่งแล้วเมื่อ Object ถึงจุดปลายทางแล้วจะทำการรวมระยะทางทั้งหมดจากต้นทางไปยังปลายทางโดยระยะทางที่ได้จะเป็นเมตร ซึ่งสมการที่ใช้ในการคำนวณ คือ

ระยะทางของแผนที่ = (จำนวนพิกเซล/72dpi)*2.54 cm.....1

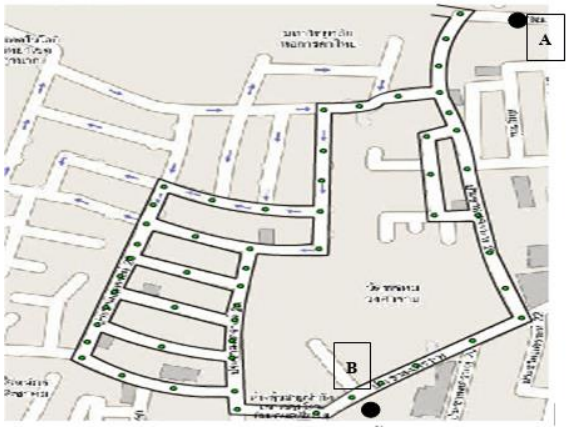
จากสมการที่ 1 จะได้ระยะทางของแผนที่ซึ่งมีหน่วยเป็นเซนติเมตร เมื่อได้ระยะทางในแผนที่ ทำการเปลี่ยนระยะทางในแผนที่ให้มาเป็นระยะทางจริง โดยใช้สมการ ดังนี้

ระยะทางจริง (m) = ระยะทางในแผนที่ (cm) *20.....2

จากสมการที่ 2 เมื่อนำเอาระยะทางในแผนที่มาคำนวณ ผลที่ได้คือ ระยะทางจริงบนแผนที่จริง หน่วยของระยะทางที่ได้ มีหน่วยเป็นเมตร

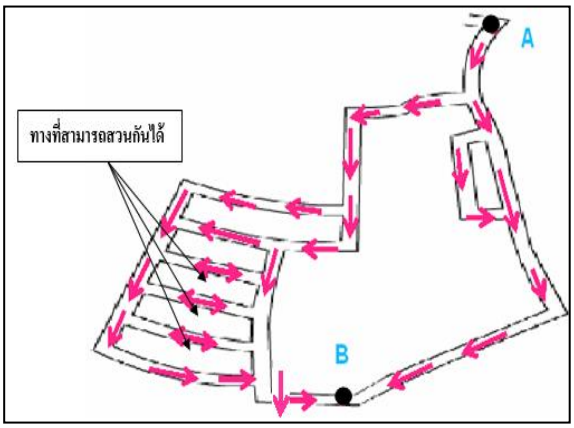
1.3 การกำหนดเส้นทางในการเดินทางที่เป็นไปได้ทั้งหมด

การกำหนดเส้นทางและการเลือกใช้แผนที่ โดยแผนที่ที่ใช้ในโปรแกรมมีมาตราส่วน 1: 2000 ซึ่งในแผนที่มีการวาง node ไว้บนแผนที่ เพื่อให้ Object สามารถวิ่งจาก A ไปยังจุด B ได้ ซึ่งกำหนด node บนเส้นทางทั้งหมด 54 nodes เพื่อทำให้ Object สามารถวิ่งไปตามเส้นทางได้อย่างถูกต้องรวมตามทิศทางที่ถูกกำหนดไว้ บางเส้นทางสามารถเดินรถทางเดียว ทั้งนี้เพื่อให้ Object สามารถเดินทางไปยังเส้นทางต่างๆ ที่มี node กำหนดทั้งทางตรงและทางแยก แสดงในรูปที่ 3



รูปที่ 3 การกำหนด node ตามเส้นทางต่างๆ จุดตั้งต้น (A) ไปยังจุดปลายทาง (B)

จากรูปที่ 3 ในส่วนของเส้นทางที่สามารถเดินรถสองทาง เมื่อ Object เดินทางมายังเส้นทางนั้นแล้วจะไม่มีการวิ่งย้อนกลับได้ แสดงในรูปที่ 4

