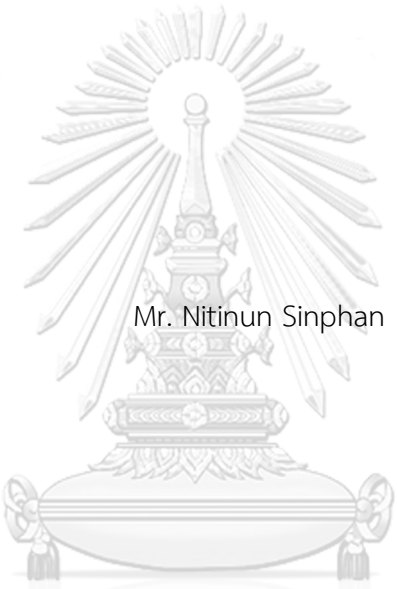
การอพยพฝูงชนพร้อมผู้นำในสถานีรถไฟด้วยการเรียนรู้แบบเสริมกำลัง

นายนิธินันท์ สินพันธุ์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2565

Crowd Evacuation with Leaders in Railway Station using Reinforcement Learning

Mr. Nitinun Sinphan



A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering in Electrical Engineering
Department of Electrical Engineering
FACULTY OF ENGINEERING
Chulalongkorn University
Academic Year 2022

หัวข้อวิทยานิพนธ์	การอพยพฝูงชนพร้อมผู้นำในสถานีรถไฟด้วยการเรียนรู้แบบเสริมกำลัง
โดย	นายนิธินันท์ สินพันธุ์
สาขาวิชา	วิศวกรรมไฟฟ้า
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.ลัญฉกร วุฒิสิทธิกุลกิจ
อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม	รองศาสตราจารย์ ดร.ณัฏฐ์ ลีละวัฒน์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.สุพจน์ เตชวรสินสกุล)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.วิทยากร อัศดรวิเศษ)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(รองศาสตราจารย์ ดร.ลัญฉกร วุฒิสิทธิกุลกิจ)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม
(รองศาสตราจารย์ ดร.ณัฏฐ์ ลีละวัฒน์)

..... กรรมการภายนอกมหาวิทยาลัย
(ผู้ช่วยศาสตราจารย์ ดร.สุวิทย์ ภูมิฤทธิกุล)

นิธินันท์ สินพันธุ์ : การอพยพฝูงชนพร้อมผู้นำในสถานีรถไฟด้วยการเรียนรู้แบบเสริมกำลัง. (Crowd Evacuation with Leaders in Railway Station using Reinforcement Learning) อ.ที่ปรึกษาหลัก : รศ. ดร.ลัญฉกร วุฒิสิทธิกุลกิจ, อ.ที่ปรึกษาร่วม : รศ. ดร.ณัฏฐ์ ลีละวัฒน์

วิทยานิพนธ์ฉบับนี้เสนอการประยุกต์ใช้การเรียนรู้แบบเสริมกำลังเทคนิคที่มีชื่อว่า การเพิ่มประสิทธิภาพนโยบายใกล้เคียง เพื่อฝึกฝนเอเจนต์ให้เป็นผู้นำการอพยพผ่านชุดเครื่องมือเอมแอลเอเจนต์ของโปรแกรมยูนิตี แบบจำลองแรงทางสังคมถูกใช้เพื่อนำเสนอการปรับเปลี่ยนพลวัตของผู้อพยพและอัลกอริทึมเอสตาร์ถูกใช้เพื่อค้นหาเส้นทางอพยพที่สั้นที่สุดของผู้อพยพแต่ละคนจากตำแหน่งเริ่มต้นไปยังทางออก ประสิทธิภาพการอพยพถูกประเมินโดยจำนวนผู้อพยพโดยเฉลี่ยที่ไม่สามารถออกจากสถานีรถไฟฟ้าทดลองได้ตามเวลาที่กำหนดภายใต้สถานการณ์การอพยพที่แตกต่างกัน จากผลการทดสอบพบว่า การอพยพฝูงชนด้วยผู้นำที่ถูกฝึกด้วยการเรียนรู้แบบเสริมกำลังสามารถอพยพผู้อพยพทั้งหมดออกจากสถานีรถไฟฟ้าทดลองได้ตามเวลาที่กำหนด ทั้งนี้เวลาที่ใช้ในการอพยพจะเร็วหรือช้าขึ้นอยู่กับตำแหน่งของผู้อพยพด้วยเช่นกัน

สาขาวิชา วิศวกรรมไฟฟ้า ลายมือชื่อนิสิต .....
ปีการศึกษา 2565 ลายมือชื่อ อ.ที่ปรึกษาหลัก .....
ลายมือชื่อ อ.ที่ปรึกษาร่วม .....

6370457121 : MAJOR ELECTRICAL ENGINEERING
KEYWORD:

Nitinun Sinphan : Crowd Evacuation with Leaders in Railway Station using Reinforcement Learning . Advisor: Assoc. Prof. LUNCHAKORN WUTTISITTIKULKIJ Co-advisor: Assoc. Prof. NATT LEELAWAT

This thesis proposes the application of a reinforcement learning technique called proximal policy optimization to train the agent to become an evacuation leader via the unity ml-agents toolkit. The social force model is used to present the modification of evacuee dynamics and the a-star algorithm is used to find the shortest evacuation path of each evacuee from the initial location to the exit. Evacuation efficiency is assessed by the average number of evacuees who cannot leave the train station for the specified time under different evacuation situations. Simulation results show that crowd evacuation with leaders trained by reinforcement learning can evacuate all evacuees from the train station for the specified time. However, the time it takes to evacuate will be faster or slower depending on the location of the evacuees as well.

Field of Study: Electrical Engineering
Academic Year: 2022

Student's Signature .....
Advisor's Signature .....
Co-advisor's Signature .....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สามารถสำเร็จลุล่วงไปได้ดีข้าพเจ้าขอกราบขอบพระคุณ รองศาสตราจารย์ ดร.ลัญฉกร วุฒิสิทธิกุลกิจ อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก และ รองศาสตราจารย์ ดร.ณัฏฐ์ ลีละวัฒน์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม เนื่องด้วยความช่วยเหลือและการเอาใจใส่เป็นอย่างยิ่ง และได้ให้คำแนะนำและข้อคิดเห็นต่าง ๆ ในการดำเนินการทำวิจัยตลอดมา

ขอกราบขอบพระคุณบิดามารดา และญาติพี่น้องของข้าพเจ้า ที่ได้ให้กำลังใจและความห่วงใยเสมอมาซึ่งเป็นแรงผลักดันที่ช่วยให้งานวิจัยสำเร็จไปได้ด้วยดี

ขอบคุณพี่ ๆ น้อง ๆ ในห้องปฏิบัติการวิจัยโทรคมนาคมทุกท่าน โดยเฉพาะนายพฤกษ์ สระศรีทอง รวมทั้งคนอื่น ๆ ที่ไม่ได้กล่าวถึง ณ ที่นี้ในการช่วยเหลือด้านต่าง ๆ ทั้งการให้คำแนะนำในการวิจัย การจัดทำรูปเล่มวิทยานิพนธ์ การดำเนินการสอบวิทยานิพนธ์ การใช้ชีวิตในระดับบัณฑิตศึกษา ซึ่งทำให้การศึกษาในจุฬาลงกรณ์มหาวิทยาลัยเต็มไปด้วยคุณค่าและความประทับใจยิ่ง ซึ่งทำให้ข้าพเจ้าได้รับประสบการณ์ที่ดีเป็นอย่างมาก

นิธินันท์ สินพันธุ์

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญรูป

หน้า

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญ

ปัจจุบันระบบรางเป็นระบบขนส่งมวลชนหลักที่สำคัญของแทบทุกประเทศทั่วโลก ด้วยเพราะระบบรางมีประสิทธิภาพสูง สามารถขนส่งผู้คนได้ครั้งละจำนวนมากต่อเที่ยว ช่วยประหยัดต้นทุนในการเดินทางและลดปัญหาการจราจรที่ติดขัดได้เป็นอย่างดี นอกจากนี้ผู้โดยสารได้รับความสะดวกสบายในการเดินทางและตรงต่อเวลา ในประเทศไทยมีการใช้งานระบบรางอย่างกว้างขวาง โดยเฉพาะในพื้นที่กรุงเทพมหานครและปริมณฑล

1.1.1 ประวัติความเป็นมาของระบบรางในประเทศไทย

จุดเริ่มต้นของระบบรางในประเทศไทยเกิดขึ้นครั้งแรกเมื่อพระบาทสมเด็จพระจุลจอมเกล้าเจ้าอยู่หัวทรงก่อตั้งการรถไฟแห่งประเทศไทย (State Railway of Thailand: SRT) [1] ขึ้นในเดือนตุลาคม พ.ศ. 2433 เพื่อเป็นหนึ่งในโครงสร้างพื้นฐานหลักด้านคมนาคม โดยทางรถไฟสายกรุงเทพ - นครราชสีมา (สายตะวันออกเฉียงเหนือ) เป็นสายแรก และต่อมามีการขยายเพิ่มทางรถไฟสายเหนือ สายใต้ สายตะวันตก สายตะวันออก ในปัจจุบันระบบรางมีระยะทางที่เปิดการเดินรถแล้วรวมทั้งสิ้น 4,507 กิโลเมตร แบ่งเป็นเส้นทางรถไฟทางเดี่ยว 4,097 กิโลเมตร เส้นทางรถไฟทางคู่ 303 กิโลเมตร การรถไฟแห่งประเทศไทยยังมีโครงการขยายเส้นทางให้เป็นทางคู่เพื่อลดเวลาการเดินทาง เพิ่มความจุตู้สินค้าและตู้โดยสาร รวมทั้งลดการใช้พลังงาน เนื่องจากได้รับการยอมรับในระดับสากลว่าเป็นวิธีการขนส่งที่มีต้นทุนการขนส่งต่อหน่วยต่ำที่สุด

การรถไฟฟ้าขนส่งมวลชนแห่งประเทศไทย ถูกก่อตั้งขึ้นเมื่อ พ.ศ. 2535 เพื่อมีหน้าที่ในการให้บริการรถไฟฟ้าขนส่งมวลชนในกรุงเทพมหานครและปริมณฑล (Mass Rapid Transit Authority of Thailand: MRTA) [2] ในวันที่ 5 ธันวาคม พ.ศ.2542 เปิดให้บริการรถไฟฟ้าบีทีเอส (Bangkok Transit System: BTS) [3] สายสุขุมวิทหรือสายสีเขียว (สถานีหมอชิต-อ่อนนุช) และสายสีลมหรือสายเขียวเข้ม (สถานีสนามกีฬาแห่งชาติ-สะพานตากสิน) เป็นระบบรถไฟฟ้ายกระดับแห่งแรกของประเทศไทยดำเนินการโดยบริษัท ระบบขนส่งมวลชนกรุงเทพ จำกัด (มหาชน) (Bangkok Mass Transit System Public Company Limited: BTSC) ในวันที่ 3 กรกฎาคม พ.ศ. 2547 เริ่มเปิดให้บริการรถไฟฟ้ามหานคร (Metropolitan Rapid Transit: MRT) [4] สายเฉลิมรัชมงคลหรือสายสีน้ำเงิน (สถานีบางซื่อ-หัวลำโพง) เป็นระบบรถไฟฟ้าใต้ดินสายแรกของประเทศไทยดำเนินการโดย

บริษัท ทางด่วนและรถไฟฟ้ากรุงเทพ จำกัด (มหาชน) (Bangkok Expressway and Metro Public Company Limited: BEM) ในวันที่ 15 พฤษภาคม พ.ศ. 2552 เปิดให้บริการส่วนต่อขยายรถไฟฟ้าบีทีเอสสายสีลมหรือสายเขียวเข้ม (สถานีสะพานตากสิน-วงเวียนใหญ่) [3] ในวันที่ 23 สิงหาคม พ.ศ. 2553 เริ่มเปิดให้บริการรถไฟฟ้าแอร์พอร์ตเรลลิงก์ (สถานีสุวรรณภูมิ-พญาไท) [5] อย่างเป็นทางการ เพื่อเชื่อมท่าอากาศยานสุวรรณภูมิกับใจกลางกรุงเทพมหานคร โดยการรถไฟแห่งประเทศไทยเป็นเจ้าของและตั้งแต่ปี พ.ศ. 2564 ดำเนินการเดินรถโดย บริษัท เอเชีย เอรา วัน จำกัด ในวันที่ 12 สิงหาคม พ.ศ.2554 เปิดให้บริการส่วนต่อขยายรถไฟฟ้าบีทีเอสสายสุขุมวิทหรือสายเขียว (สถานีอ่อนนุช-แบริ่ง) [3] ในปีพ.ศ. 2556 ได้ทยอยเปิดให้บริการส่วนต่อขยายรถไฟฟ้าบีทีเอสสายสีลมหรือสายเขียวเข้ม (สถานีวงเวียนใหญ่-บางหว้า) [3] ในวันที่ 6 สิงหาคม พ.ศ.2559 เปิดให้บริการ รถไฟฟ้ามหานคร สายฉลองรัชธรรมหรือสายสีม่วง (สถานีคลองบางไผ่-เตาปูน) [6] ในปีพ.ศ. 2560 ได้เปิดให้บริการส่วนต่อขยาย 2 สายด้วยกันได้แก่ รถไฟฟ้าบีทีเอส สายสุขุมวิทหรือสายสีเขียว (สถานีแบริ่ง-สำโรง) และไฟฟ้ามหานคร สายเฉลิมรัชมงคลหรือสายสีน้ำเงิน (สถานีบางซื่อ-เตาปูน) 6 ธันวาคม พ.ศ.2561 เปิดบริการส่วนต่อขยายรถไฟฟ้าบีทีเอส สายสุขุมวิทหรือสายสีเขียว (สถานีสำโรง-เคหะฯ) ในปี พ.ศ.2562 ทยอยเปิดบริการส่วนต่อขยายรถไฟฟ้าบีทีเอส สายสุขุมวิทหรือสายสีเขียว (สถานีหมอชิต-คูคต) และไฟฟ้ามหานคร สายเฉลิมรัชมงคลหรือสายสีน้ำเงิน (สถานีหัวลำโพง-หลักสอง) และ (สถานีเตาปูน-ท่าพระ) วันที่ 16 ธันวาคม พ.ศ.2563 เริ่มเปิดบริการเปิดให้บริการ รถไฟฟ้าบีทีเอส สายสีทอง (สถานีกรุงธนบุรี-สถานีคลองสาน) [3] ซึ่งเป็นระบบรถไฟฟ้ารางเบารูปแบบไร้คนขับโดยใช้รางนำทาง มีผิวสัมผัสระหว่างล้อและทางวิ่งเป็นยาง ปัจจุบันรถไฟฟ้าชานเมืองสายสีแดงเข้ม (สถานีบางซื่อ-รังสิต) และ รถไฟฟ้าชานเมืองสายสีแดงอ่อน (สถานีบางซื่อ-ตลิ่งชัน) [7] มีจำนวน 10 สถานี และ 3 สถานีตามลำดับซึ่งเปิดให้บริการเรียบร้อยแล้ว ในอนาคตจะมีการเปิดให้บริการรถไฟฟ้าสายสีชมพู (สถานีศูนย์ราชการนนทบุรี-มีนบุรี) [8] และ รถไฟฟ้าสายสีเหลือง (สถานีลาดพร้าว-สำโรง) [9] ซึ่งเป็นระบบรถไฟฟ้าไร้คนขับโดยใช้รางเดี่ยว แบบวิ่งคร่อมคานทางวิ่ง (straddle-beam monorail) และรถไฟฟ้ามหานคร สายสีส้ม (สถานีศูนย์วัฒนธรรมฯ – มีนบุรี) [10]

สถานีกลางบางซื่อ หรือชื่อปัจจุบันคือ สถานีกลางกรุงเทพอภิวัฒน์ [11] เป็นสถานีขนาดใหญ่ที่มีพื้นที่ใช้สอย 304,000 ตารางเมตร สามารถรองรับผู้โดยสารได้ประมาณ 624,000 คนต่อวันและมีชานชาลาทั้งสิ้น 24 ชานชาลาเพื่อรองรับรถไฟทางไกล รถไฟความเร็วสูง รถไฟฟ้าชานเมือง ดังนั้นสถานีกลางบางซื่อจึงเป็นจุดศูนย์รวมการเดินทางของระบบรางทั้งหมดของประเทศและยังเชื่อมต่อกับการเดินทางรูปแบบอื่น ๆ ตัวอย่างเช่น รถไฟฟ้าสายสีน้ำเงิน รถไฟฟ้าสายสีเขียว รถไฟฟ้าสายสีม่วง

และสถานีขนส่งผู้โดยสารกรุงเทพฯ นอกจากนี้ยังเชื่อมต่อกับท่าอากาศยานดอนเมือง ท่าอากาศยานสุวรรณภูมิและท่าอากาศยานนานาชาติอู่ตะเภา ซึ่งเป็นโครงการการเชื่อมต่อสามสนามบิน

1.1.2 ปัญหาการอพยพผู้โดยสารออกจากสถานีรถไฟในสถานการณ์ฉุกเฉิน

ในสถานีรถไฟที่มีขนาดใหญ่เป็นจุดศูนย์กลางการเชื่อมต่อเส้นทางรถไฟหลายสายและระบบขนส่งมวลชลอื่นๆ จะมีปริมาณผู้โดยสารคับคั่งมากกว่าสถานีอื่น ๆ โดยเฉพาะอย่างยิ่งในชั่วโมงเร่งด่วน สถานีรถไฟเหล่านี้จะมีปริมาณและความหนาแน่นของผู้โดยสารมากเป็นพิเศษ ระบบการเข้าออกของผู้โดยสารในสถานีเหล่านี้จึงมีการออกแบบพื้นที่ภายในสถานีให้เปนที่โลง ไมมีซอกหรือมุมที่ลับตาเจาหนาที่รักษาความปลอดภัย เพื่อให้ผู้โดยสารสามารถเข้าออกสถานีได้อย่างรวดเร็ว และปลอดภัย [12, 13] อย่างไรก็ตาม ในกรณีเกิดเหตุฉุกเฉินที่ไม่อาจคาดการณ์ได้ล่วงหน้า เช่น ไฟไหม้ การขู่วางระเบิด การจลาจล หรืออุบัติเหตุอันเกิดจากความผิดพลาดของระบบรถไฟเอง การอพยพฝูงชนออกจากสถานีในสถานการณ์เหล่านี้ให้ปลอดภัยจึงเป็นปัญหาที่ยาก ท้าทาย และมีความสำคัญยิ่ง เพราะนอกจากช่องทางออกของสถานีอาจมีจำนวนลดลงแล้ว ทางออกบางแห่งอาจมีสิ่งกีดขวาง ทำให้การเคลื่อนที่ของฝูงชนช้าลง ผู้โดยสารเองก็มีพฤติกรรมตื่นตระหนก แก่งแย่งกันอย่างไม่เป็นระเบียบ มีการกระทบกระทั่งกันระหว่างผู้โดยสาร หรือแม้แต่สะดุดหกล้มจนเหยียบทับกัน ดังนั้นการอพยพผู้โดยสารในสถานีบีรถไฟอย่างมีประสิทธิภาพและรวดเร็วจึงเป็นสิ่งที่ขาดไม่ได้ในการรับประกันความปลอดภัยในชีวิตและทรัพย์สินของผู้โดยสารในกรณีฉุกเฉิน

เนื่องจากการฝึกอพยพเพื่อทดสอบแผนการอพยพในสถานีรถไฟนั้นแทบจะไม่มี ถึงแม้จะมีการฝึกอพยพภายในสถานีรถไฟนั้นก็เป็นการฝึกกับคนจำนวนจำกัด การฝึกอพยพคนภายในสถานีรถไฟกับคนจำนวนมากนั้นเป็นเรื่องที่ยากจะเป็นไปได้ มีต้นทุนสูงและยังรบกวนการเดินทางของผู้ที่ใช้งาน ดังนั้นการสร้างแบบจำลองการอพยพเพื่อทดสอบแผนอพยพในสถานีรถไฟหรือพื้นที่ขนาดใหญ่จึงเป็นวิธีที่ง่ายกว่าในการประเมินและปรับปรุงความปลอดภัย

ในช่วงสองทศวรรษที่ผ่านมามีการนำแบบจำลองแรงทางสังคม (The social force model) มาประยุกต์ใช้กับการอพยพฝูงชนจากห้อง สถานีรถไฟ และสนามกีฬาขนาดใหญ่ [14-16] ต่อมามีการนำแบบจำลองแรงทางสังคมมาปรับแต่งสำหรับการอพยพฝูงชนด้วยผู้นำ [17] การอพยพฝูงชนด้วยป้ายสัญญาณ [18] ซึ่งเป็นการอพยพภายในชั้นเดียว

เพื่อที่จะพัฒนาและปรับปรุงการอพยพฝูงชน การเรียนรู้แบบเสริมกำลังจึงถูกเสนอมาเพื่อพัฒนาการจำลองการอพยพฝูงชน ซึ่งถูกเสนอโดย Wang et al.[19] Zheng และ Liu [20] เสนอ

การไล่ระดับนโยบายเชิงลึกสำหรับการจำลองการอพยพฝูงชนตามเส้นทางที่ได้วางแผนไว้ (Deep deterministic policy gradient for path planning-based crowd simulation) ทุกวันนี้การเพิ่มประสิทธิภาพนโยบายใกล้เคียง (The proximal policy optimization algorithm) [21] เป็นหนึ่งในอัลกอริธึมหลักในการไล่ระดับนโยบาย (Policy gradient/actor-critic field) ซึ่งถูกนำไปใช้โดยโอเพนเอไอ (OpenAI) และ ยูนิตีเทคโนโลยี (Unity Technology) อย่างไรก็ตามงานวิจัยที่มีอยู่เป็นการนำการเรียนรู้ของเครื่องและการเรียนรู้แบบเสริมกำลังมาใช้กับการแก้ปัญหาการหาเส้นทางการอพยพ ดังนั้นยังมีช่องว่างในการวิจัยที่ใช้การเรียนรู้แบบเสริมกำลังกับการจำลองการอพยพฝูงชนอีกมากมาย

จากงานวิจัยที่ได้กล่าวมา ผู้วิจัยจึงสร้างโปรแกรมจำลองการอพยพด้วยคอมพิวเตอร์โดยอาศัยแบบจำลองแรงทางสังคมและการค้นหาแบบเอสตาร์สำหรับการจำลองพฤติกรรมการเดินเท้าและการค้นหาเส้นทางที่สั้นที่สุดในการอพยพของผู้อพยพภายในสถานีรถไฟฟ้าทดลองตามลำดับ โดยโปรแกรมแบบจำลองมี 2 โปรแกรม ในส่วนแรกเป็นแบบจำลองการอพยพฝูงชนสองมิติด้วยโปรแกรมไพทอน ซึ่งเป็นโปรแกรมที่พัฒนาขึ้นเองทั้งหมด ส่วนที่สองเป็นแบบจำลองการอพยพฝูงชนสามมิติด้วยโปรแกรมยูนิตี เพื่อเปรียบเทียบประสิทธิภาพการอพยพของวิธีการที่มีอยู่กับวิธีการที่จะนำเสนอ

ดังนั้นงานวิจัยนี้ จึงเสนอการเรียนรู้แบบเสริมกำลัง (Reinforcement Learning) [22, 23] มาประยุกต์ใช้เพื่อปรับกลยุทธ์การอพยพพร้อมผู้นำโดยใช้ Unity ML-agents [23, 24] สำหรับฝึกอบรมเอเจนต์ให้กลายเป็นผู้นำอพยพ การประเมินผลการอพยพจะใช้ในรูปแบบของการพิจารณาจำนวนผู้อพยพที่ไม่สามารถออกจากพื้นที่ที่ทำการจำลองภายในระยะเวลาที่กำหนด

1.2 วัตถุประสงค์

1. การใช้การเรียนรู้แบบเสริมกำลัง (Reinforcement Learning) สำหรับการหากลยุทธ์การอพยพฝูงชนพร้อมผู้นำการอพยพที่เหมาะสม เพื่อลดจำนวนคนหลงเหลือภายในสถานีรถไฟฟ้า
2. การสร้างโปรแกรมแบบจำลองสองและสามมิติเปรียบเทียบกัน โดยใช้แบบจำลองแรงทางสังคม (Social force model) เพื่อจำลองสถานการณ์การอพยพของคนในสถานีรถไฟฟ้าทดลอง

1.3 ขอบเขตการศึกษา

1. การออกแบบกลยุทธ์การอพยพฝูงชนของผู้นำโดยการใช้การเรียนรู้แบบเสริมกำลัง

2. จำลองสถานการณ์การอพยพฝูงชนออกจากสถานีรถไฟฟ้าตัวอย่าง มีสถานการณ์ดังนี้ การอพยพฝูงชนโดยปราศจากผู้นำ และการอพยพฝูงชนพร้อมผู้นำ
3. สร้างโปรแกรมจำลองการเคลื่อนที่ของฝูงชนด้วยแบบจำลองแรงทางสังคม แบบสองมิติด้วยไพทอน และแบบสามมิติด้วยยูนิตีเปรียบเทียบกัน

1.4 ประโยชน์ที่คาดว่าจะได้รับ

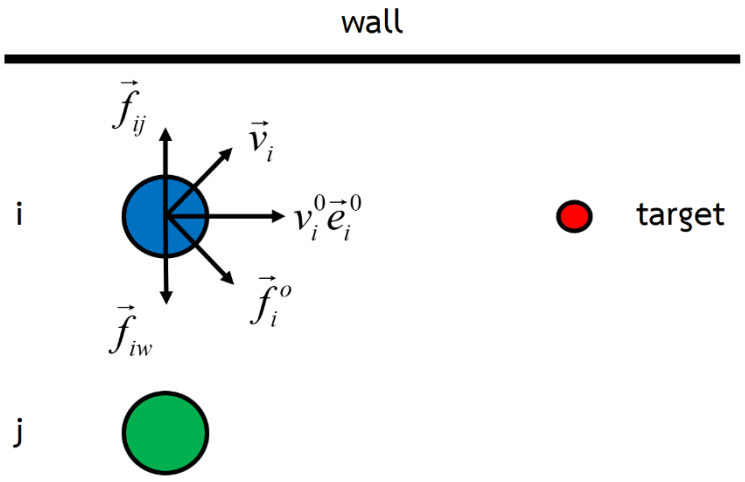
1. ได้กลยุทธ์ของผู้นำอพยพสำหรับการอพยพฝูงชนเพื่อลดจำนวนคนหลงเหลือภายในสถานีรถไฟฟ้า
2. สามารถพัฒนารูปแบบการอพยพฝูงชนออกจากสถานีรถไฟฟ้าโดยเร็วและปลอดภัย
3. โปรแกรมแบบจำลองสองมิติและสามมิติเพื่อใช้ในจำลองการอพยพคนจากสถานีรถไฟฟ้า โดยสามารถประเมินผลกระทบต่อผู้โดยสารเชิงเวลาในการอพยพ และรายงานผลกระทบเชิงเวลาที่เกิดขึ้นได้ทั้งหมด เปรียบเทียบตามการปรับเปลี่ยนวิธีการในการอพยพ

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

บทนี้กล่าวถึงทฤษฎีที่เกี่ยวข้องต่าง ๆ ที่ใช้ในการจัดทำงานวิจัยเรื่อง การประยุกต์ใช้การเรียนรู้แบบเสริมกำลังกับการอพยพฝูงชนพร้อมผู้นำในสถานีรถไฟ มีทฤษฎีที่เกี่ยวข้อง ดังนี้ หัวข้อที่ 2.1 แบบจำลองแรงทางสังคม (Social force model) หัวข้อที่ 2.2 การค้นหาแบบเอสตาร์ (A* algorithm) หัวข้อที่ 2.3 การออกแบบแบบจำลองการอพยพฝูงชนบนโลกสามมิติ หัวข้อที่ 2.4 การออกแบบแบบจำลองการอพยพฝูงชนบนระนาบสองมิติ หัวข้อที่ 2.5 การเรียนรู้แบบเสริมกำลัง (Reinforcement Learning) หัวข้อที่ 2.6 การเพิ่มประสิทธิภาพนโยบายใกล้เคียง (Proximal Policy Optimization; PPO) หัวข้อที่ 2.7 มาตรฐานการออกแบบเพื่อปองกันอัคคีภัย

2.1 แบบจำลองแรงทางสังคม (Social force model)

หัวข้อย่อยนี้จะกล่าวถึงการสร้างแบบจำลองการอพยพคนออกจากสถานีรถไฟโดยใช้แบบจำลองการจราจรคนเดินเท้าเพื่อศึกษาพฤติกรรมของผู้อพยพภายในสถานี ซึ่งจำเปนต้องศึกษาปัจจัยหลักที่มีผลตอการดำเนินงานของแบบจำลองอยางถูกตอง โดยการเคลื่อนที่และพฤติกรรมของการเดินของผู้อพยพภายในสถานี จะขึ้นอยู่กับแบบจำลองแรงทางสังคม



รูปที่ 2-1 แผนภาพแบบจำลองทางสังคม

แบบจำลองแรงทางสังคมที่มีพื้นฐานตอเนื่องจากกลศาสตรนิวตัน ซึ่งถูกเสนอโดย Dirk Helbing และ Peter Molnar, [25] เพื่อจำลองการเคลื่อนที่คนเดินเท้าและวัดแรงจูงใจภายในของ