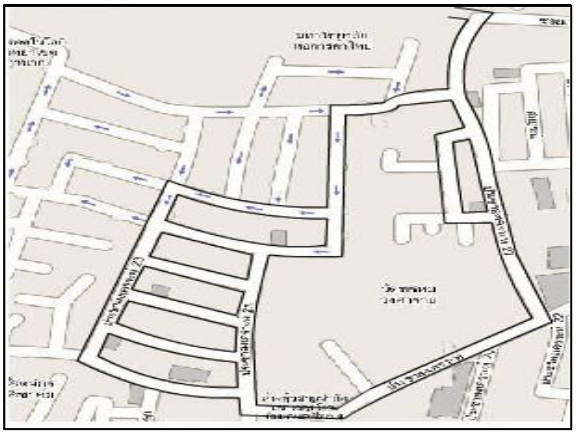


รูปที่ 4 แผนที่จำลองเส้นทางที่ให้ object วิ่งสวนกันได้

จากรูปที่ 4 จะเห็นได้ว่าบางเส้นทางเป็นทางที่สามารถสวนกันได้ แต่บางเส้นทางไม่สามารถสวนได้ เนื่องจากทิศทางที่กำหนดนั้นอิงจากเส้นทางบนแผนที่จริง

1.3.1 แผนที่จำลอง

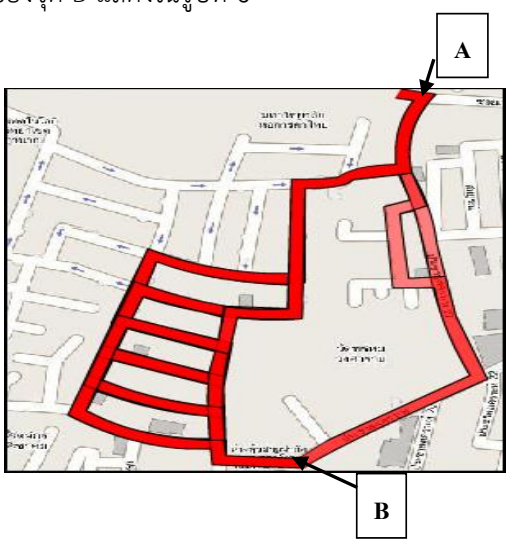
ส่วนของแผนที่จำลองที่นำมาใช้ในระบบ เพื่อใช้ในการแสดงผลที่ต้องการ แสดงเส้นขอบเขตของแผนที่ว่าสามารถเดินทางไปเส้นทางใดได้บ้าง แสดงในรูปที่ 5



รูปที่ 5 เส้นทางจำลอง

1.3.2 การกำหนดสีของเส้นทาง

ขั้นตอนนี้เป็นการกำหนดสีบนเส้นทางที่แสดงหลังจากที่มีการกำหนดเส้นทางให้ object ถึงจุดหมายปลายทางทุก node และจะทำการเทียบระยะทางที่สั้นที่สุด เมื่อได้ระยะที่สั้นที่สุดแล้วจะแสดงสีบนเส้นทางที่ได้เลือกพร้อมกับระยะทางที่สั้นที่สุด โดยเส้นทางที่เลือกนั้นเป็นเส้นทางที่สั้นที่สุด โดยกำหนดไว้มีทั้งหมด 20 เส้นทาง ซึ่ง 20 เส้นทางนี้ คือเส้นทางที่เป็นไปได้จากจุด A ไปยังจุด B แสดงในรูปที่ 6



รูปที่ 6 เส้นทางทั้งหมดที่สามารถไปได้

จากรูปที่ 6 ด้านบนจะเห็นถึงสีที่แสดงบนแผนที่ สีที่แตกต่างกันเนื่องจากการซ้อนทับของสีในแต่ละเส้นทาง ซึ่งยิ่งเส้นทางที่มีการซ้อนทับกันมากสีจะเข้มมากขึ้นตามจำนวนของเส้นทางที่ซ้อนทับ

2. การทำงานของโปรแกรม

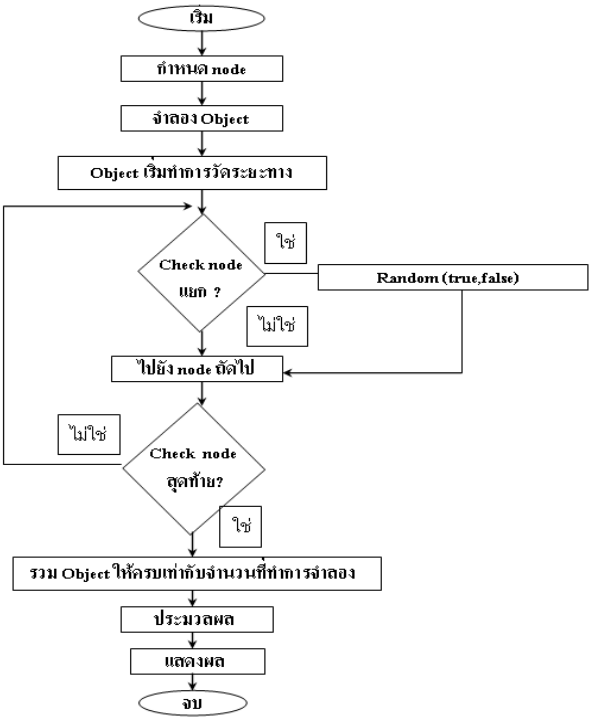
2.1 การออกแบบหน้าจอแสดงผล

การออกแบบหน้าจอแสดงผลจากโปรแกรม มีส่วนแสดงระยะทางความเป็นไปได้ของแผนที่ทั้งหมด 20 เส้นทาง และส่วนที่แสดงระยะทางที่สั้นที่สุด แสดงในรูปที่ 7

รูปที่ 7 หน้าจอแสดงผลจากโปรแกรม

2.2 ขั้นตอนการทำงานของโปรแกรม

ขั้นตอนการทำงานของโปรแกรม มี 3 ขั้นตอน แสดงในรูปที่ 8



รูปที่ 8 ขั้นตอนการทำงานของโปรแกรม

จากรูปที่ 8 การทำงานของโปรแกรมเริ่มจากกำหนด node จากนั้นจำลอง object แล้วเริ่มวัดระยะทาง มีรายละเอียดดังนี้

2.2.1 ระบบจำลองจะเริ่มทำการจำลอง node เพื่อใช้ในการคำนวณหาเส้นทาง 54 node แล้วจำลอง Object ขึ้นมาเพื่อใช้คำนวณระยะทางในแต่ละ node ที่วิ่งผ่านตั้งแต่ node แรกจนถึง node ปลายทาง

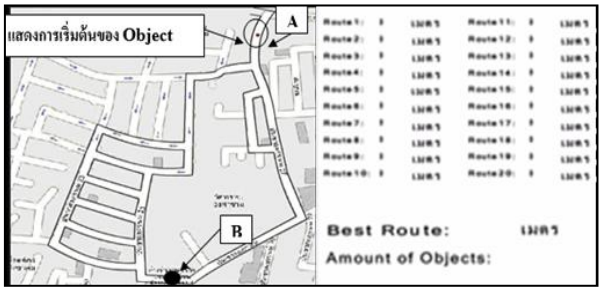
2.2.2 เมื่อ Object เริ่มวิ่งจะทำการตรวจสอบ node แรกที่เจอว่าเป็นทางแยกหรือไม่ ถ้าเป็นแยกจะทำการ Random (true, false) ว่าจะไปทางไหนเมื่อเลือกเส้นทางได้แล้ว Object จะเดินไปยัง node ต่อไป

2.2.3 เมื่อ Object ถึงจุดหมายแล้ว จะหยุดรอให้ครบทุก Object แล้วแสดงผลเป็นระยะทางในแต่ละเส้นทางที่ทำการวิ่งในแต่ละเส้นทาง แล้วเลือกเส้นทางที่สั้นที่สุดโดยการแสดงสีบนเส้นทางที่เลือกและระยะทางของเส้นทางนั้น