บุคคลในการเคลื่อนที่ โดยรูปแบบของแรงมีหลายชนิด ทั้งแรงผลัก แรงเสียดทาน แรงดึงดูด และแรงในการขับเคลื่อน เปนตน โดยแรงทั้งหมดนี้มีผลทำให้คนเดินเท้าเกิดการเคลื่อนตัวและเปลี่ยนทิศทางการเดิน ต่อมาในปี ค.ศ. 2000 แบบจำลองแรงทางสังคมสำหรับการหลบหนีด้วยความตื่นตระหนก [14] เกิดขึ้นเป็นครั้งแรก แบบจำลองแรงทางสังคมประกอบไปด้วย 3 แรง คือ แรงที่ต้องการ (the desired force), $\vec{f}_i^{\,0}$ แรงปฏิสัมพันธ์ระหว่างคนเดินเท้า (The interaction force between pedestrians), $\vec{f}_{ij}$ และแรงปฏิสัมพันธ์ระหว่างคนเดินเท้ากับกำแพง (The interaction force between pedestrian and wall) โดยแรงแรกเป็นแรงของความเร่งไปสู่ความเร็วการเคลื่อนที่ที่ต้องการ ส่วนแรงที่สองและแรงที่สามเป็นแรงที่สะท้อนว่าคนเดินเท้ารักษาระยะห่างจากคนเดินเท้าและผนังหรือสิ่งกีดขวางอื่นๆ ซึ่งสมการการเคลื่อนที่สำหรับคนเดินเท้าใดๆเขียนได้ดังนี้

$$m_i \frac{d\vec{v}_i}{dt} = \vec{f}_i^{\,0} + \sum_{j(\neq i)} \vec{f}_{ij} + \sum_{w} \vec{f}_{iw} \qquad (2\text{-}1)$$

โดยที่ - m_i คือมวลของคนเดินเท้าใดๆ

- $\vec{v}_i$ คือความเร็วเดินจริง ณ เวลานั้น

แรงที่ต้องการ (The desired force), $\vec{f}_i^{\,0}$ คือความเต็มใจของคนเดินเท้าที่ต้องการเร่งหรือลดความเร็วเพื่อที่จะเคลื่อนที่ด้วยความเร็วที่ต้องการ ซึ่งสมการของแรงที่ต้องการเขียนได้ดังนี้

$$\vec{f}_i^{\,0} = m_i \frac{v_i^0(t)\vec{e}_i^{\,0}(t) - \vec{v}_i(t)}{\tau_i} \qquad (2\text{-}2)$$

โดยที่ - $v_i^0(t)$ คือความเร็วที่ต้องการ

- $\vec{e}_i^{\,0}(t)$ คือทิศทางที่ต้องการซึ่งเป็นเวกเตอร์หนึ่งหน่วยที่ชี้ไปยังตำแหน่งเป้าหมาย
- $\vec{v}_i(t)$ คือความเร็วคนเดินถนน ณ เวลานั้น
- τ_i เป็นมาตราส่วนเวลาลักษณะเฉพาะเพื่อปรับความเร็วจริงของคนเดินเท้า

แรงปฏิสัมพันธ์ระหว่างคนเดินเท้า (The interaction force between pedestrians), $\vec{f}_{ij}$ คือแรงปฏิสัมพันธ์ระหว่างคนเดินเท้ากับคนเดินเท้าที่ประกอบไปด้วยแรงจิตวิทยาทางสังคม (The socio-psychological force), $\vec{f}_s^{\,ij}$ และแรงทางกายภาพ (The physical force), $\vec{f}_p^{\,ij}$ แรงจิตวิทยาทางสังคมเป็นแนวโน้มทางจิตวิทยาของคนเดินเท้าสองคนที่จะอยู่ห่างจากกันโดยแรงปฏิสัมพันธ์ที่

ผลักดัน ซึ่งแรงทางกายภาพประกอบด้วย “แรงของร่างกาย (body force)” เพื่อต่อต้านการกดทับของร่างกาย และ “แรงเสียดทานแบบเลื่อน (sliding friction force)” เพื่อขัดขวางการเคลื่อนที่แนวสัมผัสสัมพัทธ์ ซึ่งสมการของแรงปฏิสัมพันธ์ระหว่างคนเดินเท้ามีดังนี้

$$\vec{f}_{ij} = \vec{f}_s^{ij} + \vec{f}_p^{ij} \qquad (2\text{-}3)$$

$$\vec{f}_s = A_i \exp[(r_{ij} - d_{ij}) / B_i]\vec{n}_{ij} \qquad (2\text{-}4)$$

$$\vec{f}_p = kg(r_{ij} - d_{ij})\vec{n}_{ij} + Kg(r_{ij} - d_{ij})\vec{n}_{ij}\Delta v_{ij}^t \vec{t}_{ij} \qquad (2\text{-}5)$$

โดยที่ - A_i คือ ค่าความแรงของพลังผลักดันทางสังคม (นิวตัน)

- B_i คือ ค่ามาตราส่วนระยะทางลักษณะเฉพาะของแรงผลักทางสังคม (เมตร)

- k คือ ค่าสัมประสิทธิ์แรงเสียดทานเลื่อน (กิโลกรัม/เมตร วินาที)

- K คือ ค่าสัมประสิทธิ์การกดทับของร่างกาย (กิโลกรัม/วินาที วินาที)

- r_{ij} คือผลรวมของรัศมีคนเดินเท้า

- d_{ij} คือระยะห่างระหว่างคนเดินเท้าสองคน

- $\vec{n}_{ij} = (n_{ij}^1, n_{ij}^2) = (\vec{p}_i - \vec{p}_j) / d_{ij}$ คือเวกเตอร์หนึ่งหน่วยที่ชี้จากคนหนึ่งไปยังอีกคนหนึ่งซึ่งเป็นแรงเสียดทานเลื่อนที่เกิดขึ้นในทิศทางปกติ

- $\vec{p}_i$ เป็นเวกเตอร์ตำแหน่งของคนเดินเท้า

- $\vec{t}_{ij} = (-n_{ij}^2, n_{ij}^1)$ คือเวกเตอร์หนึ่งหน่วยซึ่งคือความเสียดทานเลื่อนที่เกิดขึ้นในแนวสัมผัส

- $g(r_{ij} - d_{ij})$ เป็นฟังก์ชันที่ถูกกำหนดโดย

$$g(x) = \begin{cases} 0, & if \quad x < 0, \\ x, & if \quad x \geq 0. \end{cases} \qquad (2\text{-}6)$$

ต่อมามีการประยุกต์ใช้แบบจำลองแรงทางสังคมกับการอพยพฝูงชนด้วยผู้นำ [15] ส่งผลให้สมการแบบจำลองแรงทางสังคมของคนเดินเท้าที่ติดตามผู้นำอพยพมีแรงนำทาง (The navigational force) เพิ่มเติมขึ้นมาดังนี้

$$m_i \frac{d\vec{v}_i}{dt} = \vec{f}_i^{\,0} + \sum_{j(\neq i)} \vec{f}_{ij} + \sum_{w} \vec{f}_{iw} + \vec{f}_{ig} \qquad (2\text{-}7)$$

โดยที่ $\vec{f}_{ig}$ คือแรงนำทาง

$$\vec{f}_{ig} = \alpha_i \cdot m_i \cdot [-b_1 \frac{\vec{x}_i(t) - \vec{x}_g(t)}{\tau_i^2} - b_2 \frac{\vec{v}_i(t) - \vec{v}_g(t)}{\tau_i}] \qquad (2\text{-}8)$$

โดยที่ - b_1 และ b_2 เป็นเป็นสัมประสิทธิ์ซึ่งสะท้อนถึงน้ำหนักของข้อเสนอแนะการนำทาง ซึ่งมีค่าเท่ากับ 0.05

- α_i เป็นค่าคงที่เท่ากับ 1 เมื่อผู้ติดตามสามารถรับข้อมูลของผู้นำอพยพได้
- $\vec{x}_i(t)$ เป็นตำแหน่งของผู้ติดตาม ณ เวลานั้น
- $\vec{x}_g(t)$ เป็นตำแหน่งของผู้นำอพยพ ณ เวลานั้น

จากการประยุกต์ใช้แบบจำลองแรงทางสังคมด้วยการเพิ่มแรงนำทางจะส่งผลให้พฤติกรรมของคนเดินเท้าเปลี่ยนไปเมื่อผู้อพยพพบเห็นผู้นำอพยพ โดยการที่ผู้อพยพจะเดินไปยังผู้นำอพยพเพื่อรับข้อมูลที่ผู้นำต้องการจะสื่อสาร

2.2 การค้นหาแบบเอสตาร์ (A* algorithm)

หัวข้อย่อยนี้จะกล่าวถึงอัลกอริธึมต่าง ๆ ที่เกี่ยวข้องกับการค้นหาเส้นทาง โดยมุ่งเน้นไปทีการค้นหาแบบเอสตาร์ เริ่มแรกเราจะเริ่มต้นด้วยอัลกอริธึมการค้นหาแบบดีที่สุด (Best-First-Search) จากนั้นต่อด้วยอัลกอริธึมไดค์สตรา (Dijkstra's Algorithm) และสุดท้ายอัลกอริธึมเอสตาร์ (A-star's algorithm) ที่เป็นอัลกอริธึมหลักที่ถูกใช้สำหรับการค้นหาเส้นทางและนำทางในงานวิจัยนี้

การค้นหาแบบดีที่สุด เป็นเทคนิคการค้นหาที่ละเอียดถี่ถ้วนในการเดินทางจากจุดเริ่มต้นไปยังจุดหมายปลายทาง โดยพยายามที่จะเสี่ยงค้นหาเส้นทางที่สามารถไปได้ทั้งหมดเพื่อเลือกเส้นทางที่เหมาะสมที่สุดที่จะไปถึงปลายทาง ซึ่งเป็นการผสมผสานระหว่างอัลกอริธึมการค้นหาเชิงลึกและอัลกอริธึมการค้นหาแบบกว้างเข้าด้วยกัน โดยที่แต่ละขั้นของการค้นหา การค้นหาแบบดีที่สุดจะเลือกเอาจุดที่ดีสุดมารวมเป็นเส้นทางที่เหมาะสมที่สุดที่จะไปถึงปลายทาง และการที่จะทราบว่าจุดใดดีที่สุดนี้สามารถทำได้โดยอาศัยฟังก์ชันฮิวริสติก ซึ่งฟังก์ชันฮิวริสติกเป็นการคาดเดาอย่างมีเหตุผลว่าเข้าใกล้เป้าหมายมากน้อยเพียงใด และให้ผลของการวัดนี้ออกมาเป็นคะแนน เพื่อช่วยบอกว่าควรไปในทิศทางใด

การค้าหาแบบไดค์สตรา เป็นเทคนิคการค้าหาเส้นทางที่สั้นที่สุดในการเดินทางจากจุดเริ่มต้นไปยังจุดหมายปลายทาง ในการค้าหาเส้นทางจะหาจุดใกล้เคียงกับจุดเริ่มต้นที่มีระยะทางสั้นที่สุด จะได้เป็นจุดถัดไปในการเชื่อมกับจุดที่ผ่านมา แล้วทำแบบนี้จนกระทั่งถึงจุดหมายปลายทาง ซึ่งจำเป็นที่จะค้นหาไปทุกจุดหรือเพื่อที่จะค้นหาเส้นทางที่สั้นที่สุด โดยจุดอ่อนประการหนึ่งของอัลกอริธึมนี้คือความช้า

การค้นหาแบบเอสตาร์ [26] เป็นอัลกอริธึมที่ใช้สำหรับค้นหาเส้นทางระหว่างจุด 2 จุด โดยมีเป้าหมายเพื่อค้นหาเส้นทางไปยังจุดหมายปลายทางที่กำหนดด้วยต้นทุนน้อยที่สุด การค้นหาแบบเอสตาร์เป็นส่วนขยายของวิธีของไดค์สตราที่สามารถใช้ค้นหาเส้นทางที่สั้นที่สุดได้ รวมกับการค้นหาแบบดีที่สุดที่สามารถเป็นการผสมผสานระหว่างอัลกอริธึมการค้นหาเชิงลึกและอัลกอริธึมการค้นหาแบบกว้างนำทางตัวเองได้เข้าด้วยกัน โดยที่แต่ละขั้นของการค้นหา การค้นหาแบบเอสตาร์จะเลือกเอาจุดที่มีค่าต่ำสุดเพื่อไปทำเส้นทาง ถ้าระหว่างการค้นหาไปแต่ละจุดแล้วพบว่าเส้นทางที่กำลังค้นหาอยู่นั้นมีค่ามากกว่าเส้นทางเก่าที่ได้ทำการค้นหา การค้นหาแบบเอสตาร์จะไม่พิจารณาเส้นทางที่มีค่ามากกว่า แต่จะย้อนกลับมาหาเส้นทางเก่าที่ค้นหามาแล้วหรือกล่าวอีกนัยนึงว่า จะเลือกไปยังเส้นทางที่มีค่าน้อยกว่าแทน กระบวนการนี้จะทำต่อไปเรื่อย ๆ จนกระทั่งถึงจุดหมายปลายทาง ค่าที่บอกว่าการค้นหาเส้นทางแบบเอสตาร์ควรไปเส้นทางใดนั้นคำนวนจากฟังก์ชันฮิวริสติกแบบค่าของระยะทาง $f(x)$ เพื่อที่จะหาลำดับการผ่านจุดระหว่างเส้นทาง โดยฟังก์ชันดังกล่าวเป็นผลรวมของสองฟังก์ชันดังนี้

$$f(x) = g(x) + h(x) \qquad (2\text{-}9)$$

โดยที่ - x คือ จุดถัดไปบนเส้นทาง

- $g(x)$ คือ ค่าต้นทุนของเส้นทางที่คิดจากจุดเริ่มต้นไปยังจุดปัจจุบัน โดยอาจจะมีหลายเส้นทางที่เริ่มจากจุดเริ่มต้นจนถึงจุดปัจจุบัน
- $h(x)$ คือ ค่าประมาณฮิวริสติกที่ประเมินต้นทุนของเส้นทางที่ถูกที่สุดจากจุดปัจจุบันถึงจุดเป้าหมาย
- $f(x)$ คือ ค่าผลรวมของ f และ g ซึ่งเป็นสิ่งที่จะบอกว่าไปจุดใดต่อไป โดยเลือกเอาจุดที่มีค่า $f(x)$ น้อยที่สุด

จากที่ได้กล่าวมาทำให้การค้าหาแบบเอสตาร์คืออัลกอริธึมการค้นหาเส้นทางที่ใช้กันอย่างแพร่หลายในเกมและการจำลอง เนื่องจากมีความยืดหยุ่นสูง รวดเร็ว และมีประสิทธิภาพสูงสุด

2.3 การออกแบบแบบจำลองการอพยพฝูงชนบนโลกสามมิติ

ในหัวข้อย่อยก่อนหน้ากล่าวถึงแบบจำลองแรงทางสังคมสำหรับการจำลองการเคลื่อนที่ของผู้อพยพและอัลกอริธึมเอสตาร์สำหรับการค้นหาเส้นทาง ซึ่งจะถูกนำมาเป็นส่วนหนึ่งของแบบจำลองการอพยพฝูงชนบนโลกสามมิติ อย่างไรก็ดี หัวข้อย่อยนี้จะกล่าวถึงโปรแกรมยูนิตี และอุปกรณ์เสริมต่าง ๆ ที่ถูกนำมาใช้จำลองการอพยพฝูงชนพร้อมผู้นำบนโลกสามมิติเพื่อศึกษาให้เกิดความรู้ความเข้าใจ

รูปที่ 2-2 โปรแกรมยูนิตี

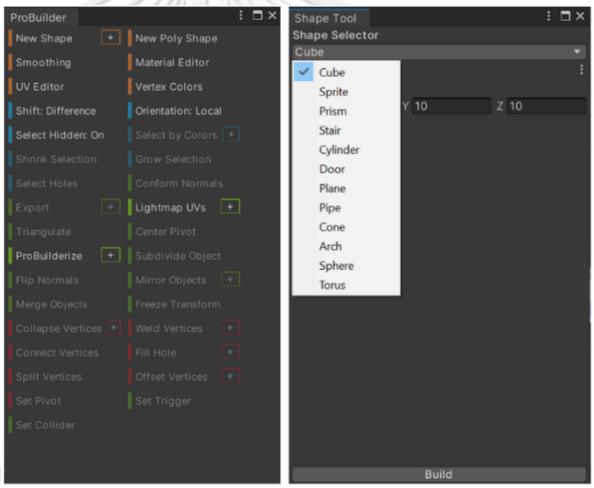
โปรแกรมยูนิตี [27] เป็นซอฟต์แวร์มีความสามารถหลากหลายได้แก่ การสร้างหรือพัฒนาเกม (Game Engine) แบบสองมิติ (2D Game) และสามมิติ (3D Game) หรือการสร้างเออาร์ (AR) และวีอาร์ (VR) ซึ่งรองรับการทำงานบนวินโดวส์ (Windows) ไอโอเอส (iOS) แอนดรอยส์ (Android) และโอคูลัส (Oculus) เป็นต้น และสามารถใช้งานได้ฟรีในเวอร์ชันของนักเรียน นักศึกษา หรือ บุคคล

ทั่วไปภายใต้ข้อตกลงและการใช้งานที่จำกัด หากผู้ใช้งานต้องการการใช้งานที่มากขึ้นจำเป็นต้องเสียค่าใช้จ่ายตามที่ทางบริษัทกำหนด

นอกจากนี้โปรแกรมยูนิตีจะเพิ่มความสามารถของวัตถุนั้น ๆ ด้วยการเขียนภาษาซีชาร์ป (C# language) เพิ่มเติม เช่น การเคลื่อนย้ายตำแหน่งของผู้อพยพและผู้นำผู้อพยพไปยังทางออกตามแบบจำลองแรงทางสังคม การกระจายตำแหน่งเริ่มต้นอย่างกระจายตัวของผู้อพยพและผู้นำผู้อพยพภายในพื้นที่สถานีรถไฟฟ้า แล้วยังมีเครื่องมือและอุปกรณ์เสริมอีก 3 อย่างที่นำมาใช้ในแบบจำลองการอพยพฝูงชนบนโลกสามมิติดังนี้

2.3.1 โปรบิวเดอร์ (ProBuilder)

โปรบิวเดอร์ [27] คืออุปกรณ์เสริมสำหรับสร้าง แก้ไข และกำหนดพื้นผิวเรขาคณิตของแบบจำลอง 3 มิติอย่างง่ายและรวดเร็ว เช่น โครงสร้างสถานีรถไฟฟ้า



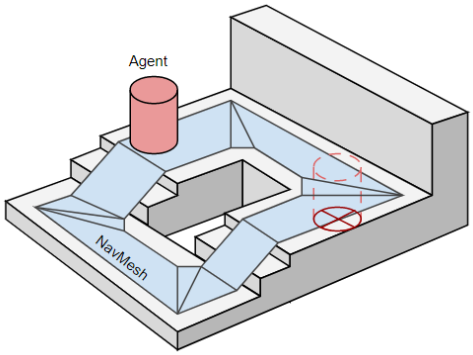
รูปที่ 2-3 อุปกรณ์เสริมโปรบิวเดอร์

2.3.2 การค้นหาเส้นทางและการนำทาง (Pathfinding and Navigation)

การค้นหาเส้นทางและการนำทาง [23, 28] เป็นแนวคิดที่นิยมใช้ในปัญหาการหาเส้นทางที่สั้นที่สุดของเกมสามมิติ ซึ่งกระบวนการค้นหาเส้นทางของอุปกรณ์เสริมนี้สามารถทำได้ด้วยอัลกอริธึมที่มีประสิทธิภาพและเป็นที่นิยมมากที่สุดในปัจจุบันก็คือ การค้นหาแบบเอสตาร์ได้กล่าวในหัวข้อ 2.2 ระบบการค้นหาเส้นทางและการนำทางประกอบด้วยส่วนต่าง ๆ ดังต่อไปนี้

- นาฟเมช (NavMesh) เป็นตาข่ายนำทางที่สร้างขึ้นโดยอัตโนมัติจากฉากที่เก็บพื้นผิวเป็นรูปหลายเหลี่ยม ซึ่งเป็นข้อมูลที่อธิบายพื้นผิวที่เสามารถเดินได้หรือเดินไม่ได้ใน

โลกของเกมและช่วยให้สามารถค้นหาเส้นทางจากที่หนึ่งไปยังอีกที่หนึ่งในโลกของเกมได้



รูปที่ 2-4 นาฟเมช (NavMesh) [28]

- นาฟเมช เอเจนต์ (NavMesh Agent) เป็นองค์ประกอบที่ช่วยให้ตัวละครที่สร้างขึ้นสามารถหลีกเลี่ยงซึ่งกันและกันในขณะที่เคลื่อนที่ไปสู่เป้าหมายได้บนพื้นผิวของนาฟเมช รวมไปถึงรู้วิธีที่จะหลีกเลี่ยงสิ่งกีดขวาง
- นาฟเมช สิ่งกีดขวาง (NavMesh Obstacle) เป็นองค์ประกอบที่ช่วยอธิบายสิ่งกีดขวางที่ตัวละครควรหลีกเลี่ยงขณะเคลื่อนย้ายไปยังเป้าหมาย

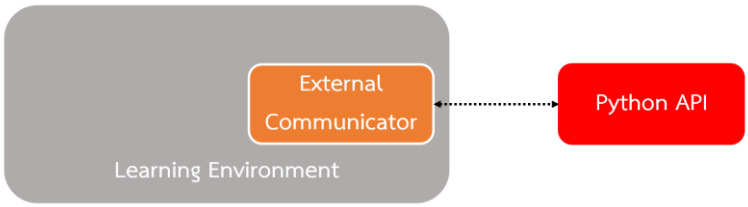
จากงานวิจัย [29] สรุปได้ว่า นาฟเมช (NavMesh) และอัลกอริธึมเอสตาร์ เป็นทางออกที่ดีที่สุดในการแก้ปัญหาการค้นหาเส้นทางอพยพที่สั้นที่สุดสำหรับความต้องการทางอุตสาหกรรมในปัจจุบัน

2.3.3 ML-Agents (Machine Learning Agent)

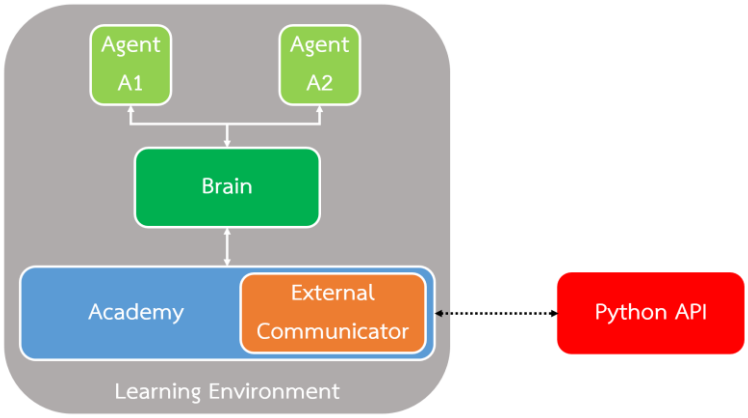
ยูนิตี ML-Agents [23, 24] เป็นอุปกรณ์เสริมแบบโอเพนซอร์สสำหรับโปรแกรมยูนิตีที่ถูกพัฒนาโดยยูนิตีเทคโนโลยี (Unity Technologies) อุปกรณ์เสริมนี้ถูกสร้างขึ้นบนสภาพแวดล้อม OpenAI Gym และสื่อสารระหว่าง Python API และ Unity C# Engine เพื่อสร้างโมเดลการเรียนรู้เชิงลึก นอกจากนี้ยังมีการใช้ไลบรารี Tensorflow กับยูนิตี ML-Agents สำหรับการอนุมานอย่างลึกซึ้งและการฝึกโมเดลผ่านโค้ดซีชาร์ปที่กำหนดขึ้นเอง และสภาพแวดล้อมที่จำลองขึ้นมา โดยอุปกรณ์เสริมนี้จะช่วยให้เกมและการจำลองสามารถใช้เป็นสภาพแวดล้อมสำหรับการฝึกอบรมตัวละครหรือเอเจนต์ด้วย

เทคนิคการเรียนรู้แบบเสริมกำลังที่เรียกว่า Proximal Policy Optimization (PPO) ซึ่งจะกล่าวในหัวข้อ 2.5 ยูนิตี ML-Agents ประกอบด้วย 3 องค์ประกอบหลักดังนี้

1. องค์ประกอบการเรียนรู้ (Learning Component) ซึ่งประกอบด้วยฉากและสิ่งแวดล้อม
2. Python API ที่มีอัลกอริธึมการเรียนรู้แบบเสริมกำลังตัวอย่างเช่น PPO และ SAC ซึ่งเป็นสิ่งที่ใช้พื่อฝึกอบรมและทดสอบ
3. เครื่องมือสื่อสารภายนอก (External Communicator) เพื่อให้เอเจนต์สื่อสารระหว่างองค์ประกอบการเรียนรู้กับผู้สอนผ่าน Python API



รูปที่ 2-5 แผนภาพแสดงองค์ประกอบหลักของยูนิตี ML-Agents [22]



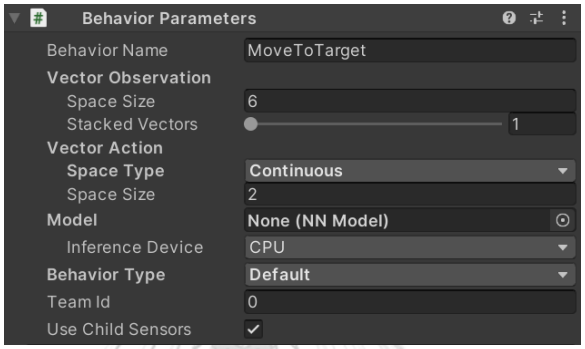
รูปที่ 2-6 แผนภาพส่วนประกอบภายในขององค์ประกอบการเรียนรู้ [22]

จากรูปที่ 6 ส่วนประกอบภายในขององค์ประกอบการเรียนรู้ (Learning Component) มีดังนี้

1. เอเจนต์ (Agent) เป็นตัวละครในฉาก มีหน้าที่นำข้อมูลจากสิ่งที่ทำการสังเกตมาร่วมกับการตัดสินใจของตัวสมอง (Brain) จนเกิดเป็นการกระทำ (Action) ขึ้นมา
2. สมอง (Brain) เป็นสิ่งที่ทำการตัดสินใจที่จะบอกเอเจนต์ว่าต้องดำเนินการอย่างไรในแต่ละสถานะ หลังจากนั้นจะส่งการตัดสินใจนั้นไปที่ตัวเอเจนต์

3. Academy เป็นสิ่งที่ทำให้ทุกๆอย่างทำงานร่วมกันได้ โดยมีตัวเครื่องมือสื่อสารภายนอก (External Communicator) เชื่อมต่อกับ Python API เพื่อเรียกใช้งาน Tensorflow

เอเจนต์ถูกสร้างขึ้นโดยถูกวางอยู่ในสภาพแวดล้อมที่ต้องการฝึกอบรมจำเป็นต้องสองคอมโพเนนต์ที่เป็นสคริปต์ของเอเจนต์ที่มีชื่อว่า “Behavior Parameter” และ “Decision Requester” ซึ่งสคริปต์ของเอเจนต์อยู่ในอุปกรณ์เสริมยูนิตี ML-agents และอนุญาตให้นักพัฒนาขยายได้โดยการสร้างสคริปต์ใหม่ที่อ้างอิงมาจากสคริปต์ของเอเจนต์เพื่อกำหนดรางวัลและการสังเกตให้กับเอเจนต์ โดยสคริปต์ของเอเจนต์เปรียบเสมือนสมองของเอเจนต์



รูปที่ 2-7 behavior parameters คอมโพเนนต์

โดยที่ - Behavior Name เป็นสิ่งที่กำหนดชุดของไฮเปอร์พารามิเตอร์ที่สมองนี้ใช้

- Vector Observation เป็นขนาดของอาร์เรย์ที่เก็บการสังเกตของสภาพแวดล้อม

- Vector Action เป็นอาร์เรย์ที่มีเก็บค่าของการกระทำสำหรับเอเจนต์ซึ่งมีค่าเป็นทศนิยม (continuous) หรือ จำนวนเต็ม (discrete)

- Model เป็นโมเดลที่ถูกฝึกแล้ว ซึ่งจะถูกใช้เมื่อเอเจนต์ไม่ได้ฝึก

- Behavior Type เป็นสามตัวเลือกที่เป็นไปได้ ได้แก่ ค่าเริ่มต้น (default) ฮิวริสติก (heuristic) และการอนุมาน (interference)

ส่วน Decision Requester คอมโพเนนต์ถูกใช้เพื่อควบคุมอัตราของการกระทำใหม่ที่ถูกร้องขอจากโมเดลการเรียนรู้แบบเสริมกำลังแล้วส่งการกระทำไปยังเอเจนต์ เอเจนต์จะรวบรวมข้อสังเกตจากฉากเกม แล้วลงมือทำ และหลังจากนั้นก็รับรางวัล

2.4 การออกแบบแบบจำลองการอพยพฝูงชนบนระนาบสองมิติ

ในหัวข้อย่อยที่ผ่านมากล่าวถึงการทดสอบและประเมินสมรรถนะของการเคลื่อนย้ายผู้โดยสารผ่านการนำเสนอในโลกสามมิติโดยใช้โปรแกรมยูนิตี ซึ่งมีข้อดีคือสามารถมองเห็นเหตุการณ์

ต่าง ๆ ที่เกิดขึ้นได้ใกล้เคียงสภาพความเป็นจริง อย่างไรก็ดี การทำงานในโลกสามมิติต้องใช้ทรัพยากรการคำนวณและประมวลผลสูง ดังนั้น ในวิทยานิพนธ์ฉบับนี้จะพัฒนาโปรแกรมการจำลองเพื่อทดสอบการเคลื่อนย้ายผู้โดยสารบนระนาบสองมิติด้วย ซึ่งในหลักการแล้วจะสามารถจำลองระบบหรือสถานการณ์ที่มีความซับซ้อนหรือมีขนาดใหญ่ได้ดีกว่าการจำลองแบบสามมิติ ทั้งนี้การพัฒนาระบบจำลองสองมิตินี้จะใช้โปรแกรมไพทอนที่พัฒนาขึ้นเองเกือบทั้งหมดโดยใช้หลักการแรงทางสังคมเพื่อใช้จำลองการเคลื่อนที่ของผู้โดยสาร