ผลการวิจัยและอภิปรายผล

1. ผลการทำงานของโปรแกรม

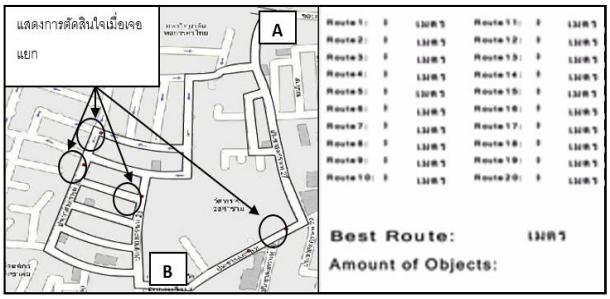
การกำหนดการเดินทางในแผนที่จำลอง จากแผนที่จำลองกำหนดจุดต้นทาง A และปลายทาง B แล้วกำหนดจำนวน Object เท่ากับจำนวนเส้นทางที่เป็นไปได้ทั้งหมดจากต้นทางไปยังปลายทางของแผนที่จำลอง ซึ่งในกรณีนี้จำนวนเส้นทางที่เป็นไปได้ทั้งหมด เท่ากับ 20 เส้นทาง จากการทดสอบการทำงานของ Object โดยโปรแกรมกำหนดให้เดินทางไปตามทางในแผนที่จำลอง พร้อมกับการวัดระยะทางในแต่ละช่วงที่ทำการเดินทางระหว่าง node แสดงในรูปที่ 9



รูปที่ 9 การแสดงตำแหน่งต้นทางของ Object

2. ผลการตัดสินใจเมื่อถึงแยก

เมื่อ Object เริ่มเดินทางไปยัง node ถัดไป โปรแกรมจะทำการตรวจสอบว่า node ถัดไปมี 1 node หรือ 2 node ถ้าโปรแกรมทดสอบว่าเป็น 1 แสดงว่าไม่เป็นทางแยก แต่ถ้าเป็น 2 แสดงว่าเป็นทางแยก เมื่อตรวจสอบแล้วว่า node ถัดไปมี 1 node จะทำการเดินทางตรงไปยัง node ถัดไป แต่ถ้าทำการตรวจสอบว่า node ถัดไปมี 2 node จะทำการตัดสินใจโดยการ Random (true, false) การ Random นั้นจะทำทุกครั้งเมื่อ node ถัดไปมี 2 node แยก ถ้า Random ได้ true จะทำการเดินทางตรงไปยัง node ถัดไป แต่ถ้า false จะทำการเลี้ยวไปยังเส้นทางที่ตั้งฉาก แสดงในรูปที่ 10

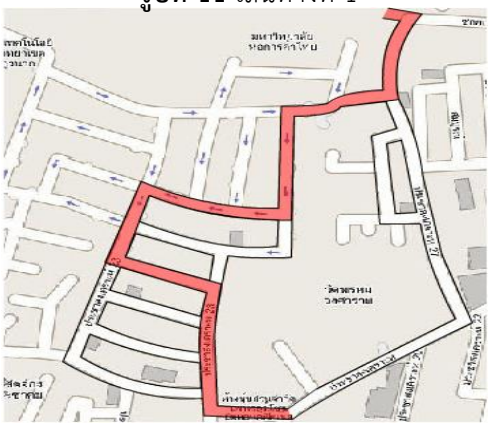


รูปที่ 10 การตัดสินใจเมื่อถึงแยก

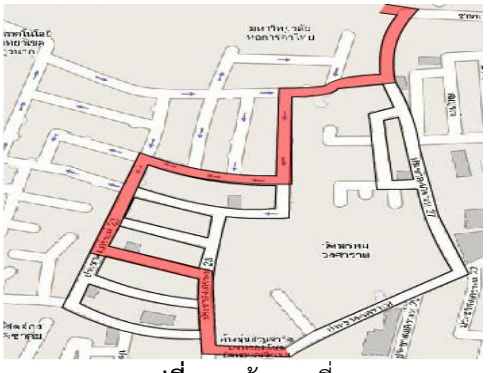
3. ผลการวัดระยะทาง

เมื่อ Object ตัวแรกมาถึง node สุดท้าย โปรแกรมจะทำการคำนวณระยะทางทั้งหมดของ Object แรกทั้งหมดตั้งแต่จุดต้นทางไปยังปลายทาง ทั้งนี้โปรแกรมจะทำงานจนครบ 20 Objects แล้วทำการเปรียบเทียบระยะทางระหว่าง 20 Objects แล้วโปรแกรมจะแสดงผลออกมา ซึ่งมีเส้นทางที่สามารถเป็นไปได้ทั้งหมด 20 เส้นทาง แสดงในรูปที่ 11-30