แบบจำลองการอพยพฝูงชนบนระนาบสองมิติพัฒนาโดยใช้โปรแกรมไพทอนเพื่อศึกษาแบบจำลองแรงทางสังคม โดยแบบจำลองการอพยพฝูงชนบนระนาบสองมิติมีขั้นตอนการทำงานหรือรหัสเทียม (Pseudo code) ดังนี้

1. กำหนดตำแหน่งเริ่มต้นและตำแหน่งเป้าหมายของเอเจนต์ ณ sim_time เท่ากับ 0
2. กำหนด delta_time เท่ากับ 0.05
3. คำนวณแรงที่ต้องการตามสมการที่ 2-2
4. คำนวณแรงปฏิสัมพันธ์ระหว่างคนเดินเท้าด้วยกันและแรงปฏิสัมพันธ์ระหว่างคนเดินเท้ากับกำแพงตามสมการที่ 2-3
5. รวมแรงทั้งหมดที่เกิดขึ้น
6. อับเดตความเร็ว ณ sim_time นั้น ๆ ตามสมการที่ 2-1
7. อับเดตตำแหน่งของเอเจนต์ ณ sim_time นั้น ๆ
8. อับเดต sim_time += delta_time
9. กลับไปที่ขั้นตอนที่ 3

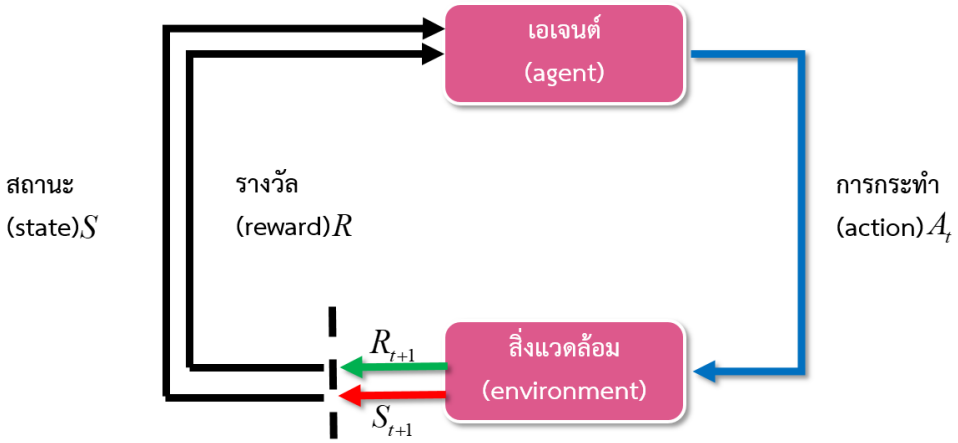
2.5 การเรียนรู้แบบเสริมกำลัง (Reinforcement Learning)

หัวข้อย่อยนี้จะอธิบายพื้นฐาน หลักการทำงาน และองค์ประกอบหลักของวิธีการเรียนรู้แบบเสริมกำลัง เนื่องจากจานวิจัยนี้ได้นำวิธีการเรียนรู้แบบเสริมกำลังมาประยุกต์ใช้กับการจำลองการอพยพฝูงชนพร้อมนำในสถานีรถไฟฟ้า

2.5.1 พื้นฐานของการเรียนรู้แบบเสริมกำลัง

การเรียนรู้แบบเสริมกำลัง [22, 23, 30] เป็นการทำให้ระบบคอมพิวเตอร์เรียนรู้ได้ด้วยตนเองโดยใช้ข้อมูล (Machine Learning Algorithm) แบบหนึ่งส่งผลให้คอมพิวเตอร์สามารถคิดได้

ด้วยตนเองนั่นเอง การเรียนรู้แบบเสริมกำลังมีหลักการทำงานเหมือนกับการที่มนุษย์เรียนรู้บางสิ่งบางอย่างด้วยการลองผิดลองถูก และมีการเรียนรู้เกิดขึ้นระหว่างทางว่าการกระทำไหนดีหรือไม่ดี โดยมีวัตถุประสงค์คือการลงมือทำการกระทำตามนโยบายที่ให้ผลตอบแทนระยะยาวที่คาดหวังสูงสุด



รูปที่ 2-8 แผนภาพแสดงการทำงานโดยรวมของการเรียนรู้แบบเสริมกำลัง

นิยามของส่วนประกอบต่าง ๆ ในแผนภาพการทำงานโดยรวมของการเรียนรู้แบบเสริมกำลัง

- เอเจนต์ (agent) คือ ผู้เรียนรู้และผู้ลงมือทำการกระทำที่มีผลกระทบต่อสิ่งแวดล้อมภายใต้สภาวะที่แตกต่างกัน
- สิ่งแวดล้อม (environment) คือ ระบบที่เอเจนต์มีปฏิสัมพันธ์ด้วย
- การกระทำ (action : A_t) คือ การกระทำของเอเจนต์ ณ เวลา t ซึ่งส่งผลบางอย่างต่อสิ่งแวดล้อม
- รางวัล (reward : R_t) คือ รางวัลที่ได้จากการตัดสินใจลงมือทำการกระทำบางอย่าง ณ เวลา t
- สถานะ (state : S_t) คือ สถานะของระบบที่ทางเอเจนต์สามารถรับรู้ได้ ณ เวลา t
- รางวัล (reward : R_{t+1}) คือ รางวัลที่ได้จากการตัดสินใจลงมือทำการกระทำ ณ เวลา $t+1$
- สถานะ (state : S_{t+1}) คือ สถานะของระบบที่ทางเอเจนต์สามารถรับรู้ได้ ณ เวลา $t+1$

- 2.5.2 หลักการการทำงานการเรียนรู้แบบเสริมกำลัง

- จากรูปที่ 8 สามารถอธิบายหลักการการทำงานของการเรียนรู้แบบเสริมกำลังเป็นลักษณะที่ฝึกให้เอเจนต์ (agent) เรียนรู้นโยบายหรือพฤติกรรม (policy) ที่เหมาะสมในสภาพแวดล้อม

(environment) ที่ตัวเองอยู่ โดยนโยบายหรือพฤติกรรมจะเป็นการบอกเอเจนต์ว่าจะต้องเลือกกระทำการกระทำใด (action) ที่ได้ทำการสังเกต (observation) ในแต่ละสถานะ (state) ถ้าการกระทำนั้นดี ก็จะได้รางวัล (reward) สูง แต่ถ้าการกระทำนั่นแย่ ก็จะได้รางวัลต่ำหรืออาจเป็นบทลงโทษ โดยเป้าหมายของเอเจนต์คือสะสมรางวัลให้ได้สูงที่สุด รวมทั้งเรียนรู้ผ่านข้อผิดพลาดในอดีตที่เกิดขึ้น รางวัลสะสม (cumulative reward) คือผลรวมของรางวัลในแต่ละสถานะซึ่งเป็นการแก้ปัญหาแบบไม่ต่อเนื่อง (discrete task) สามารถเขียนได้ดังนี้

$$R(\tau) = R_{t+1} + R_{t+2} + R_{t+3} + ...R_T = \sum_{k=0}^{\infty} r_{t+k+1} \quad (2\text{-}10)$$

- แต่เนื่องด้วยการแก้ปัญหาแบบต่อเนื่อง (continuous task) อาจทำรางวัลสะสมมีค่าเป็นอนันต์ ดังนั้นรางวัลสะสมจะถูกสิ่งที่เรียกว่า ส่วนลด (Discount Factor (γ)) ที่มีค่าระหว่าง 0 ถึง 1 เป็นตัวถ่วงน้ำหนัก เพื่อให้ผลรวมของรางวัลสะสมเป็นจำนวนที่มีค่าจำกัดและให้ความสำคัญกับรางวัลที่เกิดขึ้นในเวลาปัจจุบันมากกว่า ซึ่งสามารถเขียนได้ดังนี้

$$R(\tau) = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad (2\text{-}11)$$

- อัลกอริธึมการเรียนรู้แบบเสริมแรงที่ใช้ในตัวอย่างนี้มีชื่อเรียกว่า อัลกอริธึมการเรียนรู้คิว (Q-learning algorithm) ซึ่งเป็นวิธีพื้นฐานและง่ายต่อการเรียนรู้ วิถีของการทำงานตามอัลกอริธึมคิวจะใช้ตารางคิว (Q table) เป็นเครื่องมือในการสะสมความรู้อันได้จากการลองผิดลองถูกเพื่อการเรียนรู้ เมื่อเอเจนต์ได้ผ่านกระบวนการเรียนรู้มากเพียงพอแล้ว ความรู้ที่สะสมในตารางคิวจะถูกนำมาใช้ในการตัดสินใจเลือกการกระทำที่เหมาะสมต่อไป โดยในช่วงเริ่มต้นตัวเลขทั้งหมดในตารางคิวถูกกำหนดให้มีค่าเป็นศูนย์ ในแต่ละครั้งที่เอเจนต์ลองผิดลองถูกจะมีการอัปเดตค่าตัวเลขคิวโดยใช้สูตรต่อไปนี้

$$Q^{new}(s_t, a_t) = Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \quad (2\text{-}12)$$

- โดยที่ $Q^{new}(s_t, a_t)$ คือ ค่าอัปเดตใหม่ของฟังก์ชันคิวเมื่อเอเจนต์อยู่ที่สถานะ s_t ณ เวลา t และได้เลือกการกระทำ a_t
- $Q(s_t, a_t)$ คือ ค่าปัจจุบันของฟังก์ชันคิวเมื่อเอเจนต์อยู่ที่สถานะ s_t ณ เวลา t และได้เลือกการกระทำ a_t
- α คือ ค่าอัตราการเรียนรู้ (learning rate) โดย $\alpha \in [0,1]$

- r_{t+1} คือ รางวัลที่ได้เมื่อเอเจนต์อยู่ที่สถานะ s_t ณ เวลา t และได้เลือกการกระทำ a_t
- γ คือ ค่าตัวประกอบลด (discount factor) $\gamma \in [0,1]$
- $\max_a Q(s_{t+1}, a)$ คือ ค่าของฟังก์ชันคิวเมื่อเอเจนต์อยู่ที่สถานะถัดไป s_{t+1} ณ เวลา $t+1$ โดยให้เลือกการกระทำ a ที่ให้ค่าฟังก์ชันคิวสูงสุด

2.5.3 องค์ประกอบหลักของการเรียนรู้แบบเสริมกำลัง

1) นโยบายหรือพฤติกรรม (Policy) คือการจับคู่ (Map) สถานะกับการกระทำที่เป็นไปได้ในสถานะนั้นเพื่อหานโยบายที่เหมาะสมหรือกล่าวอีกนัยนึงว่าเป็นหลักการที่เอเจนต์ใช้ในการตัดสินใจเลือกการกระทำหลังจากประเมินสถานะแล้วเพื่อให้ได้ผลรวมรางวัลสะสมสูงสุด โดยนโยบายหรือพฤติกรรมสามารถจำแนก 2 ประเภทดังนี้
   - นโยบายเชิงกำหนด (Deterministic policy) คือ เอเจนต์ลงมือทำการกระทำ a เฉพาะเพียงอย่างเดียวที่เป็นไปได้ในสถานะ s เพื่อให้ได้ฟังก์ชันมูลค่าสูงที่สุด ซึ่งสามารถเขียนเป็นสมการดังนี้

$$\pi(s) = \arg\max_a Q(s,a) \qquad (2\text{-}13)$$

   - นโยบายเชิงสุ่ม (Stochastic policy) คือ เอเจนต์มีความน่าจะเป็นที่เลือกการหนึ่งการกระทำ a จากหลายการกระทำที่เป็นไปได้ในสถานะ s ซึ่งสามารถเขียนเป็นสมการดังนี้

$$\pi(a \mid s) = \Pr\{A_t = a \mid S_t = s\} \qquad (2\text{-}14)$$

2) ฟังก์ชันการให้รางวัล (Reward function) คือ ฟังก์ชันที่คำนวณรางวัลสะสมและเป็นสิ่งที่ใช้ประเมินผลลัพธ์ที่เกิดจากการกระทำของเอเจนต์ว่าดีหรือแย่ ดังนั้นรางวัลจึงเป็นสิ่งที่ทำให้นโยบายหรือพฤติกรรมเปลี่ยน เมื่อนโยบายหรือพฤติกรรมที่กำหนดการกระทำที่ได้รางวัลที่มีค่าน้อย นโยบายหรือพฤติกรรมก็จะถูกเปลี่ยนให้ไปเลือกการกระทำอื่นในครั้งต่อ ๆ ไป ซึ่งฟังก์ชันการให้รางวัลเป็นดังสมการที่ *(2-11)* ดังนั้นการออกแบบฟังก์ชันการให้รางวัลมีความสำคัญอย่างยิ่งในการเรียนรู้การเสริมกำลัง
3) ฟังก์ชันมูลค่า (Value function) คือ ฟังก์ชันที่ถูกกำหนดเป็นผลตอบแทนที่คาดหวังหรือรางวัลสะสมของเอเจนต์ในบางสถานะ ฟังก์ชันมูลค่าในการเรียนรู้แบบเสริมกำลังสามารถจำแนก 2 ประเภทดังนี้
   - ฟังก์ชันมูลค่าสถานะ (State value function) จะบอกว่าสถานะที่กำหนดนั้นดีเพียงใดสำหรับเอเจนต์ที่กระทำตามนโยบาย โดยมูลค่าของสถานะ s ภายใต้

นโยบายหรือพฤติกรรม π เท่ากับผลตอบแทนที่คาดหวังหรือรางวัลสะสมจากสถานะ s ตามนโยบายหรือพฤติกรรม π ซึ่งสามารถเขียนเป็นสมการดังนี้

$$V_\pi(s) = E_\pi\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} \mid S_t = s\} \quad (2\text{-}15)$$

- ฟังก์ชันมูลค่าการกระทำ (Action value function) จะบอกว่าการกระทำที่เอเจนต์ทำนั้นดีเพียงใดจากสถานะที่กำหนดขณะที่ทำตามนโยบาย โดยมูลค่าของการกระทำ a ในสถานะ s ภายใต้นโยบายหรือพฤติกรรม π เท่ากับผลตอบแทนที่คาดหวังหรือรางวัลสะสมจากสถานะ s ตามนโยบายหรือพฤติกรรม π และลงมือทำการกระทำ a ซึ่งสามารถเขียนเป็นสมการดังนี้

$$Q_\pi(s \mid a) = E_\pi\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} \mid S_t = s, A_t = a\} \quad (2\text{-}16)$$

2.6 การเพิ่มประสิทธิภาพนโยบายใกล้เคียง (Proximal Policy Optimization; PPO)

ในหัวข้อย่อยที่ผ่านมาเป็นการพยายามเรียนรู้ค่าของการกระทำต่าง ๆ แล้วเลือกการกระทำตามผลตอบแทนที่คาดหวังหรือรางวัลสะสมและมีการเปลี่ยนนโยบายหรือพฤติกรรมหากได้ผลตอบแทนที่คาดหวังหรือรางวัลสะสมน้อยเพื่อหานโยบายหรือพฤติกรรมที่ดีขึ้นกว่าเดิม หัวข้อนี้จะกล่าวถึงวิธีการไล่ระดับนโยบาย (policy gradient method) ซึ่งเป็นการปรับที่นโยบายหรือพฤติกรรมโดยตรง หลังจากนั้นจะกล่าวถึงเทคนิคการเพิ่มประสิทธิภาพนโยบายใกล้เคียงที่ถูกใช้สำหรับอบรมเอเจนต์ในยูนิตี ML-Agents

2.6.1 วิธีการไล่ระดับนโยบาย

วิธีการไล่ระดับนโยบายเป็นอีกทางเลือกหนึ่งสำหรับการเรียนรู้การเสริมกำลังแบบเชิงลึก (Deep reinforcement learning) คือการลืมค่าคิว (Q-value) และให้โครงข่ายประสาทเรียนรู้และประมาณนโยบายหรือพฤติกรรมที่เหมาะสมโดยตรงแทนด้วยการกำหนดพารามิเตอร์บางอย่าง ซึ่งก็คือ เวกเตอร์น้ำหนักนโยบายหรือพฤติกรรม (policy weight vector: θ) เพื่อให้ได้ฟังก์ชันมูลค่าสูง โดยนโยบายหรือพฤติกรรมคือความน่าจะเป็นที่เอเจนต์เลือกการหนึ่งการกระทำ a จากหลายการกระทำที่เป็นไปได้ในสถานะ s และนโยบายหรือพฤติกรรม θ ซึ่งสามารถเขียนเป็นสมการดังนี้

$$\pi(a \mid s, \theta) = \Pr\{A_t = a \mid S_t = s, \theta_t = \theta\} \quad (2\text{-}17)$$

2.6.2 การเพิ่มประสิทธิภาพนโยบายใกล้เคียง

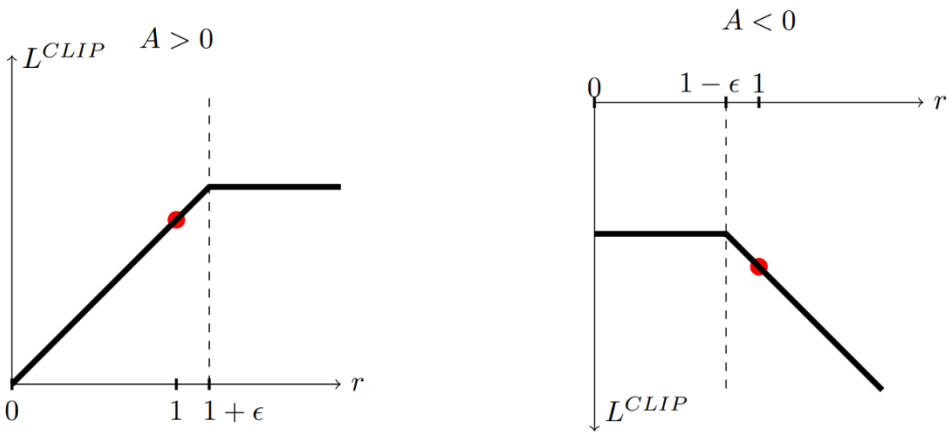
การเพิ่มประสิทธิภาพนโยบายใกล้เคียง [21] เป็นเป็นอัลกอริธึมหลักที่ถูกใช้ในอุปกรณ์เสริมยูนิตี ML-agent toolkit และเป็นการเรียนรู้แบบเสริมกำลังรูปแบบหนึ่งที่ใช้วิธีการไล่ระดับนโยบายแบบนโยบายเชิงสุ่ม ซึ่งเป็นการกระจายความน่าจะเป็นของการกระทำแทนการเลือกการกระทำ (นั้นตรงๆ เพื่อไปเปลี่ยนพฤติกรรมหรือนโยบายให้เพิ่มโอกาสที่จะทำการกระทำที่ได้รางวัลสูง และลดโอกาสที่จะทำการกระทำที่ได้รางวัลต่ำ

การเพิ่มประสิทธิภาพนโยบายใกล้เคียงเป็นอัลกอริธึมที่ใช้งานง่ายและค่อนข้างเสถียรเนื่องจากได้รับการปกป้องจากการอัปเดตนโยบายขนาดใหญ่ที่ทำลายล้าง (destructive large policy) จากการไล่ระดับ (gradient ascent) โดยจำกัดการเปลี่ยนแปลงสูงสุดของการอัปเดต นโยบายเป็นช่วงที่เล็กกว่า $[1-\varepsilon,1+\varepsilon]$ เพื่อควบคุมการเปลี่ยนแปลงพฤติกรรมหรือนโยบายในการทำซ้ำแต่ละครั้ง ซึ่งฟังก์ชันวัตถุประสงค์หลักสามารถเขียนได้ดังนี้

$$L^{CLIP}(\theta)=\hat{E}_t[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta),1-\varepsilon,1+\varepsilon)\hat{A}_t)] \qquad (2\text{-}18)$$

โดยที่ - θ คือพารามิเตอร์พฤติกรรมหรือนโยบาย

- $\hat{E}_t$ คือความคาดหวังเชิงประจักษ์ในช่วงเวลา
- r_t คืออัตราส่วนความน่าจะเป็นตามพฤติกรรมหรือนโยบายใหม่และเก่าตามลำดับ
- $\hat{A}_t$ คือความได้เปรียบโดยประมาณ ณ เวลานั้น ๆ
- ε คือไฮเปอร์พารามิเตอร์ที่มีค่าเท่ากับ 0.1 หรือ 0.2



รูปที่ 2-9 L^{CLIP} ถูกพล็อตเป็นฟังก์ชันของอัตราส่วนความน่าจะเป็น [21]

2.7 มาตรฐานการออกแบบเพื่อป้องกันอัคคีภัย

หัวข้อย่อยนี้จะกล่าวถึงมาตรฐานการออกแบบเพื่อป้องกันอัคคีภัย ซึ่งคือมาตรฐานความปลอดภัย NFPA 130 (National Fire Protection Association) ซึ่งเป็นมาตรฐานสากลที่ทันสมัยที่สุดในการออกแบบระบบขนส่งมวลชนประเภทราง เพื่อความปลอดภัยสูงสุดในการบริการแก่ประชาชน โดยมาตรฐานดังกล่าว โครงการรถไฟฟ้าในประเทศไทยได้นำมาใช้เป็นมาตราการความปลอดภัยภายในสถานีรถไฟฟ้าทั้งบนดินและใต้ดิน

มาตรฐานความปลอดภัย NFPA 130 [12] เป็นหนึ่งในมาตรฐานที่เข้มงวดมากของมาตรฐานการออกแบบรถไฟของอเมริกาสำหรับระบบการขนส่งทางไกลแบบถาวร (Fixed Guideway Transit Systems) และระบบรถไฟโดยสาร ซึ่งเป็นมาตรฐานที่นิยมใช้อย่างแพรหลายในประเทศที่ยังไม่มีมาตรฐานหรือข้อกำหนดการออกแบบเป็นของตนเอง เพื่อความปลอดภัยสูงสุดในการให้บริการแก่ประชาชน ความปลอดภัยของผู้โดยสารในสถานีจะประเมินจากระยะเวลาที่ใช้ในการอพยพ โดย NFPA 130 กำหนดไว้ดังนี้

- เวลาที่ใช้ในการอพยพจากชานชาลา ควรใช้เวลาในการอพยพไม่เกิน 4 นาที และควรมีทางออกที่เพียงพอสำหรับเคลื่อนย้ายคนออกจากสถานี ทั้งนี้การอพยพจะขึ้นอยู่กับการวิเคราะห์ทางวิศวกรรมโดยการประเมินสถานการณ์ที่เกิดขึ้น
- ระยะเวลาการอพยพจากจุดเกิดเหตุไปยังจุดปลอดภัย ควรใช้เวลาในการเคลื่อนย้ายไม่เกิน 6 นาที และควรคำนึงถึงพื้นที่ที่ปลอดภัยและการอพยพให้ได้มากที่สุด

บทที่ 3
วิธีการอพยพฝูงชน

บทนี้จะอธิบายถึงวรรณกรรมที่เกี่ยวข้องที่ผู้วิจัยได้ทำการทบทวนได้แก่ การประยุกต์ใช้แบบจำลองแรงทางสังคมกับการอพยพฝูงชน และการประยุกต์ใช้การเรียนรู้เสริมกำลังกับการอพยพฝูงชน ทั้งหมดนี้เป็นการศึกษาเพื่อให้เกิดความรู้ความเข้าใจและนำมาประยุกต์ใช้งาน

3.1 การประยุกต์ใช้แบบจำลองแรงทางสังคมกับการอพยพฝูงชน

หัวข้อย่อยนี้จะอธิบายวรรณกรรมในช่วงสองทศวรรษที่ผ่านมา ที่ใช้แบบจำลองแรงทางสังคมมาประยุกต์ใช้กับการอพยพฝูงชนจากพื้นที่ขนาดใหญ่ดังนี้

งานวิจัยหมายเลข [14] ประยุกต์ใช้แบบจำลองแรงทางสังคมครั้งแรกกับการอพยพด้วยความตื่นตระหนก ซึ่งถือว่าฝูงชนอยู่ในสถานการณ์ฉุกเฉิน เนื่องจากความกลัวหรือเหตุผลกดดันอื่น ๆ งานวิจัยนี้ต้องการศึกษาความเร็วที่ต้องการ ศึกษาเส้นทางหลบหนีที่มีพื้นที่กว้างขึ้นแล้วแคบลง ซึ่งเป็นคอขวด และพฤติกรรมการค้นหาเฉพาะตัวเองหรือการทำตามคนส่วนใหญ่ว่าส่งผลอย่างไรกับการอพยพ

โดยการศึกษาความเร็วที่ต้องการนั้นได้จำลองการอพยพคนจำนวน 200 คนออกจากห้อง พบว่า เมื่อความเร็วที่ต้องการมากกว่า 1.5 เมตรต่อวินาที เวลาในการอพยพคนออกจากห้องเริ่มเพิ่มขึ้น และ อัตราไหลของฝูงชนก็จะเริ่มลดลง เมื่อความเร็วที่ต้องการ มากกว่า 5 เมตรต่อวินาที จะเริ่มมีการชนกันของฝูงชนส่งผลให้มีคนบาดเจ็บซึ่งกลายเป็นสิ่งกีดขวางที่ไม่เคลื่อนที่ ดังนั้นเวลาการอพยพมากขึ้นเรื่อย ๆ การศึกษาเส้นทางหลบหนีที่มีพื้นที่กว้างขึ้นแล้วแคบลง หรือทดสอบปรากฏการณ์ข้อขวด พบว่ายิ่งมุมของคอขวดมีค่ามากขึ้นประสิทธิภาพการอพยพก็จะลดลง สุดท้ายการศึกษาพฤติกรรมการค้นหาเฉพาะตัวเองหรือการทำตามคนส่วนใหญ่ จำลองคนจำนวน 90 คนอพยพออกจากห้องที่เต็มไปด้วยควัน ซึ่งห้องนั้นมีประตูอยู่ 2 ฝั่งตรงข้ามกัน

พบว่าการอพยพตามพฤติกรรมการค้นหาเฉพาะตัวเองสามารถอพยพได้แต่ไม่มีประสิทธิภาพเท่าที่ควร ส่วนการอพยพตามคนส่วนใหญ่จะทำให้ประสิทธิภาพการอพยพแย่ เนื่องจากการอัดของคนจำนวนมากที่ประตูทางออกฝั่งใดฝั่งหนึ่ง ดังนั้นควรผสมผสานระหว่างการค้นหาเฉพาะตัวเองและการทำตามคนส่วนใหญ่ จะทำให้การอพยพฝูงชนมีประสิทธิภาพมากที่สุด

งานวิจัยหมายเลข [17] เสนอวิธีการปรับเปลี่ยนการเคลื่อนที่คนเดินเท้าตามแบบจำลองแรงทางสังคมเพื่อให้สมจริงยิ่งขึ้น แบบจำลองที่ได้รับการดัดแปลงนั้นใช้สำหรับการสร้างแบบจำลองกลุ่มคนเดินถนนแบบมีผู้นำ ซึ่งกำหนดแรงแรงนำทางไว้

การศึกษานี้เป็นการจำลองการอพยพออกจากสถานีรถไฟปักกิ่งส่วนใต้ ซึ่งแบ่งการอพยพออกเป็น 2 กรณี คือการอพยพโดยปราศจากผู้นำ แลการอพยพพร้อมผู้นำ เวลาในการอพยพมี 3 กรณี คือ 470 590 และ 720 วินาที ซึ่งมีจำนวนคนที่สามารถอพยพได้จากการอพยพกรณีแรกเท่ากับ 451 558 และ 669 คนตามลำดับ ส่วนจำนวนคนที่สามารถอพยพได้จากการอพยพกรณีสองเท่ากับ 676 826 และ 983 คนตามลำดับ จะเห็นได้ว่าผู้นำอพยพมีบทบาทสำคัญในการอพยพคนเดินเท้าอย่างมีประสิทธิภาพ

งานวิจัยหมายเลข [15] เสนอวิธีการหาจำนวนและตำแหน่งของผู้นำที่เหมาะสมสำหรับการอพยพฝูงชนด้วยผู้นำภายในสถานีขนส่งรถไฟในเมือง มีวัตถุประสงค์ของปัญหาคือการลดต้นทุนการอพยพทั้งหมดดังนี้ ต้นทุนค่าแรงของผู้นำอพยพ ต้นทุนของเวลาการอพยพทั้งหมด และต้นทุนค่าความปลอดภัยผู้โดยสารที่ยังไม่ได้อพยพออกจากสถานีรถไฟภายในเวลาที่กำหนด นอกจากนี้ยังเสนอกลยุทธ์เพื่อพัฒนาการอพยพด้วยผู้นำ โดยใช้ขั้นตอนวิธีเชิงพันธุกรรม (Genetic Algorithm) สำหรับการหาทางออกที่เหมาะสมควบคู่กับการค้นหาแบบเอสตาร์ (A* algorithm) สำหรับการหาเส้นทางการอพยพไปยังทางออกที่ได้ และมีการใช้กลไกความร่วมมือระหว่างผู้นำเพื่อป้องกันการเลือกเส้นทางที่ซ้ำกัน เนื่องจากการเลือกเส้นทางที่ซ้ำกันของผู้นำส่งผลให้เกิดการชะลอตัวในการอพยพการศึกษานี้

งานวิจัยนี้จำลองสถานการณ์การอพยพ 4 สถานการณ์ดังนี้ การอพยพโดยปราศจากผู้นำ การอพยพกับผู้นำที่สุ่มตำแหน่ง การอพยพพร้อมจำนวนและตำแหน่งผู้นำที่เหมาะสม และการอพยพพร้อมจำนวนและตำแหน่งผู้นำที่เหมาะสมโดยใช้กลไกความร่วมมือระหว่างผู้นำ

ผลการจำลองพบว่า การใช้ผู้นำอพยพสามารถเพิ่มประสิทธิภาพการอพยพได้โดยไม่ต้องคำนึงถึงตำแหน่งผู้นำ การระบุตำแหน่งผู้นำอย่างเหมาะสมสามารถเพิ่มประสิทธิภาพการอพยพและลดเวลาการอพยพได้ สุดท้ายการใช้กลไกความร่วมมือระหว่างผู้นำสามารถลดเวลาการอพยพได้อีกซึ่งคิดเป็นเวลา 10 วินาที

งานวิจัยหมายเลข [18] เสนอแบบจำลองแรงทางสังคมที่ปรับเปลี่ยนศึกษาผลกระทบของจำนวนและการกระจายสัญญาณฉุกเฉินต่อประสิทธิภาพการอพยพคนเดินเท้าในสถานีรถไฟใต้ดินปักกิ่ง นอกจากนี้แบบจำลองความน่าจะเป็นในการรับรู้ก็ถูกกำหนดขึ้นสำหรับคำอธิบายเชิงปริมาณ

ของความน่าจะเป็นที่คนเดินเท้าสามารถสังเกตเห็นป้ายได้สำเร็จและรับรู้ข้อมูลคำแนะนำได้อย่างชัดเจน

ผลการวิจัยพบว่าจำนวนและตำแหน่งของป้ายมีบทบาทสำคัญในการอพยพฉุกเฉินไม่ว่าป้ายสัญญาณจะมีการกระจายแบบใดก็ตาม การติดตั้งป้ายฉุกเฉินสามารถปรับปรุงประสิทธิภาพการอพยพได้ ด้วยการเลือกสถานที่ที่เหมาะสม เช่น รูปแบบการครอบคลุมสูงสุดสามารถลดเวลาการอพยพลงเหลือน้อยกว่า 50%

งานวิจัยหมายเลข [16] ใช้แบบจำลองของแรงทางสังคมเพื่อวิเคราะห์อิทธิพลของความกว้างความยาวของทางออก ความเร็วที่ต้องการ และเวลาในการอพยพ

จากการศึกษาพบว่าการเพิ่มความเร็วที่ต้องการส่งผลให้เวลาการอพยพลดลง เมื่อความกว้างของทางออกมีขนาดเล็กอิทธิพลของความยาวทางออกจะส่งผลต่อเวลาอพยพชัดเจน เมื่อความกว้างของทางออกมีขนาดใหญ่อิทธิพลของความยาวทางออกที่ส่งผลต่อเวลาอพยพจะถูกละเลย

3.2 การประยุกต์ใช้การเรียนรู้เสริมกำลังกับการอพยพฝูงชน

งานวิจัยหมายเลข [20] เสนอกรอบงานการตรวจจับความแออัดที่ใช้ Multi-Agent Reinforcement Learning (MARL) เพื่อเตรียมตัวสำหรับกระบวนการการวางแผนหาเส้นทางอพยพ กรอบงานนี้ถูกแบ่งออกเป็นสามส่วน ส่วนแรก กำหนดการแบ่งฝูงชนออกเป็นกลุ่มสำหรับแต่ละห้อง หลังจากได้กลุ่มของฝูงชนแล้ว จะเลือกผู้นำอพยพของกลุ่มจากจุดศูนย์กลางมวลของกลุ่มเป็นผู้นำ ส่วนสอง จำลองการทำแผนที่ผู้นำ ส่วนสาม กำหนดองค์ประกอบการเรียนรู้เสริมกำลังตามการตรวจจับความแออัด เช่น จำลองสิ่งแวดล้อมที่เอเจนต์ทำการเรียนรู้ กำหนดสิ่งที่เอเจนต์ทำการสังเกต กำหนดการกระทำของเอเจนต์ และออกแบบฟังก์ชนการให้รางวัลซึ่งมีความสำคัญอย่างยิ่งในการเสริมการเรียนรู้ เสนอกระบวนการการวางแผนหาเส้นทางอพยพโดยแบ่งเป็น 2 ส่วน ส่วนแรก ใช้ Improved Multi-Agent Deep Deterministic Policy Gradient (IMADDPG) algorithm เพื่อดำเนินการวางแผนเส้นทางทั้งหมดสำหรับเอเจนต์ผู้นำทั้งหมดและช่วยให้ประสิทธิภาพของงานการวางแผนร่วมกันของเอเจนต์ทั้งหมดมีประสิทธิภาพเพิ่มมากขึ้น ส่วนสอง ใช้ Reciprocal Velocity Obstacles (RVO) เพื่อหลีกเลี่ยงการชนกันของผู้นำอพยพและผู้ติดตาม รวมถึงสิ่งกีดขวาง

จากการศึกษาพบว่า ในสภาพแวดล้อมที่ซับซ้อนซึ่งมีอุปสรรคและทางออกหลายทางการตรวจจับความแออัดสามารถลดความแออัดของฝูงชนในกรณีฉุกเฉินได้ในระดับหนึ่ง ซึ่งจะช่วยป้องกัน

และลดจำนวนผู้เสียชีวิตด้วยวิธีที่ปลอดภัยและรวดเร็วยิ่งขึ้น นอกจากนี้ยังเปรียบเทียบกับ DRL และอัลกอริธึม Swarm Intelligence เพื่อยืนยันประสิทธิภาพวิธีที่เสนอ

งานวิจัยหมายเลข [19] เสนอแนวทางการวางแผนเส้นทางอพยพโดยใช้ IMARL สำหรับการจำลองการอพยพฝูงชน โดยตำแหน่งเริ่มต้นของฝูงชนจะอ้างอิงจากวิดีโอเพื่อใช้เป็นปริภูมิสถานะสำหรับการเรียนรู้แบบเสริมกำลัง จำนวนฝูงชนจะถูกแบ่งเป็นกลุ่มตามระยะห่างและค่าความสัมพันธ์ของ Improved k-Medoids แล้วเลือกผู้นำจากคนที่อยู่ใกล้ทางออกมากที่สุดและคุ้นเคยกับสภาพแวดล้อมมากที่สุดเป็นผู้นำอพยพ จากนั้นผู้นำจะใช้อัลกอริธึม IMARL เลือกเส้นทางอพยพที่ดีสุด แล้วแบบจำลองแรงทางสังคมที่ได้รับการปรับปรุงแล้วเพื่อจำลองการเคลื่อนที่ของฝูงชนสำหรับการจำลองการอพยพฝูงชน โดยแบบจำลองแรงทางสังคมแบบดั้งเดิมจะเดินในลักษณะที่เป็นอิสระต่อกัน แต่แบบจำลองแรงทางสังคมที่ได้รับการปรับปรุงจะมีแนวโน้มเดินไปกับคนที่อยู่ใกล้หรือที่อยู่กลุ่มเดียวกัน

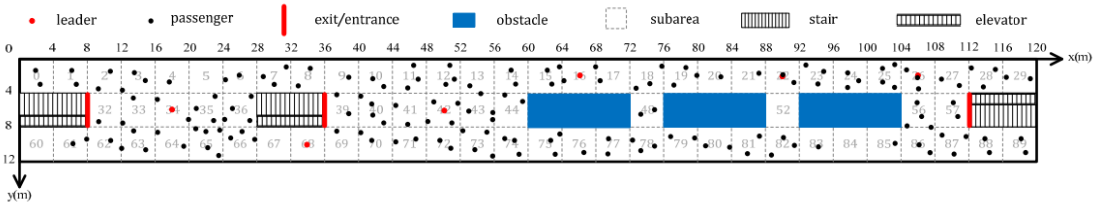
งานวิจัยนี้จำลองสถานการณ์การจำลองการอพยพฝูงชนมีดังนี้ การอพยพกรณีมี 2 ทางออก การอพยพกรณีมี 4 ทางออก โดยในแต่ละสถานการณ์มีการเปรียบเทียบกับการอพยพด้วยแบบจำลองแรงทางสังคมแบบดั้งเดิม การอพยพด้วยแบบจำลองแรงทางสังคมที่ขับเคลื่อนด้วยข้อมูลจากวิดีโอ และการอพยพด้วยวิธีการ IMARL ที่ใช้ร่วมกับแบบจำลองแรงทางสังคมที่ได้รับการปรับปรุง

จากการศึกษาพบว่า การมีทางออกเพิ่มมากขึ้นช่วยลดระยะเวลาการอพยพได้ นอกจากนี้ที่เวลาเดียวกัน การรวมตัวของวิธีการวางแผนเส้นทางโดย IMARL กับ แบบจำลองแรงทางสังคมที่ได้รับการปรับปรุงสามารถอพยพฝูงชนด้วยเวลาที่น้อยที่สุด

บทที่ 4
แบบจำลองการอพยพฝูงชนด้วยคอมพิวเตอร์

บทนี้กล่าวถึงการพัฒนาโปรแกรมจำลองการอพยพด้วยคอมพิวเตอร์โดยอาศัยแบบจำลองแรงทางสังคมในการจำลองพฤติกรรมการเดินเท้าของผู้อพยพภายในสถานีรถไฟฟ้าทดลอง และการค้นหาแบบเอสตาร์ในการค้นหาเส้นทางที่สั้นที่สุดในการอพยพของผู้อพยพภายในสถานีรถไฟฟ้าทดลอง เพื่อทดสอบประสิทธิภาพการอพยพ โดยจะถูกแบ่งออกเป็น 2 ส่วน ในส่วนแรกเป็นการพัฒนาและสร้างโปรแกรมแบบจำลองการอพยพฝูงชนบนระนาบสองมิติด้วยโปรแกรมไพทอนเองทั้งหมด ซึ่งเป็นประโยชน์กับผู้วิจัยในการเรียนรู้เกี่ยวกับการพัฒนาโปรแกรมและเข้าใจหลักการได้อย่างละเอียด ในภาคการแสดงผลจะใช้กราฟฟิคนำเสนอการเคลื่อนของฝูงชนในรูปแบบแอนิเมชันสองมิติเพื่อให้เห็นภาพเหตุการณ์ทุกขั้นตอนอย่างเป็นพลวัตทุกเวลา อย่างไรก็ดีโปรแกรมที่พัฒนาขึ้นนี้เหมาะกับการประยุกต์ใช้ในกรณีศึกษาที่มีจำนวนผู้อพยพไม่มากนัก ในส่วนที่สองเป็นการประยุกต์ใช้โปรแกรมยูนิตีซึ่งเป็นโปรแกรมในระดับเชิงพาณิชย์ที่มีขีดความสามารถในการจำลองผู้อพยพในสถานีรถไฟฟ้าทดลองที่มีความซับซ้อนและมีจำนวนผู้อพยพมากได้อย่างมีประสิทธิภาพ ซึ่งโปรแกรมยูนิตียังมีอุปกรณ์เสริมที่ช่วยให้การจำลองการอพยพที่มีความซับซ้อนสามารถทำการแสดงผลในรูปแบบแอนิเมชันการเคลื่อนของฝูงชนในรูปแบบสามมิติได้อย่างเรียบง่าย

ผู้วิจัยได้นำสมมติฐานที่สมเหตุสมผลและข้อกำหนดปัญหาจากงานวิจัยของ [2] Min Zhou (2019) มาประยุกต์ใช้สำหรับแบบจำลองการอพยพฝูงชนด้วยคอมพิวเตอร์นี้เพื่อให้ง่ายต่อการศึกษาทดลอง ซึ่งจะมีการนำพฤติกรรมการเดินของคนเดินเท้าและประเภทของคนเดินเท้าภายในสถานีรถไฟฟ้าทดลอง รวมถึงได้มีการนำสถานการณ์ของการจำลองการอพยพฝูงชนภายใต้ กลยุทธ์การอพยพที่แตกต่างกัน และนำสถานีรถไฟฟ้าปักกิ่งที่ถูกใช้สำหรับศึกษาปัญหาการอพยพฝูงชนที่มีความแออัดของฝูงชนสูงมาเป็นสถานีรถไฟฟ้าทดลองสำหรับการสร้างแบบจำลองการอพยพฝูงชนด้วยคอมพิวเตอร์ โดยพื้นที่แพลตฟอร์มมีความกว้าง 120 เมตร ยาว 12 เมตร สถานีรถไฟฟ้าทดลองนี้มีทางออกทั้งสิ้น 3 ทางออก และภายในพื้นที่แพลตฟอร์มนี้มีสิ่งกีดขวางที่ไม่สามารถผ่านได้ เป็นสี่เหลี่ยมสีฟ้าดังรูปที่ 4.1 นอกจากนี้การทดสอบแบบจำลองการอพยพฝูงชนด้วยคอมพิวเตอร์จะถูกประเมินผลจากเวลาการอพยพฝูงชนตามมาตราฐานการออกแบบเพื่อปองกันอัคคีภัย ซึ่งเวลาที่ใชในการอพยพฝูงชนออกจากแพลตฟอร์มควรใชเวลาในการอพยพไมเกิน 4 นาที



รูปที่ 4-1 ตัวอย่างการกำหนดค่าของแพลตฟอร์มสถานีรถไฟฟ้าปักกิ่ง [15]

4.1 แบบจำลองการอพยพฝูงชนด้วยโปรแกรมไพทอน

แบบจำลองการอพยพฝูงชนด้วยโปรแกรมไพทอนเป็นโปรแกรมแบบจำลองการอพยพฝูงชนบนระนาบสองมิติโดยอาศัยแบบจำลองแรงทางสังคมและการค้นหาแบบเอสตาร์ในการจำลองพฤติกรรมการเดินเท้าและการค้นหาเส้นทางที่สั้นที่สุดในการอพยพของผู้อพยพภายในสถานีรถไฟฟ้าทดลองตามลำดับ ซึ่งโปรแกรมที่พัฒนาขึ้นเองทั้งหมดนั้นไม่เหมาะกับกรณีศึกษาที่มีผู้อพยพจำนวนมาก โดยผู้วิจัยกำหนดจำนวนของผู้อพยพเท่ากับ 50 คน โดยผู้อพยพทุกคนในสถานีรถไฟฟ้าทดลองเป็นประเภทผู้ติดตามทั้งหมด สถานีรถไฟฟ้าทดลองจะถูกแบ่งออกเป็นกริดเพื่อบอกถึงพื้นที่ที่สามารถเดินได้ ซึ่งขนาดของกริดมีอยู่ 2 ขนาด โดยขนาดแรกแต่ละกริดจะมีความกว้าง 4 เมตร ยาว 4 เมตร และในส่วนที่สองกริดจะมีความกว้าง 2 เมตร ยาว 2 เมตร นอกจากนี้กริดที่เป็นทางเดินในโปรแกรมไพทอนกำหนดให้มีค่าเท่ากับ 0 ส่วนที่เป็นสิ่งกีดขวางจะมีค่าเป็นค่า 1 ดังรูปที่ 4.2 นอกจากนี้ยังมีข้อกำหนดของแบบจำลองในด้านต่าง ๆ ดังนี้

```
78  maze = np.array([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
79                   [1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1],
80                   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
```


รูปที่ 4-2 ตัวอย่างการกำหนดพื้นที่บนโปรแกรมไพทอนที่มีขนาดกริด 4x4 เมตร

4.1.1 สถานการณ์ของการอพยพฝูงชนด้วยโปรแกรมไพทอน

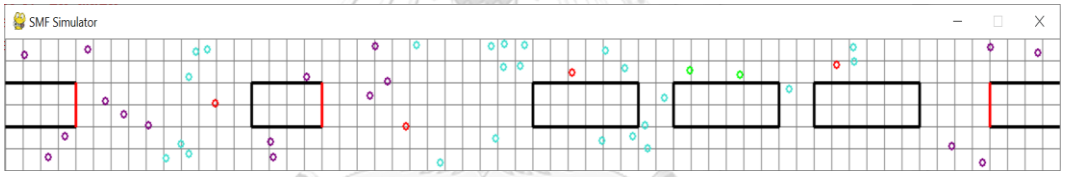
แบบจำลองการอพยพฝูงชนบนระนาบสองมิติ มีทั้งหมด 2 สถานการณ์ คือ การอพยพโดยปราศจากผู้นำ และ การอพยพพร้อมผู้นำ ดังนี้

1. การอพยพโดยปราศจากผู้นำ ผู้ติดตามเลือกทางออกที่ใกล้ที่สุดตามเส้นทางที่สั้นที่สุด ถ้าทางออกอยู่ในระยะสายตา สำหรับผู้ติดตามที่ไม่สามารถระบุตำแหน่งทางออกจะเลือกเคลื่อนที่ไปในทิศทางแบบสุ่ม
2. การอพยพพร้อมผู้นำที่สุ่มตำแหน่ง จำนวนผู้นำถูกกำหนดเท่ากับ 5 คน ซึ่งตำแหน่งเริ่มต้นของผู้นำจะถูกสุ่มอยู่อย่างกระจายตัวภายในสถานี ผู้นำจะไปยังทางออกที่ใกล้

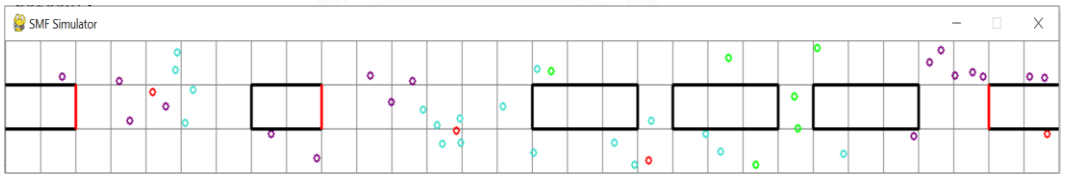
ที่สุดตามเส้นทางที่สั้นที่สุดพร้อมพาผู้ติดตามที่สามารถเห็นผู้นำได้ในระยะสายตาออกไปด้วย

4.1.2 ประเภทของคนเดินเท้าภายในสถานี

คนเดินเท้าภายในสถานีรถไฟฟ้าทดลองจะถูกแบ่งออกเป็น 2 ส่วน ในส่วนแรกคือผู้นำจะมีลักษณะเป็นวงกลมสีแดงดังรูปที่ 4.3 และ 4.4 ในส่วนที่สองคือผู้อพยพโดยจะมีการทำเป็นสีที่แตกต่างกันเพื่อให้เข้าใจถึงพฤติกรรมที่แตกต่างกัน กล่าวคือ วงกลมสีเขียวเป็นผู้อพยพที่ไม่สามารถเห็นทางออกหรือผู้นำได้ภายในระยะสายตาโดยระยะสายตาเท่ากับ 10 เมตร ซึ่งไม่สมารถเดินไปยังทางออกได้เอง ดังนั้นผู้อพยพจะพยายามเดินสุ่มเพื่อให้เจอผู้นำหรือทางออกภายในระยะสายตา วงกลมสีม่วงเป็นผู้อพยพที่เห็นทางออกภายในระยะสายตาซึ่งสามารถเดินไปยังทางออกได้โดยปราศจากผู้นำ และสุดท้ายวงกลมสีฟ้าเป็นผู้อพยพที่ไม่สามารถระบุทางออกแต่สามารถระบุตำแหน่งของผู้นำได้ภายในระยะสายตา ผู้อพยพก็จะเดินตามผู้นำจนสามารถอพยพออกจากสถานีรถไฟฟ้าทดลอง



รูปที่ 4-3 ตัวอย่างการแสดงผลแบบจำลองการอพยพด้วยโปรแกรมไพทอนขนาดกริด 2x2 เมตร

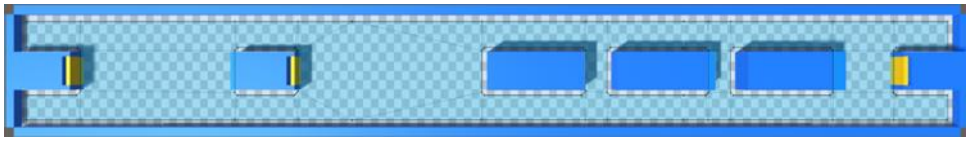


รูปที่ 4-4 ตัวอย่างการแสดงผลแบบจำลองการอพยพด้วยโปรแกรมไพทอนขนาดกริด 4x4 เมตร

4.2 แบบจำลองการอพยพฝูงชนด้วยโปรแกรมยูนิตี

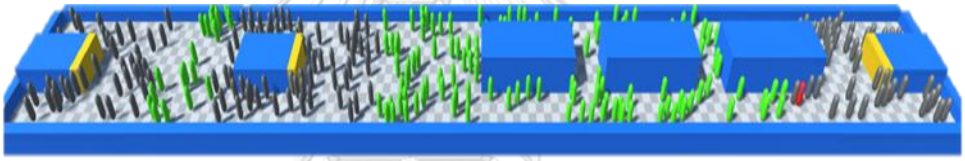
แบบจำลองการอพยพฝูงชนด้วยโปรแกรมยูนิตีเป็นโปรแกรมแบบจำลองการอพยพฝูงชนบนโลกสามมิติ ซึ่งโปรแกรมถูกพัฒนาเนื่องจากข้อจำกัดของแบบจำลองการอพยพฝูงชนด้วยโปรแกรมไพทอนที่ใช้ในกรณีศึกษาที่มีผู้อพยพจำนวนน้อย โดยจะทำการทดลองเหมือนกับแบบจำลองในหัวข้อที่ 4.1 ทุกประการเพื่อเปรียบเทียบกับแบบจำลองการอพยพฝูงชนด้วยโปรแกรมไพทอน นอกจากนี้ได้ใช้โปรแกรมที่พัฒนาขึ้นกับกรณีศึกษาที่มีผู้อพยพจำนวนมากตามงานวิจัยหมายเลข 5 เพื่อเปรียบเทียบ

ประสิทธิภาพการอพยพฝูงชนด้วยวิธีการที่จะนำเสนอในบทต่อไป ซึ่งแบบจำลองการอพยพฝูงชนด้วยโปรแกรมยูนิตีใช้อุปกรณ์เสริมของโปรแกรมยูนิตีที่มีชื่อว่า โปรบิวเดอร์ เพื่อสร้างสถานีรถไฟฟ้าทดลองและอุปกรณ์เสริมที่มีชื่อว่า การค้นหาเส้นทางและการนำทางเพื่อสร้างตาข่ายนำทางที่บอกพื้นทที่ที่ผู้อพยพหรือผู้นำอพยพสมารถเดินได้ในโลกสามิติดังรูปที่ 4.5



รูปที่ 4-5 โครงตาข่ายหลายเหลี่ยมที่แสดงถึงพื้นที่ที่สามารถเดินได้

นอกจากนี้ยังใช้อุปกรณ์เสริมของโปรแกรมยูนิตีที่มีชื่อว่า ยูนิติเอ็มแอลเอเจนต์ มาฝึกฝนเอเจนต์ด้วยการเรียนรู้เสริมกำลังเพื่อให้ได้ประสิทธิภาพการอพยพที่ดียิ่งขึ้น ซึ่งมีข้อกำหนดของแบบจำลองในด้านต่าง ๆ ดังนี้



รูปที่ 4-6 ตัวอย่างการแสดงผลแบบจำลองการอพยพฝูงชนด้วยโปรแกรมยูนิตี

4.2.1 สถานการณ์ของการอพยพฝูงชนด้วยโปรแกรมยูนิตี

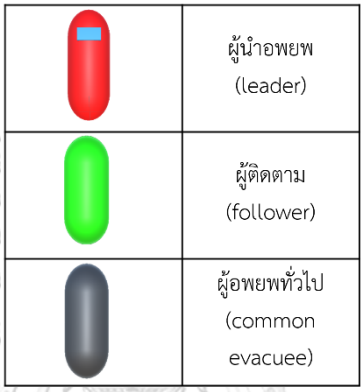
แบบจำลองการอพยพฝูงชนบนระนาบสองมิติ มีทั้งหมด 4 สถานการณ์ ดังนี้

1. การอพยพโดยปราศจากผู้นำ ผู้ติดตามเลือกทางออกที่ใกล้ที่สุดตามเส้นทางที่สั้นที่สุด ถ้าทางออกอยู่ในระยะสายตา สำหรับผู้ติดตามที่ไม่สามารถระบุตำแหน่งทางออกจะเลือกเคลื่อนที่ไปในทิศทางแบบสุ่ม
2. การอพยพพร้อมผู้นำ 5 คน ที่สุ่มตำแหน่ง จำนวนผู้นำถูกกำหนดเท่ากับ 5 คน ซึ่งตำแหน่งเริ่มต้นของผู้นำจะถูกสุ่มอยู่อย่างกระจายตัวภายในสถานี ผู้นำจะไปยังทางออกที่ใกล้ที่สุดตามเส้นทางที่สั้นที่สุดพร้อมพาผู้ติดตามที่สามารถเห็นผู้นำได้ในระยะสายตาออกไปด้วย
3. การอพยพพร้อมผู้นำ 10 คน ที่สุ่มตำแหน่ง จำนวนผู้นำถูกกำหนดเท่ากับ 10 คน ซึ่งตำแหน่งเริ่มต้นของผู้นำจะถูกสุ่มอยู่อย่างกระจายตัวภายในสถานี ผู้นำจะไปยังทางออกที่ใกล้ที่สุดตามเส้นทางที่สั้นที่สุดพร้อมพาผู้ติดตามที่สามารถเห็นผู้นำได้ในระยะสายตาออกไปด้วย

4. การอพยพด้วยเอเจนต์ โดยเอเจนต์จะถูกฝึกด้วยปัญญาประดิษฐ์ให้เป็นผู้นำการอพยพ จำนวนเอเจนต์ถูกกำหนดเท่ากับ 1 คน ซึ่งตำแหน่งเริ่มต้นของเอเจนต์จะถูกสุ่มอยู่อย่างกระจายตัวภายในสถานี เอเจนต์จะพยายามเดินไปหาผู้อพยพทุกคนภายในสถานีเพื่อบอกทางออกที่ใกล้ที่สุดตามเส้นทางที่สั้นที่สุด

4.1.2 ประเภทของผู้อพยพในสถานี

คนเดินเท้าภายในสถานีจะถูกแบ่งออกเป็น 3 ประเภท ดังนี้



รูปที่ 4-7 ประเภทของผู้อพยพภายในสถานีบนโลกสามมิติ

1. ผู้นำอพยพ (Leader) เป็นเจ้าหน้าที่สถานีหรือพนักงานรักษาความปลอดภัย ซึ่งจะถูกอบรมให้รู้ถึงทางออกของสถานีและเส้นการอพยพ
2. ผู้อพยพทั่วไป (Common evacuee) เป็นคนที่ใช้งานรถไฟฟ้าเป็นประจำ ซึ่งรู้ทางออกเป็นอย่างดีส่งผลให้สามารถเคลื่อนไปยังทางออกที่ใกล้ที่สุดตามเส้นทางที่สั้นที่สุดโดยไม่ต้องการคำแนะนำจากผู้อื่น
3. ผู้ติดตาม (Follower) จะมีขั้นตอนการอพยพดังนี้ (ก) มักจะเคลื่อนไปยังทางออกที่ใกล้ที่สุดหากพวกเขาเห็นทางออกภายในระยะสายตา (ข) มักจะเคลื่อนไปหาผู้นำที่ใกล้ที่สุดหากพวกเขาสามารถเห็นผู้นำในระยะสายตา (ค) เดินตามป้ายแนะนำทางออกกรณีไม่เห็นผู้นำอพยพหรือทางออก (ง) หากไม่เห็นป้ายหรือผู้นำจะเคลื่อนที่ในทิศทางสุ่มกำหนดให้ผู้ติดตามสามารถรับข้อมูลเกี่ยวกับทางออกและเส้นทางอพยพ และทำตามผู้นำที่ตัวเองเลือกเมื่อเริ่มอพยพโดยทันที

กำหนดให้คนเดินเท้าภายในสถานีรถไฟฟ้าทดลองทุกประเภทมีค่าพารามิเตอร์ทั่วไปของแบบจำลองแรงทางสังคมคงที่ดังตารางที่ 4.1

ตารางที่ 4-1 ตารางแสดงพารามิเตอร์ทั่วไปของแบบจำลองแรงทางสังคม

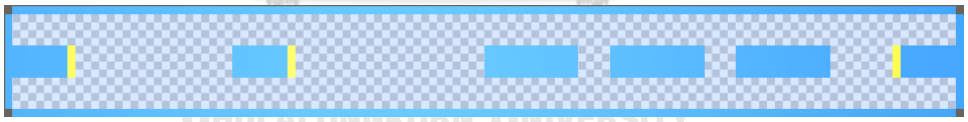
สัญลักษณ์	ความหมาย	ค่า
m	มวลของคนเดินเท้า	60 กิโลกรัม
r	รัศมีของคนเดินเท้า	0.3 เมตร
τ	เวลาเฉพาะ	0.5 วินาที
A	ค่าความหนาแน่นของแรงหลีกเลี่ยง	2000 นิวตัน
B	ค่าสัมประสิทธิ์การหลีกเลี่ยง	0.08 เมตร
v^o	ความเร็วที่ต้องการ	1.34 เมตรต่อวินาที
K	ค่าสัมประสิทธิ์แรงเสียดทานเลื่อน	240000 กิโลกรัมต่อเมตรต่อวินาที
k	ค่าสัมประสิทธิ์การกดทับของร่างกาย	120000 กิโลกรัมต่อวินาทีต่อวินาที

บทที่ 5
วิธีการอพยพฝูงชนด้วยวิธีการที่นำเสนอ

ในบทนี้จะนำเสนอวิธีการอพยพฝูงชนพร้อมผู้นำในสถานีรถไฟฟ้าที่ถูกฝึกอบรมด้วยเทคนิคการเพิ่มประสิทธิภาพนโยบายใกล้เคียง (Proximal Policy Optimization: PPO) ซึ่งเป็นเทคนิคหนึ่งของวิธีการเรียนรู้แบบเสริมกำลัง โดยใช้โปรแกรมยูนิตี ML-agents เพื่อฝึกอบรมเอเจนต์หรือผู้นำ โดยวิธีการที่นำเสนอจะถูกกำหนดองค์ประกอบหลักของการเรียนรู้แบบเสริมกำลังดังต่อนี้ หัวข้อที่ 5.1 จะกล่าวถึงพื้นที่ที่ถูกใช้เป็นสถานที่ฝึกฝนเอเจนต์ หัวข้อที่ 5.2 จะกล่าวถึงลักษณะ พฤติกรรมและเป้าหมายของเอเจนต์ หัวข้อที่ 5.3 จะกล่าวถึงสิ่งที่เอเจนต์ทำการสังเกตุเพื่อช่วยสำหรับการตัดสินใจในการทำการกระทำ หัวข้อที่ 5.4 จะกล่าวถึงรูปแบบการการะทำของเอเจนต์ที่เป็นไปได้ และหัวข้อที่ 5.5 จะกล่าวถึงการให้รางวัลแก่เอเจนต์เพื่อให้เอเจนต์เกิดการเรียนรู้