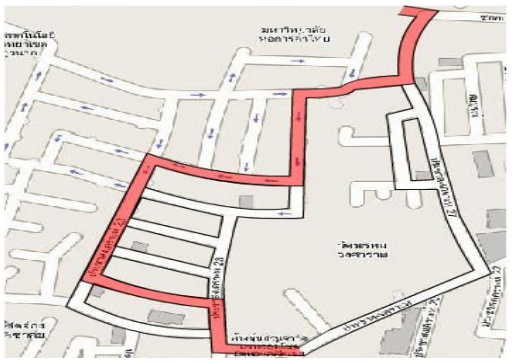
รูปที่ 11 เส้นทางที่ 1



รูปที่ 12 เส้นทางที่ 2

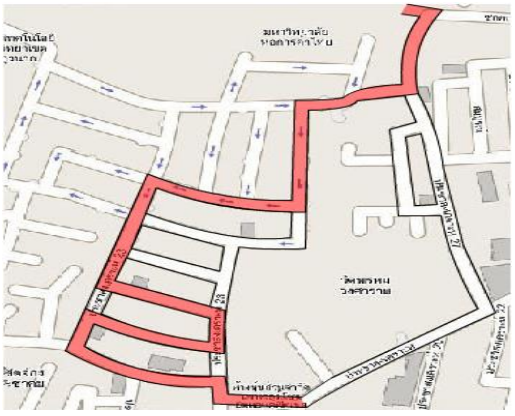


รูปที่ 13 เส้นทางที่ 3

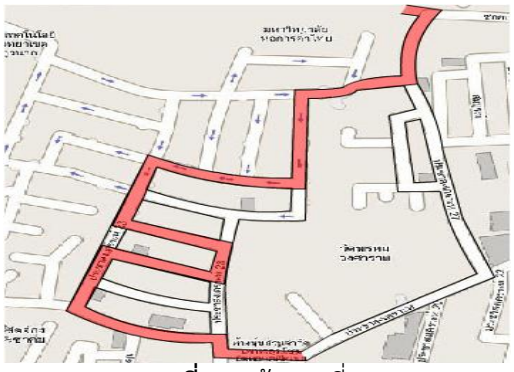


รูปที่ 14 เส้นทางที่ 4

รูปที่ 15 เส้นทางที่ 5



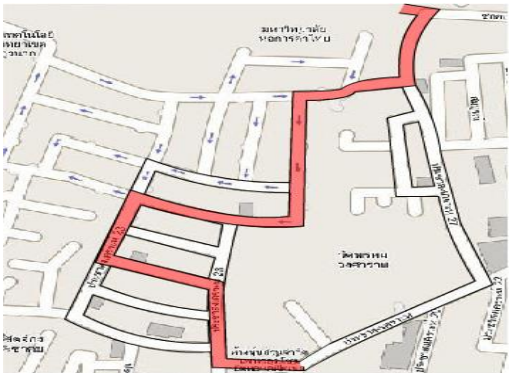
รูปที่ 16 เส้นทางที่ 6



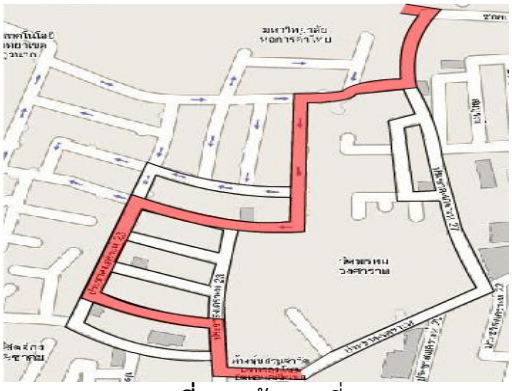
รูปที่ 17 เส้นทางที่ 7

รูปที่ 18 เส้นทางที่ 8

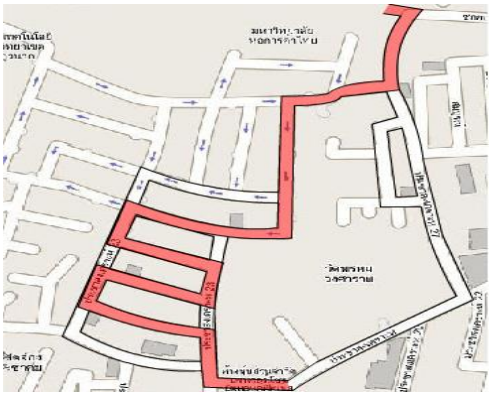
รูปที่ 19 เส้นทางที่ 9



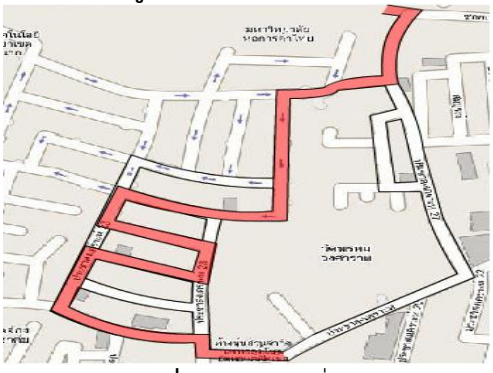
รูปที่ 20 เส้นทางที่ 10



รูปที่ 21 เส้นทางที่ 11

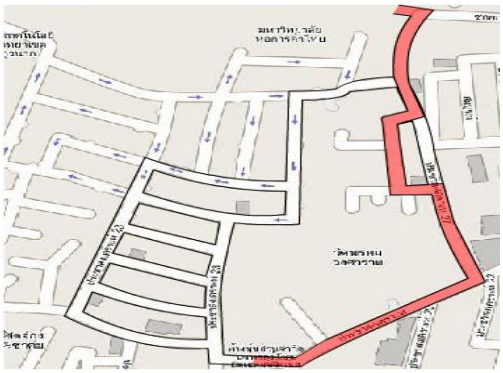


รูปที่ 22 เส้นทางที่ 12

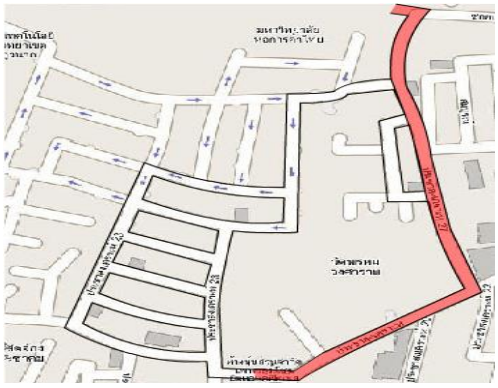


รูปที่ 23 เส้นทางที่ 13

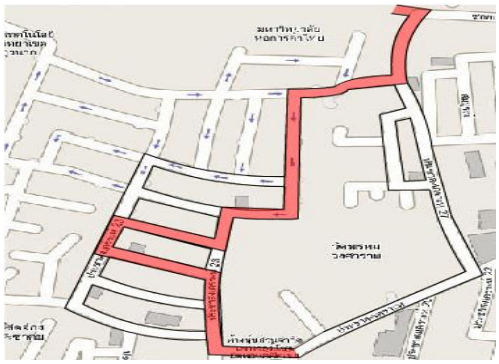
รูปที่ 24 เส้นทางที่ 14