5.1 สิ่งแวดล้อม (environment)

สิ่งแวดล้อมเป็นสถานีรถไฟฟ้าทดลองที่จะทำการฝึกเอเจนต์หรือพนักงานรักษาความปลอดภัยให้เป็นผู้นำการอพยพในสถานีรถไฟฟ้าทดลองเพื่อทำการอพยพคนภายในระยะเวลาที่กำหนดด้วยเทคนิคการเพิ่มประสิทธิภาพนโยบายใกล้เคียง ซึ่งสถานีรถไฟฟ้าทดลองจะถูกสร้างด้วยโปรแกรมยูนิตี



รูปที่ 5-1 สถานีรถไฟฟ้าทดลอง

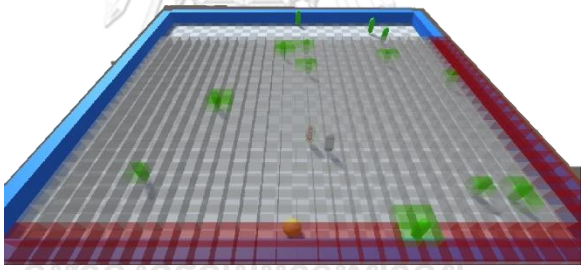
5.2 เอเจนต์ (Agent)

เอเจนต์ คือพนักงานรักษาความปลอดภัยจะทำการสังเกตผู้อพยพโดยการรับค่าตำแหน่งของผู้โดยสารในสถานีรถไฟฟ้าทดลองเพื่อประกอบการตัดสินใจสำหรับเลือกการกระทำ ซึ่งเอเจนต์จะมีทั้งการกระทำทั้งหมด 6 รูปแบบซึ่งสอดคล้องกับการเคลื่อนไปตามทิศทางที่แตกต่างกันสี่ทิศทาง การหมุนตามเข็มนาฬิกาและทวนเข็มนาฬิกา เป้าหมายของเอเจนต์ คือเอเจนต์ต้องเรียนรู้ที่จะเดินไปหาผู้อพยพภายในสถานีไฟฟ้าทดลองให้ได้มากที่สุดเพื่อให้ข้อมูลเกี่ยวกับทางออกและเส้นทางการอพยพ

โดยเอเจนต์ต้องหลีกเลี่ยงการชนกำแพง ตำแหน่งของผู้โดยสารเป็นแบบสุ่มอย่างกระจายตัวภายในสถานีรถไฟฟ้าทดลอง และตำแหน่งทางออกเป็นแบบคงที่

5.3 ปริภูมิการสังเกตการณ์ (Observation space)

การสังเกต (observation) เป็นวิธีสำหรับเอเจนต์ในการวัดสถานะของสิ่งแวดล้อม สิ่งสำคัญคือการให้ข้อมูลที่เกี่ยวข้องน้อยเกินไปจะทำให้เอเจนต์ไม่สามารถเรียนรู้และการให้ข้อมูลที่ไม่เกี่ยวข้องมากเกินไปทำให้ไม่มีข้อมูลที่เป็นประโยชน์ในการแก้ปัญหา ซึ่งการสังเกตถูกรวบรวมโดยคอมโพเนนต์ของเอเจนต์เป็นลิสต์ของค่าทศนิยม เพื่อเพื่อผลลัพธ์ที่ดีที่สุดการสังเกตจะถูกปรับอยู่ในช่วง [-1, 1] หรือ [0, 1] ซึ่งช่วยให้เครือข่ายประสาทพีพีโอ (PPO) ค้นหาวิธีแก้ปัญหาได้รวดเร็วยิ่งขึ้น การสังเกตจะถูกเพิ่มเข้าไปทุกครั้งที่เอเจนต์เข้าสู่สถานะใหม่และก่อนที่เอเจนต์จะลงมือทำ ซึ่งสิ่งที่ทำการสังเกต คือ ตำแหน่งของเอเจนต์ และใช้กริดเซนเซอร์ (grid sensor) ตรวจจับวัตถุที่มีชื่อ “Follower” และ “Wall” เพื่อระบุตำแหน่งของวัตถุนั้น โดยกริดเซนเซอร์จะมีลักษณะสีเขียวเมื่อเจอวัตถุที่มีชื่อ “Follower” และเป็นสีแดงเมื่อเจอวัตถุที่มีชื่อ “Wall” ตามที่ได้เห็นจากรูปที่ 5.2



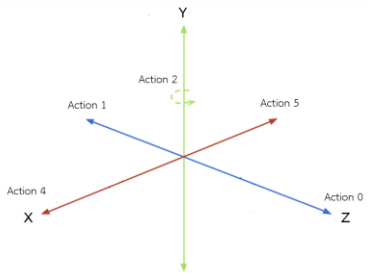
รูปที่ 5-2 ตัวอย่างการใช้งานกริดเซนเซอร์

5.4 ปริภูมิการกระทำ (Action space)

เอเจนต์สามารถเลือกการกระทำจากลิสต์ของการกระทำได้เพียงรายการเดียวเท่านั้น ซึ่งเป็นจำนวนเต็มแบบไม่ต่อเนื่อง (discrete) เพื่อกำหนดทิศทางการเคลื่อนที่ของเอเจนต์ เอเจนต์มี 1 สาขาการกระทำที่ไม่ต่อเนื่อง (discrete action branch) ที่มี 6 การกระทำซึ่งสอดคล้องกับการหมุนตามเข็มนาฬิกาและทวนเข็มนาฬิกา และเคลื่อนไปตามทิศทางที่แตกต่างกันสี่ทิศทางซึ่งถูกกำหนดดังนี้

- Action (0) เป็นการเคลื่อนที่ไปตามแกน z ขนาด 1 หน่วย
- Action (1) เป็นการเคลื่อนที่ไปตามแกน z ขนาด -1 หน่วย
- Action (2) เป็นการหมุนรอบแกน y ตามเข็มนาฬิกา

- Action (3) เป็นการหมุนรอบแกน y ทวนเข็มนาฬิกา
- Action (4) เป็นการเคลื่อนที่ไปตามแกน x ขนาด 1 หน่วย
- Action (5) เป็นการเคลื่อนที่ไปตามแกน x ขนาด -1 หน่วย



รูปที่ 5-3 ตัวอย่างการเคลื่อนที่ของเอเจนต์

5.5 ปริภูมิรางวัล (Reward space)

สิ่งที่สำคัญสำหรับการเรียนรู้แบบเสริมกำลัง คือต้องออกแบบฟังก์ชันการให้รางวัลที่เหมาะสมสำหรับเอเจนต์เพื่อเรียนรู้นโยบายหรือพฤติกรรมของผู้นำอพยพให้นำไปสู่การเลือกการกระทำที่เหมาะสมที่สุด โดยวิธีการที่นำเสนอนี้ได้กำหนดฟังก์ชันการให้รางวัลดังนี้

- เอเจนต์จะไม่ได้รับรางวัลหรือโดนลงโทษขณะเคลื่อนที่
- เอเจนต์จะได้รับรางวัลเมื่อสามารถให้ข้อมูลแก่ผู้ติดตามเท่ากับ 1 ซึ่งจะเพิ่มเรื่อยๆตามจำนวนผู้ติดตามที่ได้รับข้อมูล
- เอเจนต์ชนกำแพงจะถูกลงโทษ -1 คะแนน และจบตอน
- กรณีไม่เหลือผู้อพยพในสถานีภายในระยะเวลาที่กำหนดจะได้รับรางวัลขนาดใหญ่เท่ากับ 5

บทที่ 6
ผลการทดสอบ

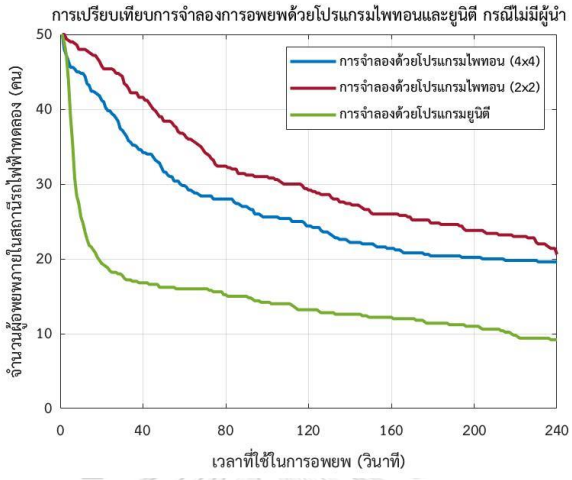
ในบทนี้จะกล่าวถึงผลทดสอบของการจำลองการอพยพฝูงชนด้วยโปรแกรมคอมพิวเตอร์ ซึ่งการทดสอบนี้ถูกแบ่งออกเป็น 3 ส่วน ในส่วนแรกจะเปรียบเทียบการจำลองการอพยพฝูงชนแบบสองมิติด้วยโปรแกรมไพทอนกับการจำลองการอพยพฝูงชนแบบสามมิติด้วยโปรแกรมยูนิตีที่มีผู้อพยพจำนวนไม่มาก ส่วนที่สองจะเป็นผลการฝึกอบรมเอเจนต์ด้วยการเรียนรู้แบบเสริมกำลังให้กลายเป็นผู้นำอพยพและมีการจำลองการอพยพฝูงชนพร้อมเอเจนต์ 1 คนแบบสามมิติด้วยโปรแกรมยูนิตีที่มีผู้อพยพจำนวนมากเพื่อเปรียบเทียบประสิทธิภาพการอพยพระหว่างเอเจนต์ที่ถูกฝึกอบรมด้วยระยะเวลาที่แตกต่างกัน ในส่วนที่สามได้มีการจำลองการอพยพฝูงชนแบบสามมิติด้วยโปรแกรมยูนิตีที่มีผู้อพยพจำนวนมากเพื่อทดสอบสมรรถนะของการอพยพฝูงชนด้วยวิธีการที่ได้นำเสนอหรือการอพยพพร้อมผู้นำ 1 คนที่ถูกฝึกด้วยการเรียนรู้แบบเสริมกำลังพร้อมเปรียบเทียบกับการการอพยพฝูงชนด้วยวิธีการอื่น ๆ ซึ่งจะถูกประเมินประสิทธิภาพของวิธีการที่ได้นำเสนอด้วยจำนวนผู้อพยพที่ไม่สามารถออกจากสถานีรถไฟฟ้าทดลองตามระยะเวลาการอพยพที่กำหนด

6.1 ผลการทดสอบของโปรแกรมการอพยพฝูงชนบนระนาบสองมิติกับโลกสามมิติ

ในการทดสอบนี้ วิธีการการอพยพฝูงชนด้วยวิธีการโดยปราศจากผู้นำอพยพและการอพยพพร้อมผู้นำ 5 คน จะถูกนำมาใช้ในการทดลองเพื่อที่จะเปรียบเทียบโปรแกรมการอพยพฝูงชนบนระนาบสองมิติที่พัฒนาขึ้นเองทั้งหมดด้วยไพทอนกับโปรแกรมการอพยพฝูงชนบนโลกสามมิติ ซึ่งตำแหน่งของผู้อพยพและตำแหน่งของผู้นำอพยพจะถูกสุ่มอย่างกระจายตัวภายในสถานีรถไฟฟ้าทดลอง โดยผู้นำอพยพจะเดินไปยังทางออกที่ใกล้ตัวเองที่สุดพร้อมพาผู้อพยพที่ไม่สามารถอพยพได้ด้วยตัวเอง ซึ่งผู้อพยพจำเป็นต้องเห็นผู้นำอพยพในระยะสายตาจึงจะสามารถเดินตามผู้นำอพยพจนสามารถออกจากสถานีรถไฟฟ้าทดลอง

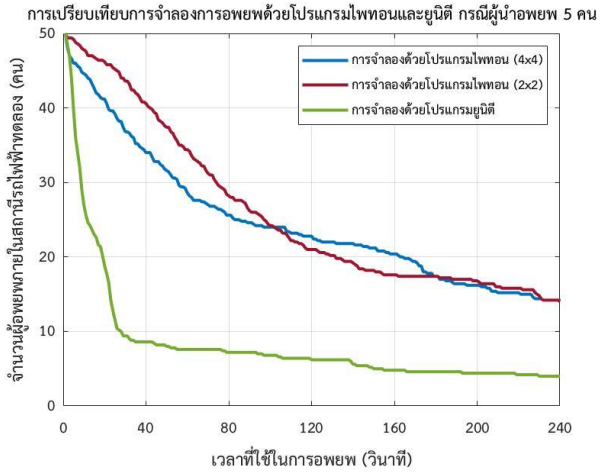
การประเมินประสิทธิภาพของการอพยพในการทดสอบนี้จะวัดจากจำนวนผู้อพยพเฉลี่ยที่ไม่สามารถออกจากสถานีรถไฟฟ้าทดลองตามระยะเวลาการอพยพที่กำหนดในสถานการณ์การอพยพกรณีไม่มีผู้นำอพยพและกรณีมีผู้นำอพยพ 5 คน ซึ่งจะเปรียบเทียบกับการจำลองการอพยพด้วยโปรแกรมการอพยพฝูงชนบนระนาบสองมิติด้วยไพทอนกับโปรแกรมการอพยพฝูงชนบนโลกสามมิติด้วยยูนิตี รูปที่ 6.1 แสดงการเปรียบเทียบผลการจำลองการอพยพด้วยโปรแกรมไพทอนกับยูนิตี กรณีไม่มีผู้นำอพยพ โดยทำการวัดจากจำนวนผู้อพยพภายในสถานีรถไฟฟ้าทดลอง จะเห็นว่าการจำลองด้วยโปรแกรมยูนิตีจำนวนผู้อพยพจะลดลงอย่างรวดเร็วในช่วง 40 วินาทีแรกของเวลาที่ใช้ในการอพยพ หลังจากนั้นจำนวนผู้อพยพจะค่อย ๆ ลดลงจนครบเวลาที่การอพยพที่ระบุ ซึ่งไม่สามารถ

สามารถทำการอพยพได้สำเร็จโดยจะเหลือผู้อพยพติดค้างเฉลี่ยภายในสถานีรถไฟฟ้าทดลองเท่ากับ 9 คน แต่การจำลองด้วยโปรแกรมไพทอนจำนวนผู้อพยพจะค่อย ๆ ลดลงตั้งแต่เริ่มการทดสอบจนกระทั่งสิ้นสุดการทดสอบ โดยจะเหลือผู้อพยพติดเฉลี่ยภายในสถานีรถไฟฟ้าทดลองเท่ากับ 20 คน



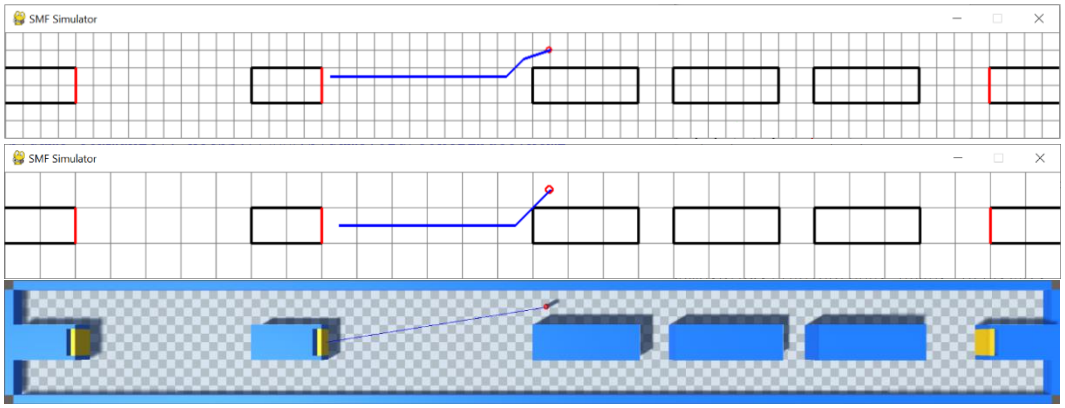
รูปที่ 6-1 เปรียบเทียบผลการจำลองด้วยโปรแกรมไพทอนและยูนิตี กรณีไม่มีผู้นำอพยพ

รูปที่ 6.2 แสดงการเปรียบเทียบผลการจำลองการอพยพด้วยโปรแกรมไพทอนกับยูนิตี กรณีผู้นำอพยพ 5 คน โดยทำการวัดจากจำนวนผู้อพยพภายในสถานีรถไฟฟ้าทดลอง จะเห็นว่าการจำลองด้วยโปรแกรมยูนิตีจำนวนผู้อพยพจะลดลงอย่างรวดเร็วในช่วง 40 วินาทีแรกของเวลาที่ใช้ในการอพยพจนจำนวนผู้อพยพเหลือเท่ากับ 9 คน หลังจากนั้นจำนวนผู้อพยพจะค่อย ๆ ลดลงจนครบเวลาที่การอพยพที่ระบุ ซึ่งไม่สามารถสามารถทำการอพยพได้สำเร็จโดยจะเหลือผู้อพยพติดค้างเฉลี่ยภายในสถานีรถไฟฟ้าทดลองเท่ากับ 4 คน แต่การจำลองด้วยโปรแกรมไพทอนจำนวนผู้อพยพจะค่อย ๆ ลดลงตั้งแต่เริ่มการทดสอบจนกระทั่งสิ้นสุดการทดสอบ โดยจะเหลือผู้อพยพติดค้างเฉลี่ยภายในสถานีรถไฟฟ้าทดลองเท่ากับ 15 คน

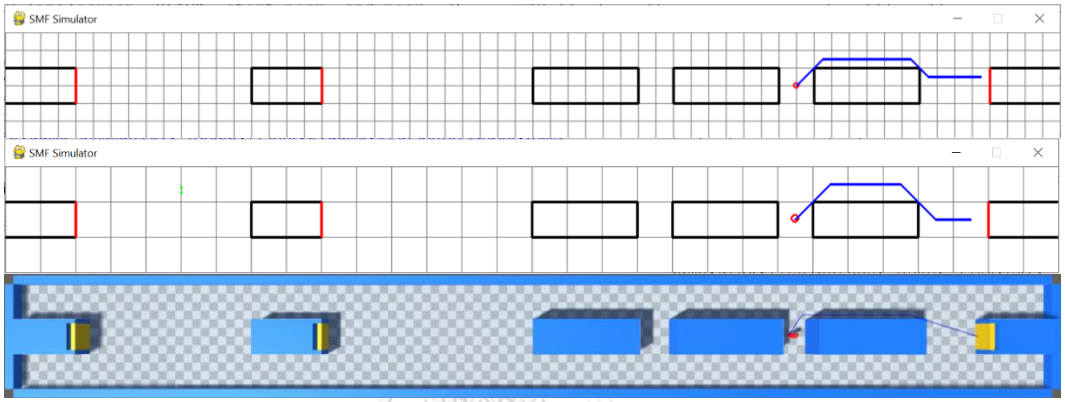


รูปที่ 6-2 เปรียบเทียบผลการจำลองด้วยโปรแกรมไพทอนและยูนิตี กรณีผู้นำอพยพ 5 คน

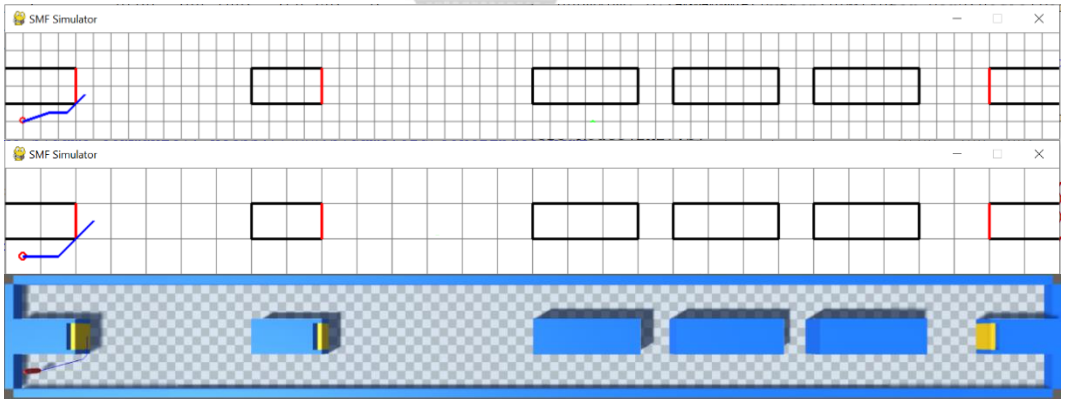
จากผลการการจำลองการอพยพด้วยโปรแกรมไพทอนและยูนิตี จะเห็นได้ว่าจำนวนผู้อพยพภายในสถานีรถไฟฟ้าทดลองมีแนวโน้มลดลงเหมือนกัน แต่มีความแตกต่างกันตรงที่ความเร็วในการอพยพคนออกจากสถานี ซึ่งจำนวนผู้อพยพเฉลี่ยภายในสถานีรถไฟฟ้าทดลองของการจำลองด้วยโปรแกรมไพทอนที่ขนาดกริด 2 และ 4 เมตร นั้นจะลดลงอย่างต่อเนื่องจนครบเวลาการทดลอง แต่ในส่วนของการจำลองด้วยโปรแกรมยูนิตีจะลดลงอย่างรวดเร็วมากในช่วงแรกของการทดลองและค่อย ๆ ลดในช่วงที่เหลือของการทดลอง ซึ่งเป็นผลมากจากกระบวนการค้นหาเส้นทางที่แตกต่างกัน โดยกระบวนการค้นหาเส้นทางสำหรับการจำลองการอพยพด้วยโปรแกรมไพทอนจะใช้อัลกอริธึมเอสตาร์ที่ทำการแบ่งสถานีรถไฟฟ้าทดลองออกเป็นกริด เพราะฉะนั้นทำให้ได้เส้นทางที่เป็นการเชื่อมต่อระหว่างกริดเริ่มต้นจนไปถึงกริดที่เป็นทางออก แต่ในส่วนการจำลองการอพยพด้วยโปรแกรมยูนิตีมีขั้นตอนการหาเส้นทางการใช้อุปกรณ์เสริมที่ชื่อว่า “การค้นหาเส้นทางและการนำทาง (Pathfinding and Navigation)” ซึ่งเป็นการใช้นาฟเมช (NavMesh) ที่มีลักษณะเป็นโครงข่ายหลายเหลี่ยมที่ได้กล่าวในหัวข้อที่ 2.3.2 ทำงานร่วมกับอัลกอริธึมเอสตาร์และแนวคิดเกี่ยวกับสามเหลี่ยมเพื่อที่จะเพิ่มประสิทธิภาพการค้นหาเส้นทางที่สั้นที่สุด กล่าวคือรูปหลายเหลี่ยมของของนาฟเมชจะถูกแทนที่ด้วยสามเหลี่ยมทำให้สามารถได้เส้นทางการอพยพที่สั้นกว่าที่คำนวณได้จากอัลกอริธึมเอสตาร์เพียงอย่างเดียว ด้วยเหตุผลด้านกระบวนการค้นหาเส้นทางของทั้งสองโปรแกรมที่ได้กล่าวมาทำให้เส้นทางอพยพที่ได้จากโปรแกรมยูนิตีนั้นเป็นเส้นทางอพยพที่สั้นกว่าเส้นทางอพยพที่ได้จากโปรแกรมไพทอน ดังแสดงในรูปที่ 6.3



(ก)



(ข)



(ค)

รูปที่ 6-3 เปรียบเทียบเส้นทางการอพยพของผู้อพยพบนโปรแกรมไพทอนกับยูนิตี

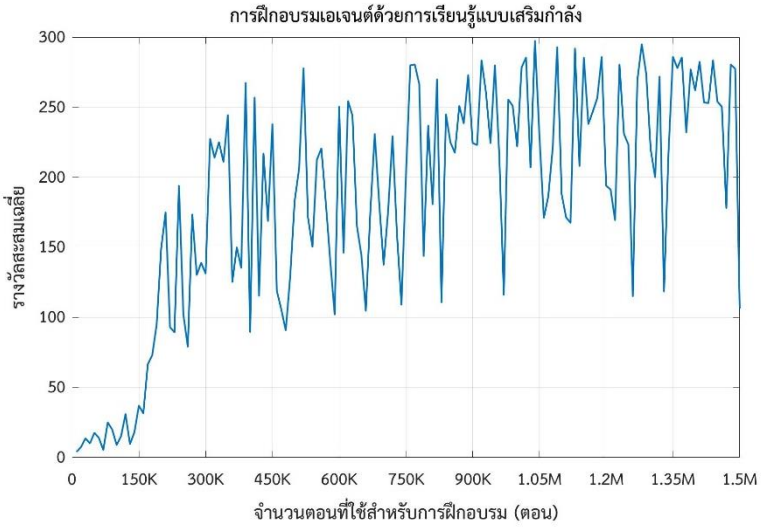
6.2 ผลการทดสอบการฝึกอบรมเอเจนต์ด้วยการเรียนรู้แบบเสริมกำลัง

ในการทดสอบนี้ เป็นการฝึกอบรมเอเจนต์ด้วยการเรียนรู้แบบเสริมกำลังด้วยเทคนิคการเพิ่มประสิทธิภาพนโยบายใกล้เคียง ให้กลายเป็นผู้นำอพยพ โดยแบบจำลองการฝึกอบรมเอเจนต์มีการกำหนดค่าฝึกอบรมเอเจนต์ทั่วไปดังรูปที่ 6-4 ซึ่งจำนวนขั้นตอนทั้งหมดที่เอเจนต์ต้องดำเนินการใน

สิ่งแวดล้อมเท่ากับ 1,500,000 ตอน และมีการเก็บสมองอัจฉริยะของเอเจนต์ที่ผ่านการฝึกอบรมทุก ๆ 150,000 ตอน และในการจำลองการฝึกอบรมนี้มีการเร่งเร่งเวลาเป็น 20 เท่าจากเวลาจริงเพื่อทำให้การฝึกอบรมเอเจนต์นั้นไวยิ่งขึ้น หลังจากเอเจนต์ผ่านการฝึกอบรมแล้วจะนำสมองอัจฉริยะมาทำการเปรียบเทียบประสิทธิการอพยพของการจำลองการอพยพฝูงชนพร้อมเอเจนต์ 1 คนที่มีสมองอัจฉริยะที่แตกต่างกัน

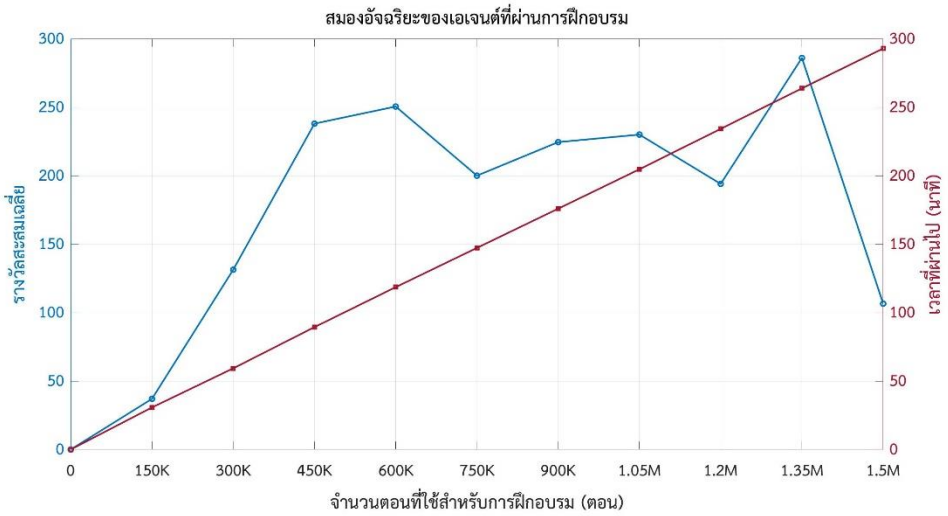
```
behaviors:
  Evacuation:
    trainer_type: ppo
    hyperparameters:
      batch_size: 1024
      buffer_size: 10240
      learning_rate: 0.0003
      beta: 0.005
      epsilon: 0.2
      lambd: 0.95
      num_epoch: 3
      learning_rate_schedule: linear
      beta_schedule: linear
      epsilon_schedule: linear
    keep_checkpoints: 11
    checkpoint_interval: 150000
    max_steps: 1500000
engine_settings:
  time_scale: 20.0
```

รูปที่ 6-4 ตัวอย่างการกำหนดค่าฝึกอบรมเอเจนต์ทั่วไป



รูปที่ 6-5 กราฟการเรียนรู้การเสริมแรงในแง่ของรางวัลสะสม

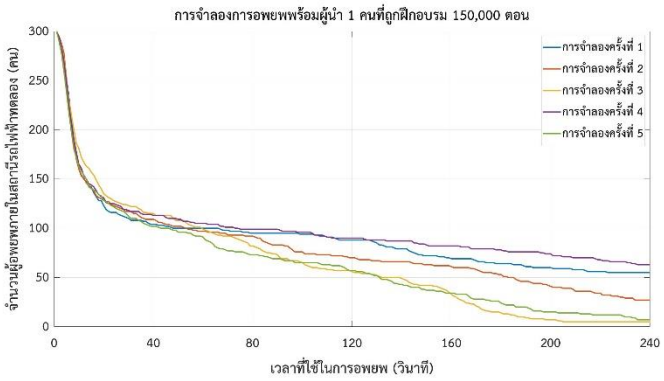
สำหรับการฝึกอบรมเอเจนต์เพื่อเป็นผู้นำอพยพ ผู้วิจัยใช้การเรียนรู้แบบเสริมกำลังที่มีชื่อเทคนิคว่า การเพิ่มประสิทธิภาพนโยบายใกล้เคียง (Proximal Policy Optimization; PPO) เพื่อฝึกอบรมเอเจนต์ ซึ่งจะทำการฝึกให้เอเจนต์กลายเป็นผู้นำอพยพที่เดินหาผู้อพยพทุกคนภายในสถานีรถไฟฟ้าทดลอง ในช่วงแรกของการฝึกจะกำหนดให้ผู้อพยพหยุดนิ่ง เมื่อเอเจนต์เดินชนกับผู้อพยพจะได้รับคะแนน แต่ถ้าเดินชนกำแพงจะถูกหักคะแนนและตัดจบตอน ซึ่งคะแนนจะถูกเก็บสะสมไปเรื่อย ๆ โดยคะแนนเพิ่มขึ้นในช่วงการฝึกที่ประสบความสำเร็จ จากรูปที่ 6-5 แสดงกราฟการเรียนรู้แบบเสริมกำลังเป็นรางวัลสะสมที่แสดงผลหลังการฝึก 1,500,000 ตอน ในช่วงเริ่มต้นของการฝึกอบรมเอเจนต์รางวัลสะสมค่อย ๆ เพิ่มขึ้นอย่างช้า เนื่องจากเอเจนต์นั้นเดินไปชนผู้อพยพน้อยครั้งแต่เดินชนกำแพงอยู่บ่อยครั้ง หลังจากนั้นในช่วงที่ 150,000 ถึง 200,000 ตอน รางวัลสะสมเพิ่มขึ้นอย่างรวดเร็วจาก 35 เป็นประมาณ 170 หลังจากที่เอเจนต์ได้รับการฝึกฝนมาแล้ว 300,000 ตอน รางวัลสะสมเริ่มมีลักษณะลู่เข้าโดยมีค่ารางวัลสะสมเฉลี่ยอยู่ที่ระหว่าง 100 ถึง 300 ซึ่งหมายความว่าเอเจนต์ที่ผ่านการฝึกอบรมเรียนรู้ที่จะเข้าหาผู้ติดตามทุกคนอย่างมีประสิทธิภาพมากขึ้นในตอนต่อ ๆ ไปหรือเอเจนต์เรียนรู้จนสามารถแก้ปัญหาการอพยพนี้จนค่ารางวัลสะสมเฉลี่ยเริ่มคงที่



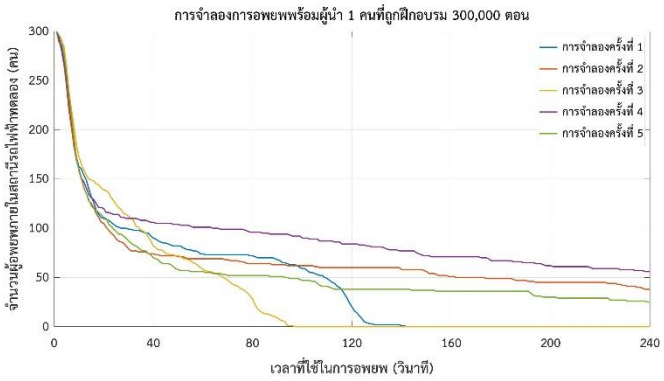
รูปที่ 6-6 กราฟแสดงเวลาและการเรียนรู้ของสมองอัจฉริยะที่ผ่านการฝึกอบรม

หลังจากการฝึกอบรมเอเจนต์ด้วยการเรียนรู้แบบเสริมกำลังการฝึกอบรมเอเจนต์ทั้งสิ้น 1,500,000 ตอนและบันทึกสมองอัจฉริยะของเอเจนต์ที่ผ่านการฝึกอบรมทุก ๆ 150,000 ตอน ดังนั้นจะได้สมองอัจฉริยะของเอเจนต์ทั้งสิ้น 10 สมอง จากรูปที่ 6-6 แสดงให้เห็นว่า สมองแรกใช้เวลาฝึกอบรม 30 นาทีและมีรางวัลสะสมเฉลี่ยเท่ากับ 37 สมองที่สองใช้เวลาฝึกอบรม 59 นาทีและมีรางวัลสะสมเฉลี่ยเท่ากับ 131.3 สมองที่สามใช้เวลาฝึกอบรม 89 นาทีและมีรางวัลสะสมเฉลี่ยเท่ากับ 238 สมองที่สี่ใช้เวลาฝึกอบรม 118 นาทีและมีรางวัลสะสมเฉลี่ยเท่ากับ 250.5 สมองที่ห้าใช้เวลาฝึกอบรม 147 นาทีและมีรางวัลสะสมเฉลี่ยเท่ากับ 200 สมองที่หกใช้เวลาฝึกอบรม 175 นาทีและมี

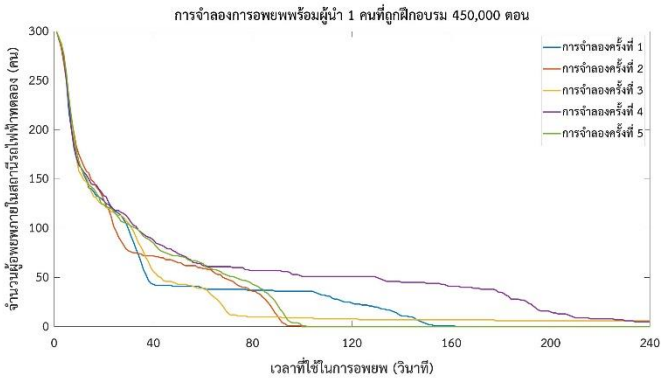
รางวัลสะสมเฉลี่ยเท่ากับ 224.5 สมองที่เจ็ดใช้เวลาฝึกอบรม 204 นาทีและมีรางวัลสะสมเฉลี่ยเท่ากับ 230 สมองที่แปดใช้เวลาฝึกอบรม 234 นาทีและมีรางวัลสะสมเฉลี่ยเท่ากับ 194 สมองที่เก้าใช้เวลาฝึกอบรม 263 นาทีและมีรางวัลสะสมเฉลี่ยเท่ากับ 286 ซึ่งมีรางวัลสะสมเฉลี่ยสูงที่สุด และสุดท้ายสมองที่สิบใช้เวลาฝึกอบรม 292 นาทีและมีรางวัลสะสมเฉลี่ยเท่ากับ 106.5



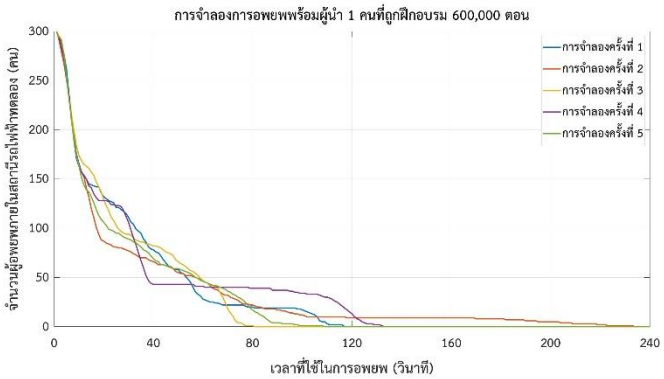
(ก)



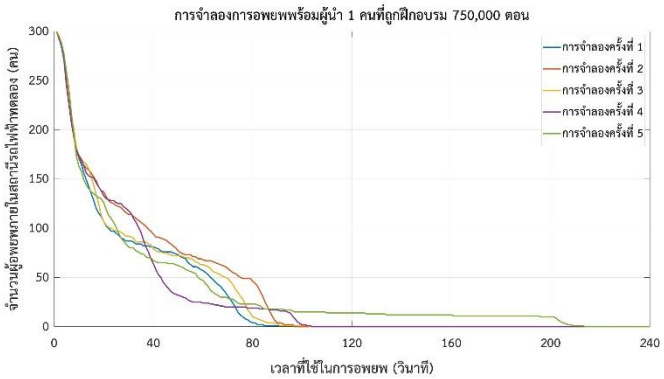
(ข)



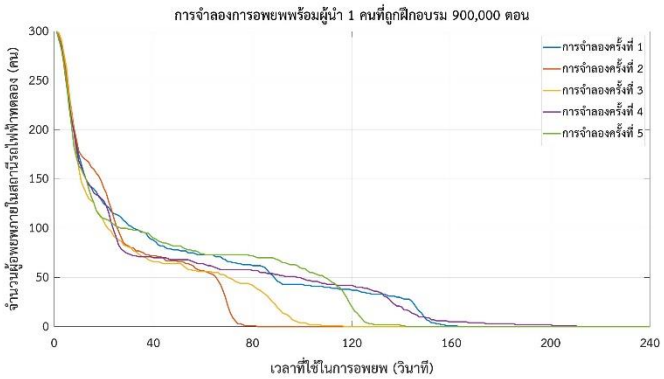
(ค)


การจำลองการอพยพพร้อมผู้นำ 1 คนที่ถูกฝึกอบรม 600,000 ตอน
จำนวนผู้อพยพภายในสถานีรถไฟฟ้าทดลอง (คน)
เวลาที่ใช้ในการอพยพ (วินาที)
การจำลองครั้งที่ 1
การจำลองครั้งที่ 2
การจำลองครั้งที่ 3
การจำลองครั้งที่ 4
การจำลองครั้งที่ 5

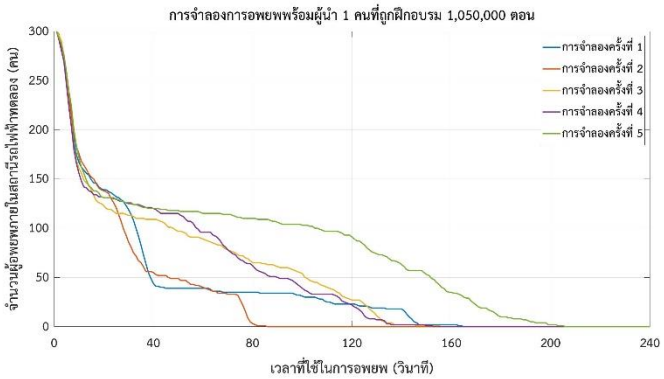
(ง)


การจำลองการอพยพพร้อมผู้นำ 1 คนที่ถูกฝึกอบรม 750,000 ตอน
จำนวนผู้อพยพภายในสถานีรถไฟฟ้าทดลอง (คน)
เวลาที่ใช้ในการอพยพ (วินาที)
การจำลองครั้งที่ 1
การจำลองครั้งที่ 2
การจำลองครั้งที่ 3
การจำลองครั้งที่ 4
การจำลองครั้งที่ 5

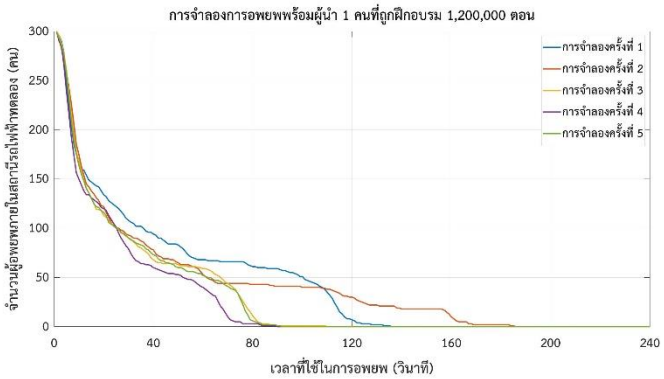
(จ)


การจำลองการอพยพพร้อมผู้นำ 1 คนที่ถูกฝึกอบรม 900,000 ตอน
จำนวนผู้อพยพภายในสถานีรถไฟฟ้าทดลอง (คน)
เวลาที่ใช้ในการอพยพ (วินาที)
การจำลองครั้งที่ 1
การจำลองครั้งที่ 2
การจำลองครั้งที่ 3
การจำลองครั้งที่ 4
การจำลองครั้งที่ 5

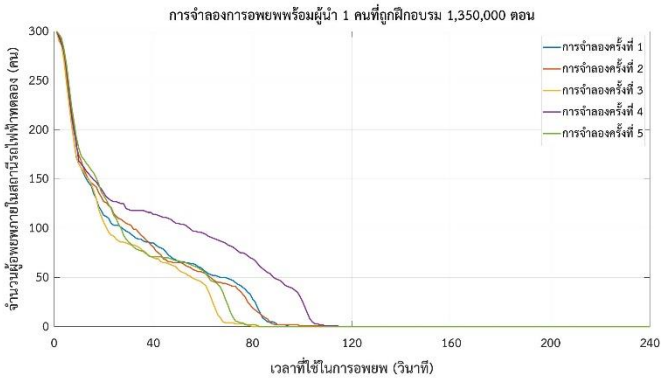
(ฉ)


การจำลองการอพยพพร้อมผู้นำ 1 คนที่ถูกฝึกอบรม 1,050,000 ตอน
จำนวนผู้อพยพภายในสถานีรถไฟฟ้าทดลอง (คน)
เวลาที่ใช้ในการอพยพ (วินาที)
การจำลองครั้งที่ 1
การจำลองครั้งที่ 2
การจำลองครั้งที่ 3
การจำลองครั้งที่ 4
การจำลองครั้งที่ 5

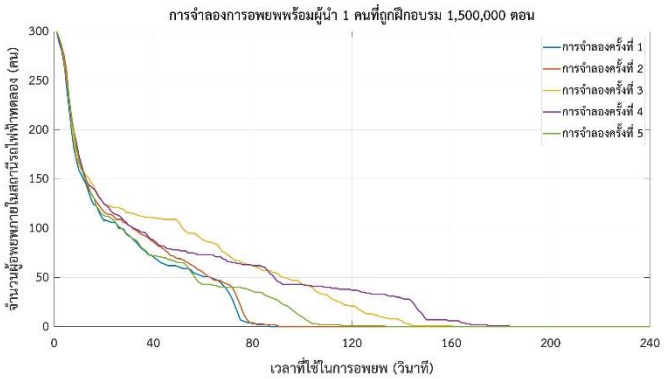
(ช)



(ช)



(ฌ)

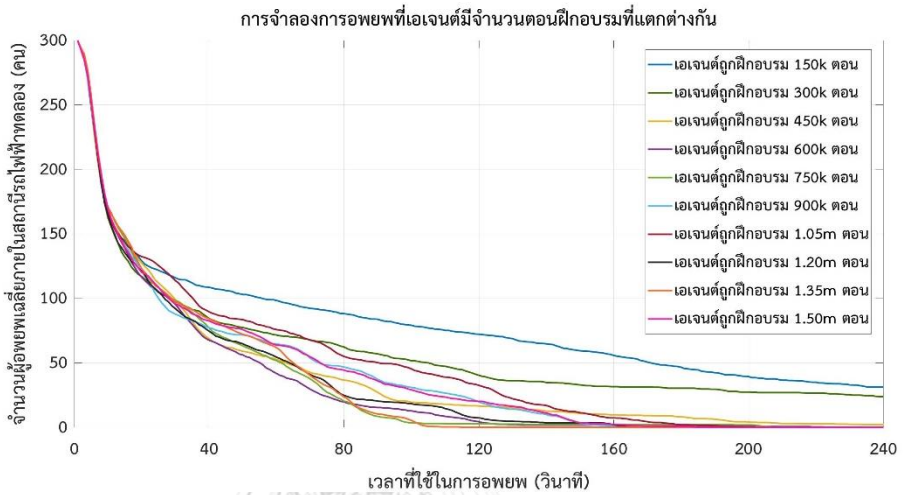


(ญ)

รูปที่ 6-7 ผลการทดสอบการอพยพฝูงชนด้วยผู้นำ 1 คนที่ถูกฝึกอบรมด้วยจำนวนตอนที่แตกต่างกัน

จากรูปที่ 6-7 แสดงผลการทดสอบการอพยพฝูงชนด้วยผู้นำ 1 คนที่ถูกฝึกอบรมด้วยจำนวนตอนที่แตกต่างกันภายในระยะเวลา 240 วินาที ซึ่งแต่ละครั้งจะมีการจำลองอย่างละ 5 ครั้ง ในรูปที่ 6-7 (ก) แสดงผลการอพยพฝูงชนด้วยผู้นำ 1 คนที่ถูกฝึกอบรมด้วยการเรียนรู้แบบเสริมกำลัง 150,000 ตอน จะเห็นว่าไม่มีสักครั้งที่การอพยพฝูงชนสามารถอพยพสำเร็จในระยะเวลาที่กำหนด รูปที่ 6-7 (ข) แสดงผลการอพยพฝูงชนด้วยผู้นำ 1 คนที่ถูกฝึกอบรมด้วยการเรียนรู้แบบเสริมกำลัง 300,000 ตอน จะเห็นว่าสามารถอพยพฝูงชนสำเร็จในระยะเวลาที่กำหนดเพียง 2 ครั้ง รูปที่ 6-7 (ค) แสดงผลการ

อพยพฝูงชนด้วยผู้นำ 1 คนที่ถูกฝึกอบรมด้วยการเรียนรู้แบบเสริมกำลัง 450,000 ตอน จะเห็นว่าสามารถอพยพฝูงชนสำเร็จในระยะเวลาที่กำหนดทั้งสิ้น 3 ครั้ง ส่วนรูปที่ 6-7 (ง) ถึง รูปที่ 6-7 (ญ) เป็นการจำลองการอพยพฝูงชนด้วยผู้นำ 1 คนที่ถูกฝึกอบรมด้วยการเรียนรู้แบบเสริมกำลัง 750,000 ตอน 900,000 ตอน 1,050,000 ตอน 1,200,000 ตอน 1,350,000 ตอน และ 1,500,000 ตอน ตามลำดับ ซึ่งผลแสดงให้เห็นว่าสามารถอพยพฝูงชนสำเร็จในระยะเวลาที่กำหนดทั้งหมด



รูปที่ 6-8 เปรียบเทียบผลการทดสอบการอพยพฝูงชนด้วยผู้นำที่ถูกฝึกอบรมด้วยจำนวนตอนที่แตกต่างกัน

การประเมินประสิทธิภาพของการอพยพในการทดสอบนี้จะวัดจากจำนวนผู้อพยพเฉลี่ยภายในสถานีรถไฟฟ้าทดลองหลังจากระยะเวลาการอพยพ 240 วินาที โดยจะเปรียบเทียบกับการอพยพฝูงชนด้วยผู้นำ 1 คนที่ถูกฝึกอบรมด้วยการเรียนรู้แบบเสริมกำลังด้วยจำนวนตอนการฝึกอบรมที่แตกต่างกันดังแสดงในรูปที่ 6-8 จะเห็นได้ว่าการจำลองการอพยพฝูงชนด้วยผู้นำ 1 คนที่ถูกฝึกอบรมด้วยการเรียนรู้แบบเสริมกำลัง 150,000 ตอน 300,000 ตอน 450,000 ตอน ไม่สามารถอพยพฝูงชนได้สำเร็จภายในระยะเวลาที่กำหนด ซึ่งมีจำนวนผู้อพยพติดค้างเฉลี่ยภายในสถานีรถไฟฟ้าทดลองเท่ากับ 31.4 คน 23.6 คนและ 2.2 คนตามลำดับ ในส่วนการจำลองการอพยพฝูงชนด้วยผู้นำ 1 คนที่ถูกฝึกอบรมด้วยการเรียนรู้แบบเสริมกำลังที่เหลือสามารถอพยพฝูงชนได้สำเร็จภายในระยะเวลาที่กำหนด ซึ่งการจำลองการอพยพฝูงชนด้วยผู้นำ 1 คนที่ถูกฝึกอบรมด้วยการเรียนรู้แบบเสริมกำลัง 1,350,000 ตอน สามารถอพยพฝูงชนออกจากสถานีรถไฟฟ้าทดลองได้รวดเร็วที่สุด โดยใช้เวลำหรับการอพยพฝูงชนเพียง 114 วินาที

ตารางที่ 6-1 ตารางแสดงผลการจำลองการอพยพคนภายในสถานีรถไฟฟ้าทดลองจำนวนตอนที่ฝึกอบรมเอเจนต์ที่แตกต่างกัน

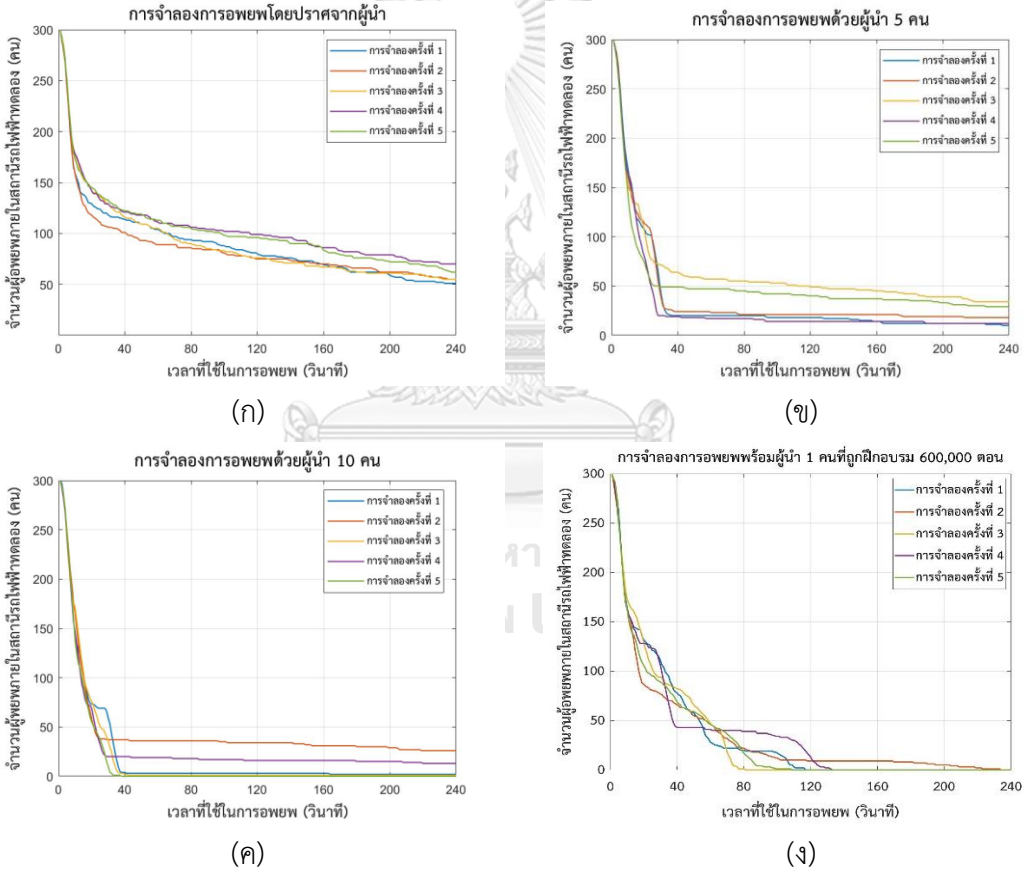
จำนวนตอนที่ฝึกอบรมเอเจนต์ (ตอน)	เวลาการฝึกอบรมที่ผ่านไป (นาที)	เวลาจริงที่ผ่านไปสำหรับฝึกอบรม (นาที)	รางวัลสะสมเฉลี่ย	จำนวนคนที่ติดค้างในสถานีรถไฟฟ้าทดลอง (คน)					
				40	80	120	160	200	240
150000	30.73	614.6	37	108	87.8	72.2	55.2	38.6	31.4
300000	59.27	1185.4	131.33	83.2	61	40.4	31.6	27.2	23.6
450000	89.39	1787.8	238	66.8	36.4	16.6	9.8	4	2.2
600000	118.58	2371.6	250.5	66.6	18.8	4.4	1.8	1	0
750000	147.25	2945	200	74.4	19.8	2.8	2.2	2	0
900000	175.9	3518	224.5	76.6	46.2	19.8	1.2	0.2	0
1050000	204.67	4093.4	230	88.4	54.2	32.6	7.2	0.4	0
1200000	234.19	4683.8	194	72.8	24	7.4	1.6	0	0
1350000	263.78	5275.6	286	82.8	22.2	0	0	0	0

จากตารางที่ 6-1 แสดงประสิทธิภาพของการอพยพฝูงชนออกจากสถานีรถไฟฟ้าทดลองด้วยผู้นำ 1 คนที่ถูกฝึกอบรมด้วยการเรียนรู้แบบเสริมกำลังด้วยจำนวนตอนการฝึกอบรมที่แตกต่างกัน ซึ่งถูกประเมินจากจำนวนคนเฉลี่ยที่ติดค้างในสถานีรถไฟฟ้าทดลองในแต่ละช่วงเวลา โดยการอพยพฝูงชนด้วยผู้นำ 1 คนที่ถูกฝึกอบรม 1,350,000 ตอนมีรางวัลสะสมเฉลี่ยสูงสุดอยู่ที่ 286 คะแนนและใช้เวลาสำหรับการอพยพฝูงชนทั้งหมดไม่เกิน 120 วินาทีซึ่งเร็วที่สุด นอกจากนี้ยังแสดงเวลาที่ผ่านไปสำหรับฝึกอบรมในการจำลอง และเวลาจริงสำหรับฝึกอบรมเนื่องจากในการจำลองมีการเร่งเวลาดังนั้นเวลาจริงสำหรับฝึกอบรมจะเป็นยี่สิบเท่าของเวลาที่ผ่านไปสำหรับฝึกอบรมในการจำลอง ซึ่งการอพยพฝูงชนด้วยผู้นำ 1 คนที่ถูกฝึกอบรม 1,350,000 ตอนมีประสิทธิภาพมากที่สุด แต่ใช้เวลาจริงสำหรับฝึกอบรมนานถึง 5275.6 นาที หรือประมาณ 3 ถึง 4 วัน

6.3 ผลการทดสอบของโปรแกรมการอพยพฝูงชนบนโลกสามมิติ

ในการทดสอบนี้ วิธีการการอพยพฝูงชนด้วยวิธีการที่แตกต่างกันจะถูกนำมาใช้ในการทดลองเพื่อเปรียบเทียบกับวิธีที่นำเสนอจะมีอยู่ด้วยกัน 4 วิธีคือ การอพยพโดยปราศจากผู้นำอพยพ โดยวิธีการที่หนึ่งผู้อพยพจะทำการอพยพออกจากสถานีรถไฟฟ้าทดลองด้วยตัวเอง ส่วนวิธีการที่สองและ

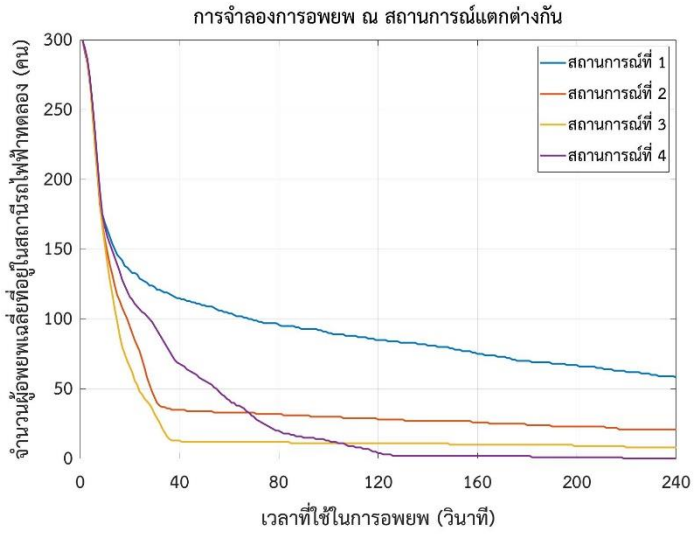
วิธีการที่สามเป็นการอพยพพร้อมผู้นำ 5 คน และผู้นำ 10 คน ตามลำดับ ซึ่งตำแหน่งของผู้นำอพยพจะถูกสุ่มอย่างกระจายตัวภายในสถานีรถไฟฟ้าทดลอง โดยผู้นำอพยพจะเดินไปยังทางออกที่ใกล้ตัวเองที่สุดพร้อมพาผู้อพยพที่ไม่สามารถอพยพได้ด้วยตัวเองและผู้อพยพจำเป็นต้องเห็นผู้นำอพยพในระยะสายตาจึงจะสามารถออกจากสถานีรถไฟฟ้าทดลองได้สำเร็จ ส่วนวิธีการที่สี่หรือวิธีการที่ผู้วิจัยได้เสนอเป็นการอพยพด้วยเอเจนต์เพียงหนึ่งเดียวที่ถูกฝึกด้วยการเรียนรู้แบบเสริมกำลังที่มีชื่อเทคนิคว่าการเพิ่มประสิทธิภาพนโยบายใกล้เคียงเพื่อให้เป็นผู้นำอพยพ โดยเอเจนต์จะพยายามเดินไปหาผู้อพยพทุกคนที่ไม่สามารถอพยพออกจากสถานีรถไฟฟ้าทดลองได้ด้วยตัวเองเพื่อบอกตำแหน่งของทางออกที่ใกล้ที่สุดและเส้นทางการอพยพแก่ผู้อพยพ หลังจากผู้อพยพได้รับข้อมูลจากผู้นำอพยพเรียบร้อยแล้วผู้อพยพจะสามารถออกจากสถานีรถไฟฟ้าทดลองได้ด้วยตัวเอง



รูปที่ 6-9 ผลการทดสอบการอพยพฝูงชนของแต่ละสถานการณ์

ในการประเมินประสิทธิภาพของการอพยพในการทดสอบนี้จะวัดจากจำนวนผู้อพยพเฉลี่ยที่ไม่สามารถออกจากสถานีรถไฟฟ้าทดลองตามระยะเวลาการอพยพที่กำหนด ที่เปรียบเทียบกับการอพยพในสถานการณ์ต่าง ๆ รูปที่ 6-9 (ก) แสดงการเปรียบเทียบประสิทธิภาพของวิธีการอพยพฝูงชนโดยปราศจากผู้นำที่ถูกจำลองเพื่อประเมินทั้งสิ้น 5 ครั้ง จะเห็นว่ายังมีผู้อพยพจำนวน 50 คนขึ้นไปที่

ไม่สามารถอพยพออกจากสถานีรถไฟฟ้าทดลองได้ ในส่วนของรูปที่ 6-9 (ข) และรูปที่ 6-9 (ค) แสดงการเปรียบเทียบประสิทธิภาพของวิธีการอพยพฝูงชนด้วยผู้นำจำนวน 5 คนและ 10 คนตามลำดับ ซึ่งที่ถูกจำลองเพื่อประเมินทั้งสิ้น 5 ครั้ง จะเห็นว่ายังมีผู้อพยพจำนวน 10 คนขึ้นไปที่ไม่สามารถอพยพออกจากสถานีรถไฟฟ้าทดลองได้ ในส่วนของการอพยพด้วยผู้นำจำนวน 10 คนสามารถอพยพคนออกจากสถานีรถไฟฟ้าทดลองได้หมดเพียงหนึ่งครั้ง ซึ่งก็คือการจำลองครั้งที่ 5 สำหรับรูปที่ 6-9 (ง) แสดงการเปรียบเทียบประสิทธิภาพของวิธีการอพยพฝูงชนด้วยเอเจนต์ 1 คนที่ถูกฝึกอบรม 600,000 ตอนให้เป็นผู้นำการอพยพด้วยการเรียนรู้แบบเสริมกำลัง ซึ่งเป็นวิธีการที่ผู้วิจัยได้นำเสนอ จะเห็นได้ว่าวิธีที่นำเสนอจะสามารถอพยพคนออกจากสถานีรถไฟฟ้าทดลองได้ทั้งหมดแต่ระยะเวลาอาจจะมีความช้าหรือเร็วแตกต่างกัน



รูปที่ 6-10 เปรียบเทียบผลการทดสอบการอพยพฝูงชนด้วยสถานการณ์ที่แตกต่างกัน

จากรูปที่ 6-10 แสดงการเปรียบเทียบผลการทดสอบการอพยพฝูงชนในสถานการณ์ต่าง ๆ จะถูกแสดงผลออกมาในรูปแบบของจำนวนผู้อพยพเฉลี่ยที่ไม่สามารถออกจากสถานีรถไฟฟ้าทดลองภายในระยะเวลา 240 วินาที จะเห็นได้ว่าในช่วง 10 วินาทีแรกของการอพยพฝูงชนแต่ละสถานการณ์จำนวนผู้อพยพเฉลี่ยที่ติดอยู่ในสถานีรถไฟฟ้าทดลองจะลดลงอย่างรวดเร็วเหมือนกันซึ่งเป็นผลมาจากผู้อพยพสามารถมองเห็นทางออกและสามารถเดินไปยังทางออกได้ด้วยตนเอง ในส่วนของการอพยพฝูงด้วยผู้นำจำนวนผู้อพยพเฉลี่ยที่ติดอยู่ในสถานีรถไฟฟ้าทดลองจะลดลงอย่างรวดเร็วต่อจนถึงประมาณ 40 วินาที หลังจากนั้นจำนวนผู้อพยพจะค่อย ๆ ลดลงแต่ก็ไม่สามารถอพยพออกจากสถานีรถไฟฟ้าทดลองได้ทั้งหมด เนื่องจากผู้นำอพยพได้ออกไปหมดแล้ว โดยสถานการณ์แรกซึ่งเป็นการอพยพโดยปราศจากผู้นำมีผู้อพยพเฉลี่ยที่ติดค้างภายในสถานีรถไฟฟ้าทดลองเท่ากับ 58 คน

สถานการณ์ที่สองและสามเป็นการอพยพพร้อมผู้นำที่ถูกสุ่มตำแหน่ง 5 คนและ 10 คน ซึ่งมีผู้อพยพเฉลี่ยที่ติดค้างภายในสถานีรถไฟฟ้าทดลองเท่ากับ 21 คนและ 8 คนตามลำดับ ในส่วนสถานการณ์ที่สี่หรือการอพยพฝูงชนด้วยเอเจนต์หรือผู้นำ 1 คนที่ถูกฝึกด้วยการเรียนรู้แบบเสริมกำลัง ซึ่งหลังจากผ่าน 10 วินาทีแรกจำนวนผู้อพยพเฉลี่ยที่ติดอยู่ในสถานีรถไฟฟ้าทดลองจะค่อย ๆ ลดลงอย่างต่อเนื่องจนไม่หลงเหลือผู้อพยพอยู่ภายในสถานีรถไฟฟ้าทดลองหรือมีค่าเป็นศูนย์ ซึ่งการจำลองการอพยพสถานการณ์ที่สี่นี้สามารถอพยพคนได้ทั้งหมด 300 คนภายในระยะเวลา 218 วินาที

ตารางที่ 6-2 ตารางแสดงผลการจำลองการอพยพคนภายในสถานีรถไฟฟ้าทดลอง

	จำนวนคนที่ติดค้างในสถานีรถไฟฟ้าทดลอง (คน)					
เวลาที่ใช้ในการอพยพ (วินาที)	40	80	120	160	200	240
ไม่มีผู้นำ	186	205	215	225	66	242
ผู้นำ 5 คน	265	268	272	274	277	279
ผู้นำ 10 คน	288	288	289	290	291	292
วิธีการที่นำเสนอ	204	243	272	288	297	300

ตารางที่ 6-2 แสดงผลการทดลองของการจำลองการอพยพกรณีการอพยพฝูงชนโดยปราศจากผู้นำ การอพยพฝูงชนพร้อมผู้นำ 5 คนและ 10 คนซึ่งไม่ได้ถูกฝึกฝนด้วยการเรียนรู้แบบเสริมกำลัง และการอพยพฝูงชนพร้อมผู้นำ 1 คนที่ถูกฝึกด้วยการเรียนรู้แบบเสริมกำลัง โดยมีจำนวนผู้อพยพทั้งสิ้น 300 คน ในช่วง 40 วินาทีแรกการอพยพฝูงชนโดยปราศจากผู้นำมีผู้อพยพที่อพยพสำเร็จถึง 186 คนแล้วในช่วงเวลาต่อมาจำนวนผู้อพยพสำเร็จจะค่อย ๆ เพิ่มขึ้นแต่ไม่สามารถอพยพสำเร็จได้ทั้งหมดภายในระยะเวลาที่กำหนด ส่วนการอพยพพร้อมผู้นำ 5 หรือ 10 คน ผู้นำอพยพจะพาผู้อพยพที่พบเห็นผู้นำในระยะสายตาของตัวเองออกไปพร้อมกัน กล่าวคือ ผู้อพยพจะเดินตามผู้นำออกไปจนอพยพสำเร็จ โดยในช่วง 40 วินาทีแรกมีผู้อพยพสำเร็จสูงถึง 265 และ 288 คนตามลำดับ แต่จำนวนผู้อพยพสำเร็จ ณ เวลาที่ 80 120 160 200 และ 240 วินาที นั้นกับมีเพิ่มขึ้นทีละไม่เกิน 5 คน ซึ่งแสดงให้เห็นว่าผู้นำอพยพทั้งหมดได้ออกไปพร้อมกับผู้อพยพตั้งแต่ 40 วินาทีแรกแล้ว โดยยังมีผู้อพยพที่อพยพไม่สำเร็จอยู่ภายในสถานีรถไฟฟ้าทดลอง ในส่วนของการอพยพพร้อมผู้นำที่ถูกฝึกด้วยวิธีการที่นำเสนอ มีผู้อพยพสำเร็จ 233 คน ในช่วง 40 วินาที และจำนวนผู้อพยพสำเร็จก็เพิ่มขึ้นอย่างต่อเนื่อง ณ เวลาต่อมา จนกระทั่งผู้อพยพสำเร็จมีจำนวนทั้งสิ้น 300 คนภายในระยะเวลา 240 วินาที การอพยพกรณีนี้ผู้นำอพยพจะเดินไปบอกทางออกที่ดีแก่ผู้อพยพที่ไม่สามารถอพยพออกจากสถานีรถไฟฟ้าทดลองได้ด้วยตนเอง ซึ่งส่งผลให้ผู้อพยพสามารถอพยพออกจากสถานีได้ด้วยตัวเองโดย

ปราศจากผู้นำ ส่วนผู้นำก็จะเดินไปบอกทางที่ดีสุดแก่ผู้อพยพคนต่อ ๆ ไปจนสามารถอพยพได้สำเร็จทั้งหมด

จากการเปรียบเทียบผลการทดลองจำลองการอพยพคนภายในสถานีรถไฟฟ้าทดลองในตารางที่ 6-2 จะเห็นได้ชัดว่าจำนวนผู้อพยพที่อพยพสำเร็จกรณีมีผู้นำมีมากกว่าการอพยพที่ไม่มีผู้นำ และการอพยพกรณีมีผู้นำที่ถูกฝึกด้วยวิธีการที่นำเสนอเป็นการอพยพกรณีเดียวที่จำนวนผู้อพยพทั้งหมดสามารถอพยพสำเร็จภายในเวลาอพยพเดียวกัน

บทที่ 7
บทสรุปและแนวทางในการพัฒนาต่อในอนาคต

ในบทนี้จะกล่าวถึงบทสรุปของงานวิทยานิพนธ์ฉบับนี้ โดยวิทยานิพนธ์นี้มีการสร้างโปรแกรมแบบจำลองสองและสามมิติเพื่อเปรียบเทียบประสิทธิภาพการอพยพฝูงชนภายในสถานีรถไฟฟ้าทดลอง ซึ่งโปรแกรมสองมิติจะใช้อัลกอริธึมเอสตาร์เพียงอย่างเดียวในการแก้ปัญหาการค้นหาเส้นทางอพยพแต่โปรแกรมสามมิติจะใช้อัลกอริธึมเอสตาร์กับตาข่ายนำทางหรือนาฟเมชในการแก้ปัญหาการค้นหาเส้นทางอพยพที่สั้นที่สุด นอกจากนี้มีการประยุกต์ใช้การเรียนรู้แบบเสริมกำลังกับกลยุทธ์การอพยพฝูงชนพร้อมผู้นำ เพื่อลดจำนวนคนหลงเหลือภายในสถานีรถไฟฟ้าเพื่อมาเปรียบเทียบกับวิธีการอพยพแบบเดิมบนโปรแกรมสามมิติที่พัฒนาด้วยโปรแกรมยูนิตี โดยใช้แบบจำลองแรงทางสังคมจำลองจำลองสถานการณ์การอพยพ จากผลการทดลองที่แสดงในบทก่อนหน้านี้จะเห็นได้ว่าวิธีการที่นำเสนอสามารถลดจำนวนคงเหลือภายในสถานีลงได้จริง และยังแสดงเห็นว่าความแตกต่างระหว่างเส้นทางอพยพของโปรแกรมแบบจำลองสองและสามมิตินั้นส่งผลต่อประสิทธิภาพการอพยพฝูงชน จากทั้งหมดที่ได้กล่าวมานี้สรุปได้ว่าวิทยานิพนธ์ฉบับนี้สามารถประยุกต์ใช้องค์ความรู้ที่พัฒนาขึ้นในการสนับสนุนการอพยพฝูงชนให้รวดเร็วและเกิดประสิทธิผลสูง ประชาชนผู้ใช้บริการรถไฟฟ้าจะได้รับบริการขนส่งมวลชนที่มีความปลอดภัยในชีวิตมากขึ้น นอกจากนี้โปรแกรมจำลองการอพยพสองมิติและสามมิติในวิทยานิพนธ์ฉบบนี้เป็นแบบโอเพนซอร์ส ดังนั้นผู้ที่สนใจสามารถติดต่อและนำไปโปรแกรมไปศึกษาทดลองและปรับปรุงเพื่อเพิ่มประสิทธิภาพให้มากยิ่งขึ้นได้

7.1 บทสรุป

ในงานวิทยานิพนธ์นี้ได้ออกแบบและสร้างแบบจำลองการอพยพฝูงชนบนระนาบสองมิติด้วยโปรแกรมไพทอนเองทั้งหมดและแบบจำลองการอพยพฝูงชนบนโลกสามมิติด้วยโปรแกรมยูนิตี รวมถึงเสนอวิธีการอพยพฝูงชนภายในสถานีรถไฟฟ้าทดลองด้วยผู้นำที่ถูกด้วยฝึกฝนด้วยการเรียนรู้แบบเสริมกำลังที่มีเทคนิคที่ชื่อว่า การเพิ่มประสิทธิภาพนโยบายใกล้เคียง เพื่อที่เอเจนต์จะกลายเป็นผู้นำอพยพและเรียนรู้ที่จะเดินไปหาผู้ติดตามทั้งหมดแล้วบอกตำแหน่งของทางออกและเส้นทางอพยพที่สั้นที่สุดไปยังทางออกนั้น ซึ่งแตกต่างจากผู้นำอพยพแบบเดิมที่จะเดินไปยังทางออกแล้วพาผู้อพยพที่เห็นผู้นำอพยพในระยะสายตาออกไปด้วย กล่าวคือผู้อพยพจะเดินตามผู้นำอพยพไปจนกว่าจะทำการอพยพสำเร็จ โดยการเคลื่อนไหวของผู้อพยพและผู้นำอพยพในการจำลองสถานการณ์การอพยพจะถูกขับเคลื่อนด้วยแบบจำลองแรงทางสังคมซึ่งประกอบไปด้วยแรงที่ต้องการ แรงปฏิสัมพันธ์ระหว่างคนเดินเท้ากับคนเดินเท้า แรงปฏิสัมพันธ์ระหว่างคนเดินเท้ากับกำแพง และแรงนำทางสำหรับผู้อพยพที่ไม่สามารถหาทางออกได้ด้วยตนเองหรือที่เรียกว่า ผู้ติดตาม เท่านั้นเพื่อใช้สำหรับการสร้างแบบจำลอง

กลุ่มคนเดินเท้าแบบมีผู้นำ ในแบบจำลองสามารถกำหนดตัวแปรต่าง ๆ ได้แก่ จำนวนผู้อพยพ จำนวนผู้นำอพยพ ตำแหน่งของทางออกและสิ่งกีดขวาง ตลอดจนถึงความเร็วในการเดินและเวลาการอพยพ ซึ่งอ้างอิงจากมาตรฐานการออกแบบเพื่อป้องกันอัคคีภัยที่เรียกว่า มาตรฐานความปลอดภัย NFPA 130 ส่วนของเส้นทางการอพยพของแบบจำลองการอพยพฝูงชนบนระนาบสองมิตินั้นหาจากการค้นหาแบบเอสตาร์ ซึ่งเป็นการหาเส้นทางที่มีระยะทางน้อยที่สุดจากกริดเริ่มต้นไปยังกริดเป้าหมายซึ่งอาจจะมีได้หลายกริด ในส่วนของแบบจำลองการอพยพฝูงชนบนโลกสามมิติจะใช้ตาข่ายนำทาง (NavMesh) ซึ่งเป็นอุปกรณ์เสริมของโปรแกรมยูนิตีและอัลกอริธึมเอสตาร์ทำงานร่วมกันในการแก้ปัญหาการค้นหาเส้นทางอพยพที่สั้นที่สุด

ตารางที่ 7-1 ตารางเปรียบเทียบการทดลองที่ทำการศึกษาบนโปรแกรมสองมิติและสามมิติ

การศึกษาแบบจำลองการอพยพ	โปรแกรม		จำนวนผูอพยพ		สถานการณ์การอพยพ			
	สองมิติ	สามมิติ	50 คน	300 คน	ไม่มีผู้นำ	ผู้นำ 5 คน	ผู้นำ 10 คน	ผู้นำ 1 คน
1	✓	✓	✓		✓	✓		
2		✓		✓				✓
3		✓		✓	✓	✓	✓	✓
หมายเหตุ	การอพยพพร้อมผู้นำ 1 คนเป็นวิธีการเดียวที่นำการเรียนรู้แบบเสริมกำลังมาประยุกต์ใช้							

จากตารางที่ 7-1 แสดงการทดลองที่ทำการศึกษา โดยการศึกษาแรกเป็นการทดสอบบนโปรแกรมจำลองการอพยพสองมิติบนไพทอนกับโปรแกรมจำลองการอพยพสามมิติบนยูนิตี เพื่อเปรียบเทียบประสิทธิภาพของโปรแกรมจำลองการอพยพสองมิติที่สร้างขึ้นเองทั้งหมดกับโปรแกรมจำลองการอพยพสามมิติ ซึ่งมีข้อกำหนดของการทดลองคือ จำนวนผู้อพยพเท่ากับ 50 คนและมีสถานการณ์การอพยพ 2 กรณีคือ การอพยพโดยปราศจากผู้นำและการอพยพพร้อมผู้นำ 5 คน โดยการศึกษาแรกนี้จะไม่มีการนำการเรียนรู้แบบเสริมกำลังมาประยุกต์ใช้ ส่วนการศึกษาที่สองเป็นการทดสอบบนโปรแกรมจำลองการอพยพสามมิติบนยูนิตีและนำการเรียนรู้แบบเสริมกำลังมาประยุกต์ใช้การจำลองการอพยพ โดยการฝึกอบรมเอเจนต์ให้กลายเป็นผู้นำอพยพเพื่อเปรียบเทียบประสิทธิภาพของการอพยพของผู้นำที่ถูกฝึกด้วยระยะเวลาการฝึกอบรมที่แตกต่างกัน ซึ่งมีข้อกำหนดของการทดลองคือ จำนวนผู้อพยพเท่ากับ 300 คนและสถานการณ์การอพยพ 1 กรณีคือ การอพยพพร้อมผู้นำ 1 คนที่ถูกฝึกด้วยการเรียนรู้แบบเสริมกำลัง ในส่วนการศึกษาที่สามเป็นการทดสอบบนโปรแกรมจำลองการอพยพสามมิติบนยูนิตี โดยทำการเปรียบเทียบประสิทธิภาพของการอพยพด้วยวิธีแบบเดิมกับการอพยพพร้อมผู้นำ 1 คนที่ดีที่สุดจากการศึกษาที่สอง ซึ่งมีข้อกำหนดของการทดลองคือ จำนวนผู้อพยพ

เท่ากับ 300 คนและสถานการณ์การอพยพ 4 กรณี ได้แก่ การอพยพโดยปราศจากผู้นำ การอพยพพร้อมผู้นำ 5 คน การอพยพพร้อมผู้นำ 10 คน และการอพยพพร้อมผู้นำ 1 คนที่ถูกฝึกด้วยการเรียนรู้แบบเสริมกำลัง

ตารางที่ 7-2 ตารางเปรียบเทียบผลการทดลองทั้งหมดที่ทำการศึกษา

การศึกษาแบบจำลองการอพยพ	โปรแกรม	การค้นหาเส้นทาง	จำนวนผู้อพยพ (คน)	จำนวนผู้อพยพที่ติดค้างในกรณีศึกษา			
				ไม่มีผู้นำ	ผู้นำ 5 คน	ผู้นำ 10 คน	ผู้นำ 1 คน
1	Python (2x2)	อัลกอริทึมเอสตาร์	50	21	15	-	-
	Python (4x4)	อัลกอริทึมเอสตาร์	50	20	15	-	-
	Unity	อัลกอริทึมเอสตาร์กับตาข่ายนำทาง	50	9	4	-	-
2	Unity	อัลกอริทึมเอสตาร์กับตาข่ายนำทาง	300	-	-	-	0
3	Unity	อัลกอริทึมเอสตาร์กับตาข่ายนำทาง	300	58	21	8	0
หมายเหตุ*	เวลาที่ใช้ในการอพยพเท่ากับ 240 วินาที						
หมายเหตุ**	การอพยพพร้อมผู้นำ 1 คนที่ถูกฝึกอบรม 600,000 ตอน						

จากตารางที่ 7-2 การประเมินผลการศึกษาแรกที่เป็นการเปรียบเทียบประสิทธิภาพการอพยพฝูงชนออกจากสถานีรถไฟฟ้าทดลองของโปรแกรมแบบจำลองสองและสามมิตินั้นพบว่า ในเวลาการอพยพที่เท่ากันโปรแกรมแบบจำลองสามมิติมีประสิทธิภาพการอพยพที่ดีกว่าโปรแกรมแบบจำลองสองมิติ โดยดูจากจำนวนคนติดค้างภายในสถานีมีน้อยกว่า ซึ่งเป็นผลมาจากการค้นหาเส้นทางการอพยพที่แตกต่างกันและด้วยโปรแกรมยูนิตีมีประสิทธิภาพในการแสดงผลและการจำลองการอพยพที่ดีกว่าโปรแกรมไพทอน ส่วนการประเมินผลการศึกษาที่สองเป็นการเปรียบเทียบการอพยพด้วยวิธีแบบเดิมกับการอพยพพร้อมผู้นำ 1 คนที่ถูกฝึกอบรมด้วยจำนวนตอนที่แตกต่างกันพบว่า การอพยพพร้อมผู้นำ 1 คนที่ถูกฝึกอบรม 1,350,000 ตอนมีประสิทธิภาพการอพยพดีที่สุด เนื่องจากใช้เวลาการอพยพเพียง 120 วินาทีแต่ใช้เวลาจริงนานประมาณ 4 วันในการฝึกอบรม แต่การอพยพพร้อมผู้นำ 1 คนที่ถูกฝึกอบรม 600,000 ตอนก็สามารถอพยพฝูงชนทั้งหมดภายในระยะเวลา 240 นาที ซึ่งไม่เร็ว

เท่าการอพยพพร้อมผู้นำ 1 คนที่ถูกฝึกอบรม 1,350,000 ตอนแต่ใช้เวลาจริงในการฝึกอบรม 1 วันครึ่งซึ่งน้อยกว่า ในส่วนการประเมินผลการศึกษาที่สามเป็นการเปรียบเทียบการอพยพด้วยวิธีแบบเดิมกับการอพยพพร้อมผู้นำ 1 คนที่ถูกฝึกอบรม 600,000 เนื่องจากสามารถอพยพฝูงชนทั้งหมดได้ภายในระยะเวลาที่กำหนดและใช้เวลาจริงในการฝึกอบรมไม่นานมากนัก ซึ่งพบว่าเป็นวิธีเดียวที่สามารถช่วยเหลือผู้ติดตามทั้งหมดได้ภายในระยะเวลาที่กำหนด เนื่องจากเอเจนต์จะเดินไปหาผู้อพยพทั้งหมดที่ไม่สามารถอพยพออกจากสถานีรถไฟฟ้าทดลองได้ด้วยตนเองแล้วบอกเส้นทางการอพยพที่สั้นที่สุดที่หาจากตาข่ายนำทางกับอัลกอริธึมเอสตาร์ ซึ่งหลังจากที่ผู้อพยพได้รับเส้นทางการอพยพแล้วผู้อพยพจะเดินไปยังทางออกด้วยตัวเอง ส่วนเอเจนต์ก็จะเดินไปหาผู้อพยพถัดไป ซึ่งแตกต่างไปจากวิธีการอพยพของผู้นำอพยพในงานวิจัยก่อนหน้า นอกจากนี้เอเจนต์ที่ถูกฝึกมานี้สามารถใช้ได้กับสภาพแวดล้อมนี้และข้อกำหนดของแบบจำลองนี้เท่านั้น ถ้าต้องการใช้กับสภาพแวดล้อมอื่นหรือมีข้อกำหนดของการอพยพที่ซับซ้อนมากขึ้นจำเป็นต้องฝึกฝนเอเจนต์ใหม่

7.2 ข้อจำกัดของงานในวิทยานิพนธ์

ในงานวิทยานิพนธ์นี้มีข้อจำกัดของตัวโปรแกรมที่เป็นสองมิติและสามมิติ ดังนี้

1. โปรแกรมสองมิติมีข้อจำกัดคือไม่สามารถจำลองการอพยพกับกรณีศึกษาที่มีผู้อพยพจำนวนมากได้ อาจมีได้หลายสาเหตุ เช่น มีการคำนวณค่าแรงทางสังคมบ่อยครั้งเกินไปและมีการพิจารณาปฏิสัมพันธ์ระหว่างผู้อพยพจำนวนมากเกินจำเป็น ดังนั้น ยังมีความจำเป็นที่จะต้องออปติไมซ์โปรแกรมให้มีประสิทธิมากขึ้นเพื่อรองรับผู้อพยพที่สูงขึ้น

2. ในส่วนโปรแกรมสามมิติกรณีการจำลองการอพยพด้วยผู้นำ 1 คนที่ถูกฝึกด้วยเทคนิคการเรียนรู้แบบเสริมกำลังที่มีชื่อว่า การเพิ่มประสิทธิภาพนโยบายใกล้เคียง ที่ใช้ได้กับสภาพแวดล้อมของสถานีรถไฟฟ้าทดลองที่มีต้นแบบจากสถานีรถไฟฟ้าปักกิ่งเท่านั้นและข้อกำหนดของการทดลองนี้เท่านั้น ถ้าจะมีการเปลี่ยนแปลงส่งผลประสิทธิการอพยพด้วยวิธีการนี้ลดลงหรือไม่สามารถทำการอพยพได้เลย ดังนั้นจำเป็นต้องฝึกฝนเอเจนต์ผู้นำใหม่ถ้านำไปใช้กับสภาพแวดล้อมอื่น

3. จำนวนเอเจนต์มากกว่าหนึ่งมีความจำเป็นกับสภาพแวดล้อมที่มีขนาดใหญ่ขึ้น ในส่วนนี้ต้องมีการคิดค้นหาฟังก์ชันรางวัลใหม่เพื่อทำให้เอเจนต์เกิดการทำงานร่วมกันได้อย่างมีประสิทธิภาพมากยิ่งขึ้น

7.3 แนวทางการพัฒนาต่อในอนาคต

ในงานวิทยานิพนธ์นี้มีแนวทางที่ควรจะได้รับการพัฒนาต่อในอนาคต กล่าวคือ

1. งานวิจัยได้นำเสนอวิธีการอพยพฝูงชนพร้อมผู้นำในสถานีรถไฟฟ้าทดลองที่ได้ต้นแบบจากสถานีรถไฟฟ้าปักกิ่ง ถ้าหากต้องการนำเอเจนต์ที่ฝึกมานี้ไปใช้กับสถานีรถไฟฟ้าอื่น ๆ หรือมีข้อกำหนดของการอพยพซับซ้อนมากขึ้นจำเป็นต้องทำการฝึกอบรมเอเจนต์ในสภาพแวดล้อมหรือข้อกำหนดการอพยพใหม่

2. งานวิจัยได้นำเสนอวิธีการอพยพฝูงชนพร้อมผู้นำในสถานีรถไฟฟ้าทดลองที่ถูกฝึกอบรมด้วยเทคนิคการเพิ่มประสิทธิภาพนโยบายใกล้เคียง ซึ่งเป็นการอพยพฝูงชนด้วยเอเจนต์เพียงหนึ่งเดียว งานวิจัยในอนาคตอาจเป็นการศึกษาเพื่อหาแนวทางการอพยพฝูงชนด้วยเอเจนต์หลายตัว ซึ่งเอเจนต์จะถูกฝึกอบรมให้มีการทำงานร่วมกันเพื่อพัฒนาประสิทธิภาพการอพยพฝูงชนให้มากยิ่งขึ้น

ภาคผนวก

โปรแกรมการค้นหาแบบเอสตาร์

```
class Node():

    """A node class for A* Pathfinding"""
    def __init__(self, parent=None, position=None):
        self.parent = parent
        self.position = position
        self.g = 0
        self.h = 0
        self.f = 0
    def __eq__(self, other):
        return self.position == other.position
def astar(maze, start, end):
    # Create start and end node
    start_node = Node(None, start)
    start_node.g = start_node.h = start_node.f = 0
    end_node = Node(None, end)
    end_node.g = end_node.h = end_node.f = 0
    # Initialize both open and closed list
    open_list = []
    closed_list = []
    # Add the start node
    open_list.append(start_node)
    # Loop until you find the end
    while len(open_list) > 0:
        # Get the current node
        current_node = open_list[0]
        current_index = 0
        for index, item in enumerate(open_list):
```

```
            if item.f < current_node.f:
                current_node = item
                current_index = index
        # Pop current off open list, add to closed list
        open_list.pop(current_index)
        closed_list.append(current_node)
        # Found the goal
        if current_node == end_node:
            path = []
            current = current_node
            while current is not None:
                path.append(current.position)
                current = current.parent
            return path[::-1] # Return reversed path
        # Generate children
        children = []
        for new_position in [(0, -1), (0, 1), (-1, 0), (1, 0), (-1, -1), (-1, 1), (1, -1), (1, 1)]: #
Adjacent squares
            # Get node position
            node_position = (current_node.position[0] + new_position[0],
current_node.position[1] + new_position[1])
            # Make sure within range
            if node_position[0] > (len(maze[len(maze)-1]) -1) or node_position[0] < 0 or
node_position[1] >(len(maze) - 1) or node_position[1] < 0:
                continue
            # Make sure walkable terrain
            if maze[node_position[1]][node_position[0]] != 0 and
maze[node_position[1]][node_position[0]] != 2:
                continue
            # Create new node
            new_node = Node(current_node, node_position)
```

```
            children.append(new_node)
        # Loop through children
        for child in children:
            # Child is on the closed list
            for closed_child in closed_list:
                if child.position == closed_child.position:
                    break

            else:
                # Create the f, g, and h values
                child.g = current_node.g + 1
                child.h = ((child.position[0] - end_node.position[0]) ** 2) +
((child.position[1] - end_node.position[1]) ** 2)
                child.f = child.g + child.h
                # Child is already in the open list
                for open_node in open_list:
                    if child.position == open_node.position and child.g > open_node.g:
                        break
                else:
                    # Add the child to the open list
                    open_list.append(child)
```