รูปที่ 25 เส้นทางที่ 15

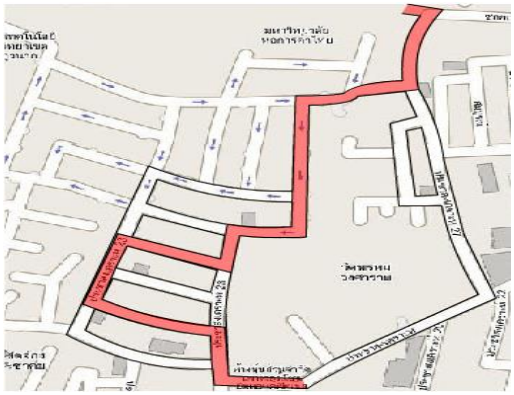


รูปที่ 26 เส้นทางที่ 16

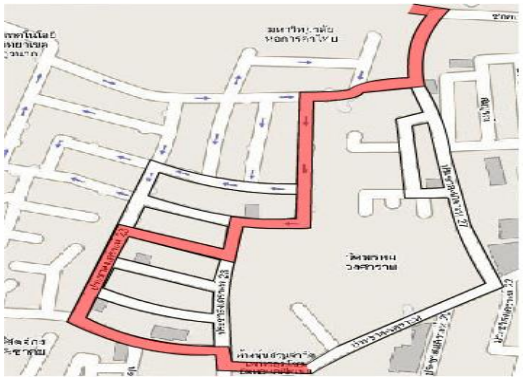


รูปที่ 27 เส้นทางที่ 17

รูปที่ 28 เส้นทางที่ 18

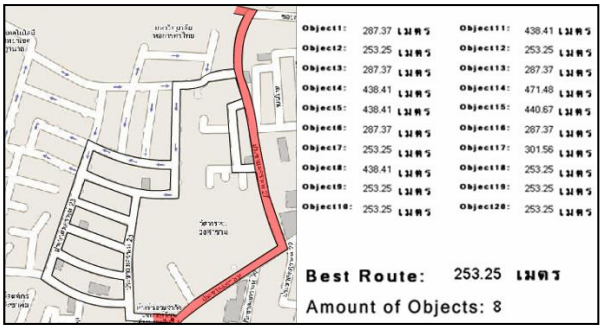


รูปที่ 29 เส้นทางที่ 19



รูปที่ 30 เส้นทางที่ 20

จากรูปที่ 11-30 แสดงให้เห็นถึงเส้นทางที่สามารถไปยังจุดหมายได้ทั้งหมด ซึ่งมีเส้นทางที่สั้นที่สุดแสดงในรูปที่ 31

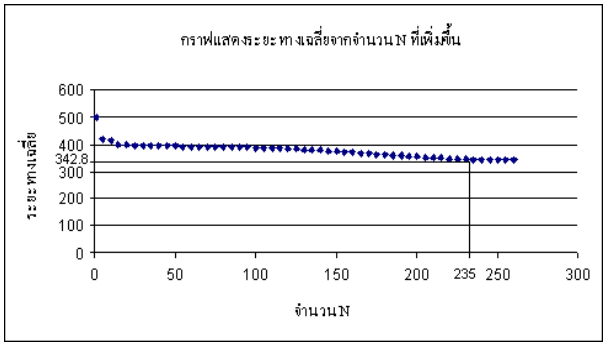


รูปที่ 31 ผลหลังจากทุก Object ถึงจุดสุดท้าย

4. ผลการทดสอบจำนวน Object ที่เหมาะสมกับแบบจำลอง

เนื่องจากโอกาสที่ Object จะไม่ไปในเส้นทางที่สั้นที่สุด จึงได้มีการทดลองหาจำนวน Object ที่เหมาะสม เพื่อที่จะได้เส้นทางที่สั้นที่สุดแน่นอนในการทำงานของโปรแกรมทุกครั้ง ในส่วนนี้เป็นการทดลองการเพิ่มจำนวนของ n หรือ Object เพื่อที่จะหาจำนวนที่เหมาะสมที่จะนำมาใช้กับแบบจำลองได้อย่างมีประสิทธิภาพ ซึ่งวิธีการหา Object ที่เหมาะสมกับแบบจำลองนั้นทำโดยการเพิ่มจำนวน Object ทีละ 5 แล้วนำระยะทางที่ได้ในแต่ละ Object มารวมกันแล้วหารด้วยจำนวนของ Object ซึ่งผลที่ได้จะเป็นระยะทางเฉลี่ย จากการทดลองเพิ่ม Object จนกระทั่งระยะทางเฉลี่ยรวมคงที่ แสดงในรูปที่ 32

จากรูปที่ 32 จำนวน Object ที่เหมาะสมกับแบบจำลองตั้งแต่ 235 Object เป็นต้นไป จะได้เส้นทางที่สั้นที่สุดอย่างน้อย 1 Object



รูปที่ 32 ระยะทางเฉลี่ยเมื่อจำนวน N เพิ่มขึ้น

5. ผลการทดสอบความเร็วของโปรแกรม

ในการทำงานของโปรแกรมแสดงที่ 24-30 เฟรมต่อวินาที ซึ่งได้ผลออกมาอย่างต่อเนื่อง แต่ถ้าปรับความเร็วแล้วให้สูงขึ้น เพื่อต้องการให้การแสดงผลเร็วขึ้นนั้น โปรแกรมจะไม่ทำงานต่อไป เนื่องจากการแสดงผลที่ใช้ความเร็วที่สูงนั้นจำเป็น ต้องใช้เครื่องคอมพิวเตอร์ที่มีประสิทธิภาพในการประมวลผลที่สูงตามไปด้วย

สรุปผลการวิจัย

จากการจำลองการหาเส้นทางของระบบจำลองเส้นทางที่ได้จะเป็นเส้นทางที่สั้นที่สุด โดยเส้นทางที่ได้นั้นไม่ได้อิงเวลาในการทำงานของโปรแกรมหรือการจราจร ซึ่งผลที่ได้จะเป็นผลที่ได้จากการคำนวณเส้นทางที่สั้นที่สุดจริง แล้วใช้โปรแกรมแสดงให้เห็นว่าเส้นทางที่เลือกนั้นเป็นเส้นทางที่สั้นที่สุด โดยแสดงระยะทางจริง และสีบนเส้นทางที่เลือกเพื่อให้ทราบว่าเส้นทางไหนเป็นเส้นทางที่สั้นที่สุด โดยการที่จะได้ระยะทางที่สั้นที่สุดนั้นจริงๆ จะขึ้นอยู่กับจำนวนของ Object ด้วย เมื่อเพิ่มจำนวน Object เข้าไปในแผนที่ที่ทำการจำลองนั้นมากขึ้นเท่าไหร่ก็จะมีโอกาสที่จะได้เส้นทางที่สั้นที่สุดมากขึ้น ดังนั้น Dijkstra อัลกอริธึมมีความเหมาะสมที่จะนำมาใช้ในการค้นหาเส้นทาง เนื่องจากเปรียบเทียบกับอัลกอริธึมอื่นๆ แล้ว พบว่าใช้เวลาในการทำงาน (Running Time) น้อยที่สุด (Thorup, 2004)

ข้อเสนอแนะ

สำหรับการศึกษาต่อไป เสนอแนะให้พัฒนา Dijkstra Algorithm ในการค้นหาเส้นทางที่สั้นที่สุด โดยพิจารณาค่าน้ำหนักที่แตกต่างกัน มีการกำหนด Priority ในการค้นหาเฉพาะในส่วนของจุดเริ่มต้นและปลายทาง โดยให้จุดเริ่มต้นและปลายทางต่างเป็นจุดกึ่งกลางของวงกลมของตนเอง และกำหนด Priority เฉพาะในวงกลม เมื่อได้จุดต่อไปที่เชื่อมกับจุดเริ่มต้น ก็ทำเช่นนี้ไปเรื่อยๆ จนอยู่ในเขตรัศมีเดียวกับปลายทาง เพื่อเปรียบเทียบเวลาในการประมวลผล

กิตติกรรมประกาศ

ขอขอบคุณ คุณทรงวุฒิ อนุตรโพธิ์แก้ว ที่ช่วยในการสำรวจเส้นทาง เพื่อใช้ในการจำลองเส้นทางที่สั้นที่สุด

เอกสารอ้างอิง

Denardo, E.V. 2003. **Dynamic Programming: Models and Applications.** Mineola, New York: Dover Publications.

Gondran, M. and Minoux, M. 2008. **Graphs, Dioids and Semirings: New Models and Algorithms.** Springer Science & Business Media. Chapter 4.

Kroger, M. 2005. Shortest multiple disconnected path for the analysis of entanglements in two- and three-dimensional polymeric systems. **Computer Physics Communications.** 168 (168): 209–232. doi:10.1016/j.cpc. 2005. 01.020.

Pouly, M. and Kohlas, J. 2011. **Generic Inference: A Unifying Theory for Automated Reasoning.** John Wiley & Sons. Chapter 6. Valuation Algebras for Path Problems.

Pettie, S. 2004. A new approach to all-pairs shortest paths on real-weighted graphs. **Theoretical Computer Science.** 312(1):

47–74. doi:10.1016/s0304-3975(03)00402-x.

Schrijver, A. 2004. **Combinatorial Optimization-Polyhedra and Efficiency. Algorithms and Combinatorics.** 24. Springer. Here: vol.A, sect.7.5b, p. 103.

Sniedovich, M. 2006. Dijkstra's algorithm revisited: the dynamic programming connexion (PDF). **Journal of Control and Cybernetics.** 35 (3): 599–620.

Thorup, M. 2004. Integer priority queues with decrease key in constant time and the single source shortest paths problem. **Journal of Computer and System Sciences.** 69 (3): 330–353. doi:10.1016/j.jcss.2004.04.003.

Williams, R. 2014. **Faster all-pairs shortest paths via circuit complexity.** Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC 14). New York: ACM. pp. 664–673. arXiv: 1312. 6680. doi:10.1145/2591796.2591811.MR 3238994.