โปรแกรมแบบจำลองแรงทางสังคม (ผู้อพยพ)

```
import numpy as np
RED = (255, 0, 0)
# Function to find direction
def normalize(v):
    norm = np.linalg.norm(v) # [linalg.norm]return one of eight different matrix norms
    if norm == 0:
       return v
    return v/norm
```

```
# Creating Social-Force-Model of each Agent
class Agent(object):
    def __init__(self):
        self.mass = 60 # kg
        self.radius = 4 # Pedestrian radius (m)
        self.desiredSpeed = 1.34 # m/s
        self.characteristicTime = 0.5
        self.A = 2000 # Strength of social repulsive force (N)
        self.B = 0.08 # Characteristic distance of social repulsive force (m)
        self.k = 240000 # Coefficient of sliding friction (kg/m s)
        self.K = 120000 # Body compression coefficient (kg/s s)
        self.position = np.array([100, 400])
        self.destination = np.array([700,400])
        self.direction  = normalize(self.destination-self.position)
        self.actualVelocity = np.array([0, 0])
        self.desiredVelocity = self.desiredSpeed*self.direction
        self.startGridPosition = 0
        self.endGridPosition = 0
        self.currentGridPosition = 1
        self.path = []
        self.color = RED
        self.sightRange = 110
        self.alpha = 1
        self.beta = 1
        self.b1 = 0.005
        self.b2 = 0.005
        self.foundExit = False
        self.foundLeader = False
        self.success = False
    def desired_force(self): # change according to velocity
```

```
        return (self.mass*(self.desiredVelocity-
self.actualVelocity))/self.characteristicTime
    # The interaction force between people
    def f_ij(self, other):
        r_ij = self.radius + other.radius
        d_ij = np.linalg.norm(self.position - other.position)
        if d_ij == 0:
            d_ij = 0.000001
        n_ij = (self.position - other.position)/d_ij
        if(r_ij-d_ij < 0):
            g = 0
        else:
            g = r_ij-d_ij
        t_ij = np.array([-n_ij[1],n_ij[0]])
        deltaV_ij = other.actualVelocity - self.actualVelocity
        #socio-psychological force
        s_force = self.A*np.exp((r_ij-d_ij)/self.B)*n_ij
        #physical force
        p_force = self.k*g*n_ij + self.K*g*deltaV_ij*t_ij
        #value = s_force + p_force
        value = s_force
        return value
    # The interaction force between agent and wall
    def f_iw(self, wall):
        r_i = self.radius
        vec_iw = self.distance_agent_to_wall([wall[0],wall[1]],[wall[2],wall[3]])
        d_iw = np.linalg.norm(vec_iw)
        n_iw = vec_iw/d_iw
        if(r_i-d_iw < 0):
            g = 0
        else:
```

```
            g = r_i-d_iw
        t_iw = np.array([-n_iw[1],n_iw[0]])
        deltaV_iw = 0 - self.actualVelocity
        s_force = self.A*np.exp((r_i-d_iw)/self.B)*n_iw
        p_force = self.k*g*n_iw + self.K*g*deltaV_iw*t_iw
        value = s_force + p_force
        return value
    # The navigation force
    def f_ig(self, other):
        d_ig = self.position - other.position
        deltaV_ig = other.actualVelocity - self.actualVelocity
        n_force = self.alpha*self.mass*(-self.b1*(d_ig/(self.characteristicTime)**2 )-
self.b2*(deltaV_ig/self.characteristicTime))
        return n_force
    # Function to find distance between point and line
    def distance_agent_to_wall(self, start, end):
        line_vec = np.array(end) - np.array(start)
        pnt_vec = self.position - np.array(start)
        line_len = np.linalg.norm(line_vec) # Find the length of the line vector
('line_len')
        line_unitvec = line_vec/line_len # Convert line_vec to a unit vector
('line_unitvec')
        pnt_vec_scaled = pnt_vec/line_len
        t = np.dot(line_unitvec, pnt_vec_scaled) # Nearest distance from Agent and
wall
        if t < 0.0:
            t = 0.0
        elif t > 1.0:
            t = 1.0
        nearest = line_vec*t
        dist = pnt_vec - nearest
```

```
        nearest = nearest + start
        nearest_vec = self.position - nearest
        return nearest_vec
```

โปรแกรมแบบจำลองการอพยพฝูงชนบนระนาบสองมิติแบบขนาดกริด 4x4 เมตร

```
import numpy as np
import pygame
import time
from Agent import *
from Astar import *
from Leader import *
class Spot:
        def __init__(self, row, col, width, total_rows):
                self.row = row
                self.col = col
                self.x = row * width
                self.y = col * width
                self.color = WHITE
                self.width = width
                self.total_rows = total_rows
        def draw(self, screen):
                pygame.draw.rect(screen, self.color, (self.x, self.y, self.width,
                self.width))
def make_grid(rows, width):
        grid = []
        for i in range(rows):
                grid.append([])
                for j in range(rows):
                        spot = Spot(i, j, gap, rows)
                        grid[i].append(spot)
        return grid
```

```
def draw_grid(screen, rows, width):
        for i in range(rows):
                pygame.draw.line(screen, GREY, (0, i * gap), (width, i * gap))
                for j in range(rows):
                        pygame.draw.line(screen, GREY, (j * gap, 0), (j * gap, width))
def draw(screen, grid, rows, width):
        screen.fill(WHITE)
        for row in grid:
                for spot in row:
                        spot.draw(screen)
        draw_grid(screen, rows, width)
def check_position(positionX,positionY,maze):
        position = np.array([positionX,positionY])
        GridPosition = position//gap
        gridPositionX = GridPosition[0]
        gridPositionY = GridPosition[1]
        while maze[int(gridPositionY),int(gridPositionX)] == 1 or
maze[int(gridPositionY),int(gridPositionX)] == 2 :
                position = np.array([np.random.uniform(5,1195),
np.random.uniform(5,115)])
                GridPosition = position//gap
                gridPositionX = GridPosition[0]
                gridPositionY = GridPosition[1]
        return position
# Creating screen
pygame.init()
width = 1200
ROWS = 30
gap = width // ROWS
screen = pygame.display.set_mode((width,width-1080))
pygame.display.set_caption("SMF Simulator")
```

```
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLUE = (0, 255, 0)
YELLOW = (255, 255, 0)
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
PURPLE = (128, 0, 128)
ORANGE = (255, 165 ,0)
GREY = (128, 128, 128)
TURQUOISE = (64, 224, 208)
color_list = [GREEN,BLUE,PURPLE,ORANGE,TURQUOISE]
def main():
    # Define envitonment
    maze = np.array([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
          [1, 1, 0, 0, 0, 0, 0, 1, 2, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 2, 1],
          [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
    exits_position = [(90,60),(350,60),(1130,60)]
    # Initialize agents
    agents = []
    number_of_agent = 50
    for nAgent in range(number_of_agent):
        a = Agent()
        a.color = GREEN
        a.position = np.array([np.random.uniform(5,1195), np.random.uniform(5,115)])
        a.position = check_position(a.position[0],a.position[1],maze)
        a.startGridPosition = a.position//gap
        # In case see exit
        for exitt in exits_position:
                exit_destinationX = exitt[0]
                exit_destinationY= exitt[1]
```

```
            distance = (((exit_destinationX - a.position[0]) ** 2) + ((exit_destinationY -
a.position[1]) ** 2))**0.5
            if distance <= a.sightRange:
                a.destination =np.array([exit_destinationX, exit_destinationY])
                a.endGridPosition = a.destination//gap
                a.foundExit = True
                a.color = PURPLE
                break
        #In case not see exit
        else:
            a.destination = np.array([np.random.uniform(5,1195),
np.random.uniform(5,115)])
            a.endGridPosition = a.destination//gap
            a.destination = check_position(a.destination [0],a.destination [1],maze)
            a.endGridPosition = a.destination//gap
        agents.append(a)
        start = (int(a.startGridPosition[0]),int(a.startGridPosition[1]))
        end = (int(a.endGridPosition[0]),int(a.endGridPosition[1]))
        a.path = astar(maze, start, end)
    # Initialize leader
    leaders = []
    number_of_leader = 5
    for nLeader in range(number_of_leader):
        b = Leader()
        b.color = RED
        b.position = np.array([np.random.uniform(5,1195), np.random.uniform(5,115)])
        b.position = check_position(b.position[0],b.position[1],maze)
        b.startGridPosition = b.position//gap
        distance_values = []
        minimum_distance = 0
        exit_positions = []
```

```
        # Check the nearest exit
        for exitt in exits_position:
            exit_destinationX = exitt[0]
            exit_destinationY= exitt[1]
            select_exit = (exit_destinationX, exit_destinationY)
            distance = (((exit_destinationX - b.position[0]) ** 2) + ((exit_destinationY -
b.position[1]) ** 2))**0.5
            distance_values.append(distance)
            exit_positions.append(select_exit)
        # Define destination of each leader
        minimum = min(distance_values)
        for i in range(len(distance_values)):
            if distance_values[i] == minimum:
                b.destination =np.array([exit_positions[i][0], exit_positions[i][1]])
                b.endGridPosition = b.destination//gap
        leaders.append(b)
        start_leader = (int(b.startGridPosition[0]),int(b.startGridPosition[1]))
        end_leader = (int(b.endGridPosition[0]),int(b.endGridPosition[1]))
        b.path = astar(maze, start_leader, end_leader)
    # Start simulation
    sim_time = 0
    delta_time = 0.1
    grid = make_grid(ROWS, width)
    while sim_time <= 1000:
        draw(screen, grid, ROWS, width)
        sim_time += delta_time
        walls =[[0,40,80,40],[0,80,80,80],
            [280,40,360,40],[280,80,360,80],[280,40,280,80],
            [600,40,720,40],[600,80,720,80],[600,40,600,80],[720,40,720,80],
            [760,40,880,40],[760,80,880,80],[760,40,760,80],[880,40,880,80],
            [920,40,1040,40],[920,80,1040,80],[920,40,920,80],[1040,40,1040,80],
```

```
            [1120,40,1200,40],[1120,80,1200,80]]
        exit_walls = [[80,40,80,80],[360,40,360,80],[1120,40,1120,80]]

        # Check exit in sight range of follower
        for exitt in exits_position:
            exit_destinationX = exitt[0]
            exit_destinationY= exitt[1]
            distance = (((exit_destinationX - a.position[0]) ** 2) + ((exit_destinationY -
a.position[1]) ** 2))**0.5
            # Define destination of each follower
            if distance <= a.sightRange:
                a.destination = np.array([exit_destinationX,exit_destinationY])
                a.endGridPosition = a.destination//gap
        # Calculate the social force model of each follower
        for i in range(number_of_agent):
            a = agents[i]
            if not a.foundExit:
                for exitt in exits_position:
                    exit_destinationX = exitt[0]
                    exit_destinationY= exitt[1]
                    distance = (((exit_destinationX - a.position[0]) ** 2) +
((exit_destinationY - a.position[1]) ** 2))**0.5
                    if distance <= a.sightRange:
                        a.nowGridPosition = a.position//gap
                        a.destination =np.array([exit_destinationX, exit_destinationY])
                        a.endGridPosition = a.destination//gap
                        start = (int(a.nowGridPosition[0]),int(a.nowGridPosition[1]))
                        end = (int(a.endGridPosition[0]),int(a.endGridPosition[1]))
                        a.path = astar(maze, start, end)
                        a.foundExit = True
                        a.color = PURPLE
```

```
                    a.currentGridPosition = 0
                    break

        if a.currentGridPosition < len(a.path):
            x = (np.array(a.path[a.currentGridPosition])*gap)+(gap/2)
        else:
            x = np.array([0,0])
        a.direction = normalize(x-a.position)
        a.desiredVelocity = a.desiredSpeed*a.direction
        aDesired_force = a.desired_force()
        people_interaction = 0
        for j in range(number_of_agent):
            if(j!=i):
                b = agents[j]
                people_interaction += a.f_ij(b)
        for j in range(number_of_leader):
            c = leaders[j]
            people_interaction += b.f_ij(c)
        wall_interaction = 0
        for wall in walls:
            wall_interaction += a.f_iw(wall)
        navigation_force = 0
        for k in range(number_of_leader):
            c = leaders[k]
            distance = (((c.position[0] - a.position[0]) ** 2) + ((c.position[1]-
a.position[1]) ** 2))**0.5
            if distance <= a.sightRange:
                if a.foundExit == False and a.foundExit == False:
                    a.foundLeader = True
                    a.color = TURQUOISE
                    a.destination = c.position
```

```
                        a.endGridPosition = a.destination//gap
                        a.nowGridPosition = a.position//gap
                        start = (int(a.nowGridPosition[0]),int(a.nowGridPosition[1]))
                        end = (int(a.endGridPosition[0]),int(a.endGridPosition[1]))
                        a.path = astar(maze, start, end)
                        a.currentGridPosition = 0
                        navigation_force += a.f_ig(c)
                    else:
                        navigation_force += 0
                if distance < 15:
                    if a.foundLeader == True:
                        a.destination = c.destination
                        a.endGridPosition = a.destination//gap
                        a.nowGridPosition = a.position//gap
                        start = (int(a.nowGridPosition[0]),int(a.nowGridPosition[1]))
                        end = (int(a.endGridPosition[0]),int(a.endGridPosition[1]))
                        a.path = astar(maze, start, end)
                        a.currentGridPosition = 0
                        navigation_force += 0
            totalForce = aDesired_force + people_interaction + wall_interaction +
navigation_force
            a.actualVelocity = (totalForce/a.mass)*delta_time+a.actualVelocity
        # Calculate the social force model of each leader
        for i in range(number_of_leader):
            b = leaders[i]
            z = (np.array(b.path[b.currentGridPosition])*gap)+(gap/2)
            b.direction = normalize(z-b.position)
            b.desiredVelocity = b.desiredSpeed*b.direction
            bDesired_force = b.desired_force()
            people_interaction = 0
            for j in range(number_of_leader):
```

```
            if(j!=i):
                c = leaders[j]
                people_interaction += b.f_ij(c)
        for j in range(number_of_agent):
            c = agents[j]
            people_interaction += b.f_ij(c)
        wall_interaction = 0
        for wall in walls:
            wall_interaction += b.f_iw(wall)
        totalForce = bDesired_force + people_interaction + wall_interaction
        b.actualVelocity = (totalForce/b.mass) * delta_time + b.actualVelocity
    # Check if the evacuation is successful or not.
    for i in range(number_of_agent-1,-1,-1):
        a = agents[i]
        a.position = a.position + a.actualVelocity*delta_time
        if len(a.path) > 1:
            if tuple(a.position//gap) == a.path[a.currentGridPosition]:
                if a.currentGridPosition < len(a.path)-1:
                    a.currentGridPosition += 1
                elif a.foundExit == True:
                    a.success = True
                    agents.pop(i)
                    number_of_agent -= 1
                else:
                    a.destination = np.array([np.random.uniform(5,1195),
np.random.uniform(5,115)])
                    a.endGridPosition = a.destination//gap
                    a.destination = check_position(a.destination [0],a.destination
[1],maze)
                    a.endGridPosition = a.destination//gap
                    start = (int(a.startGridPosition[0]),int(a.startGridPosition[1]))
```

```
                    end = (int(a.endGridPosition[0]),int(a.endGridPosition[1]))
                    a.path = astar(maze, start, end)
                    a.currentGridPosition = 0

            elif a.foundExit == True :
                a.success = True
                agents.pop(i)
                number_of_agent -= 1
            else:
                a.destination = np.array([np.random.uniform(5,1195),
np.random.uniform(5,115)])
                a.endGridPosition = a.destination//gap
                a.destination = check_position(a.destination [0],a.destination [1],maze)
                a.endGridPosition = a.destination//gap
                start = (int(a.startGridPosition[0]),int(a.startGridPosition[1]))
                end = (int(a.endGridPosition[0]),int(a.endGridPosition[1]))
                a.path = astar(maze, start, end)
                a.currentGridPosition = 0
        if number_of_agent == 0:
            break
        for i in range(number_of_leader):
            b = leaders[i]
            b.position = b.position + b.actualVelocity*delta_time
            if len(b.path) > 1:
                if tuple(b.position//gap) == b.path[b.currentGridPosition]:
                    if b.currentGridPosition < len(b.path)-1:
                            b.currentGridPosition += 1
                    else:
                            b.success = True
        # Draw agent
        for i in range(number_of_agent):
```

```
            a = agents[i]
            if (a.success == False):
                pygame.draw.circle(screen, a.color, a.position, round(a.radius), 2)
        # Draw leader
        for i in range(number_of_leader):
            b = leaders[i]
            if (b.success == False):
                pygame.draw.circle(screen, b.color, b.position, round(b.radius), 2)
        # Draw wall
        for wall in walls:
            start_posw = np.array([wall[0],wall[1]])
            end_posw = np.array([wall[2],wall[3]])
            start_posx = start_posw
            end_posx = end_posw
            pygame.draw.line(screen, BLACK, start_posx, end_posx, 3)
        # Draw exit
        for exitt in exit_walls:
            start_posw = np.array([exitt[0],exitt[1]])
            end_posw = np.array([exitt[2],exitt[3]])
            start_posx = start_posw
            end_posx = end_posw
            pygame.draw.line(screen, RED, start_posx, end_posx, 3)
        pygame.display.update()
main()
```


ข้อกำหนดและผลการฝึกอบรมเอเจนต์ด้วยการเรียนรู้แบบเสริมกำลัง

```
Microsoft Windows [Version 10.0.19044.2251]
(c) Microsoft Corporation. All rights reserved.
C:\Unity\ML-Agents-Test04>venv\Scripts\activate
(venv) C:\Unity\ML-Agents-Test04>mlagents-learn
config/EvacuationConfiguration.yaml --run-id=TrainingEvacuation3
```

Version information:
ml-agents: 0.28.0,
ml-agents-envs: 0.28.0,
Communicator API: 1.5.0,
PyTorch: 1.7.1+cu110
[INFO] Listening on port 5004. Start training by pressing the Play button in the Unity Editor.
[INFO] Connected to Unity environment with package version 2.2.1-exp.1 and communication version 1.5.0
[INFO] Connected new brain: Evacuation?team=0
[INFO] Hyperparameters for behavior name Evacuation:
trainer_type: ppo
hyperparameters:
batch_size: 1024
buffer_size: 10240
learning_rate: 0.0003
beta: 0.005
epsilon: 0.2
lambd: 0.95
num_epoch: 3
learning_rate_schedule: linear
beta_schedule: linear
epsilon_schedule: linear
network_settings:
normalize: False
hidden_units: 256
num_layers: 1
vis_encode_type: simple
memory: None
goal_conditioning_type: hyper
deterministic: False

```
    reward_signals:
      extrinsic:
        gamma:       0.99
        strength:    1.0
        network_settings:
          normalize:         False
          hidden_units:      128
          num_layers:        2
          vis_encode_type:   simple
          memory:    None
          goal_conditioning_type:    hyper
          deterministic:     False
    init_path:       None
    keep_checkpoints:        10
    checkpoint_interval:     150000
    max_steps:       1500000
    time_horizon:    64
    summary_freq:    10000
    threaded:        False
    self_play:       None
    behavioral_cloning:      None
```


[INFO] Evacuation. Step: 10000. Time Elapsed: 145.208 s. Mean Reward: 4.000. Std of Reward: 5.000. Training.

[INFO] Evacuation. Step: 20000. Time Elapsed: 272.869 s. Mean Reward: 7.500. Std of Reward: 1.500. Training.

[INFO] Evacuation. Step: 30000. Time Elapsed: 408.689 s. Mean Reward: 13.667. Std of Reward: 11.441. Training.

[INFO] Evacuation. Step: 40000. Time Elapsed: 546.346 s. Mean Reward: 10.000. Std of Reward: 5.000. Training.

[INFO] Evacuation. Step: 50000. Time Elapsed: 668.177 s. Mean Reward: 17.667. Std of Reward: 7.587. Training.

[INFO] Evacuation. Step: 60000. Time Elapsed: 786.091 s. Mean Reward: 14.000. Std of Reward: 1.000. Training.

[INFO] Evacuation. Step: 70000. Time Elapsed: 905.025 s. Mean Reward: 5.400. Std of Reward: 6.499. Training.

[INFO] Evacuation. Step: 80000. Time Elapsed: 1024.465 s. Mean Reward: 25.000. Std of Reward: 6.000. Training.

[INFO] Evacuation. Step: 90000. Time Elapsed: 1142.301 s. Mean Reward: 20.000. Std of Reward: 0.000. Training.

[INFO] Evacuation. Step: 100000. Time Elapsed: 1263.407 s. Mean Reward: 9.000. Std of Reward: 8.124. Training.

[INFO] Evacuation. Step: 110000. Time Elapsed: 1382.253 s. Mean Reward: 15.250. Std of Reward: 15.817. Training.

[INFO] Evacuation. Step: 120000. Time Elapsed: 1501.267 s. Mean Reward: 31.000. Std of Reward: 17.000. Training.

[INFO] Evacuation. Step: 130000. Time Elapsed: 1612.941 s. Mean Reward: 9.500. Std of Reward: 4.717. Training.

[INFO] Evacuation. Step: 140000. Time Elapsed: 1730.265 s. Mean Reward: 18.167. Std of Reward: 15.699. Training.

[INFO] Evacuation. Step: 150000. Time Elapsed: 1843.638 s. Mean Reward: 37.000. Std of Reward: 15.000. Training.

[INFO] Exported results\TrainingEvacuation3\Evacuation\Evacuation-149989.onnx

[INFO] Evacuation. Step: 160000. Time Elapsed: 1951.984 s. Mean Reward: 31.500. Std of Reward: 31.500. Training.

[INFO] Evacuation. Step: 170000. Time Elapsed: 2070.250 s. Mean Reward: 66.500. Std of Reward: 42.500. Training.

[INFO] Evacuation. Step: 180000. Time Elapsed: 2183.542 s. Mean Reward: 73.000. Std of Reward: 9.000. Training.

[INFO] Evacuation. Step: 190000. Time Elapsed: 2300.568 s. Mean Reward: 95.000. Std of Reward: 0.000. Training.

[INFO] Evacuation. Step: 200000. Time Elapsed: 2416.316 s. Mean Reward: 149.000. Std of Reward: 15.000. Training.

[INFO] Evacuation. Step: 210000. Time Elapsed: 2532.632 s. Mean Reward: 175.000. Std of Reward: 25.000. Training.

[INFO] Evacuation. Step: 220000. Time Elapsed: 2642.321 s. Mean Reward: 93.000. Std of Reward: 0.000. Training.

[INFO] Evacuation. Step: 230000. Time Elapsed: 2755.934 s. Mean Reward: 89.333. Std of Reward: 29.238. Training.

[INFO] Evacuation. Step: 240000. Time Elapsed: 2867.167 s. Mean Reward: 194.000. Std of Reward: 0.000. Training.

[INFO] Evacuation. Step: 250000. Time Elapsed: 2982.377 s. Mean Reward: 102.000. Std of Reward: 36.286. Training.

[INFO] Evacuation. Step: 260000. Time Elapsed: 3095.031 s. Mean Reward: 79.000. Std of Reward: 0.000. Training.

[INFO] Evacuation. Step: 270000. Time Elapsed: 3211.049 s. Mean Reward: 173.500. Std of Reward: 46.500. Training.

[INFO] Evacuation. Step: 280000. Time Elapsed: 3328.471 s. Mean Reward: 130.250. Std of Reward: 57.578. Training.

[INFO] Evacuation. Step: 290000. Time Elapsed: 3439.830 s. Mean Reward: 139.000. Std of Reward: 0.000. Training.

[INFO] Evacuation. Step: 300000. Time Elapsed: 3556.415 s. Mean Reward: 131.333. Std of Reward: 82.661. Training.

[INFO] Exported results\TrainingEvacuation3\Evacuation\Evacuation-299944.onnx

[INFO] Evacuation. Step: 310000. Time Elapsed: 3688.889 s. Mean Reward: 227.500. Std of Reward: 6.500. Training.

[INFO] Evacuation. Step: 320000. Time Elapsed: 3817.791 s. Mean Reward: 214.000. Std of Reward: 11.000. Training.

[INFO] Evacuation. Step: 330000. Time Elapsed: 3944.569 s. Mean Reward: 225.000. Std of Reward: 0.000. Training.

[INFO] Evacuation. Step: 340000. Time Elapsed: 4064.820 s. Mean Reward: 211.000. Std of Reward: 23.000. Training.

[INFO] Evacuation. Step: 350000. Time Elapsed: 4193.476 s. Mean Reward: 244.500. Std of Reward: 9.500. Training.

[INFO] Evacuation. Step: 360000. Time Elapsed: 4318.957 s. Mean Reward: 125.250. Std of Reward: 99.253. Training.

[INFO] Evacuation. Step: 370000. Time Elapsed: 4433.655 s. Mean Reward: 150.000. Std of Reward: 4.000. Training.

[INFO] Evacuation. Step: 380000. Time Elapsed: 4547.927 s. Mean Reward: 135.333. Std of Reward: 79.672. Training.

[INFO] Evacuation. Step: 390000. Time Elapsed: 4663.709 s. Mean Reward: 267.500. Std of Reward: 9.500. Training.

[INFO] Evacuation. Step: 400000. Time Elapsed: 4783.327 s. Mean Reward: 89.500. Std of Reward: 35.008. Training.

[INFO] Evacuation. Step: 410000. Time Elapsed: 4896.419 s. Mean Reward: 257.000. Std of Reward: 0.000. Training.

[INFO] Evacuation. Step: 420000. Time Elapsed: 5015.781 s. Mean Reward: 115.333. Std of Reward: 54.585. Training.

[INFO] Evacuation. Step: 430000. Time Elapsed: 5131.620 s. Mean Reward: 217.000. Std of Reward: 67.000. Training.

[INFO] Evacuation. Step: 440000. Time Elapsed: 5248.871 s. Mean Reward: 168.667. Std of Reward: 54.707. Training.

[INFO] Evacuation. Step: 450000. Time Elapsed: 5363.667 s. Mean Reward: 238.000. Std of Reward: 44.000. Training.

[INFO] Exported results\TrainingEvacuation3\Evacuation\Evacuation-449980.onnx

[INFO] Evacuation. Step: 460000. Time Elapsed: 5481.453 s. Mean Reward: 119.000. Std of Reward: 80.337. Training.

[INFO] Evacuation. Step: 470000. Time Elapsed: 5599.479 s. Mean Reward: 105.667. Std of Reward: 11.614. Training.

[INFO] Evacuation. Step: 480000. Time Elapsed: 5719.272 s. Mean Reward: 90.700. Std of Reward: 90.104. Training.

[INFO] Evacuation. Step: 490000. Time Elapsed: 5838.179 s. Mean Reward: 129.000. Std of Reward: 89.869. Training.

[INFO] Evacuation. Step: 500000. Time Elapsed: 5951.010 s. Mean Reward: 183.000. Std of Reward: 0.000. Training.

[INFO] Evacuation. Step: 510000. Time Elapsed: 6070.275 s. Mean Reward: 206.333. Std of Reward: 33.290. Training.

[INFO] Evacuation. Step: 520000. Time Elapsed: 6185.647 s. Mean Reward: 278.000. Std of Reward: 4.000. Training.

[INFO] Evacuation. Step: 530000. Time Elapsed: 6302.406 s. Mean Reward: 171.333. Std of Reward: 91.376. Training.

[INFO] Evacuation. Step: 540000. Time Elapsed: 6419.698 s. Mean Reward: 150.333. Std of Reward: 66.620. Training.

[INFO] Evacuation. Step: 550000. Time Elapsed: 6535.632 s. Mean Reward: 212.500. Std of Reward: 58.500. Training.

[INFO] Evacuation. Step: 560000. Time Elapsed: 6651.050 s. Mean Reward: 220.667. Std of Reward: 35.198. Training.

[INFO] Evacuation. Step: 570000. Time Elapsed: 6768.637 s. Mean Reward: 182.000. Std of Reward: 112.395. Training.

[INFO] Evacuation. Step: 580000. Time Elapsed: 6884.348 s. Mean Reward: 140.333. Std of Reward: 77.590. Training.

[INFO] Evacuation. Step: 590000. Time Elapsed: 6998.359 s. Mean Reward: 102.000. Std of Reward: 79.022. Training.

[INFO] Evacuation. Step: 600000. Time Elapsed: 7114.810 s. Mean Reward: 250.500. Std of Reward: 37.500. Training.

[INFO] Exported results\TrainingEvacuation3\Evacuation\Evacuation-599937.onnx

[INFO] Evacuation. Step: 610000. Time Elapsed: 7232.958 s. Mean Reward: 146.000. Std of Reward: 63.131. Training.

[INFO] Evacuation. Step: 620000. Time Elapsed: 7346.150 s. Mean Reward: 254.500. Std of Reward: 15.500. Training.

[INFO] Evacuation. Step: 630000. Time Elapsed: 7459.890 s. Mean Reward: 244.000. Std of Reward: 0.000. Training.

[INFO] Evacuation. Step: 640000. Time Elapsed: 7573.948 s. Mean Reward: 165.000. Std of Reward: 76.912. Training.

[INFO] Evacuation. Step: 650000. Time Elapsed: 7690.652 s. Mean Reward: 144.500. Std of Reward: 104.186. Training.

[INFO] Evacuation. Step: 660000. Time Elapsed: 7807.936 s. Mean Reward: 104.667. Std of Reward: 100.599. Training.

[INFO] Evacuation. Step: 670000. Time Elapsed: 7918.610 s. Mean Reward: 175.000. Std of Reward: 120.000. Training.

[INFO] Evacuation. Step: 680000. Time Elapsed: 8032.077 s. Mean Reward: 231.000. Std of Reward: 21.000. Training.

[INFO] Evacuation. Step: 690000. Time Elapsed: 8140.885 s. Mean Reward: 182.000. Std of Reward: 47.000. Training.

[INFO] Evacuation. Step: 700000. Time Elapsed: 8258.913 s. Mean Reward: 137.500. Std of Reward: 68.485. Training.

[INFO] Evacuation. Step: 710000. Time Elapsed: 8375.758 s. Mean Reward: 175.667. Std of Reward: 98.253. Training.

[INFO] Evacuation. Step: 720000. Time Elapsed: 8492.273 s. Mean Reward: 229.500. Std of Reward: 49.500. Training.

[INFO] Evacuation. Step: 730000. Time Elapsed: 8608.367 s. Mean Reward: 159.000. Std of Reward: 126.200. Training.

[INFO] Evacuation. Step: 740000. Time Elapsed: 8723.352 s. Mean Reward: 109.000. Std of Reward: 70.370. Training.

[INFO] Evacuation. Step: 750000. Time Elapsed: 8835.131 s. Mean Reward: 200.000. Std of Reward: 54.000. Training.

[INFO] Exported results\TrainingEvacuation3\Evacuation\Evacuation-749969.onnx

[INFO] Evacuation. Step: 760000. Time Elapsed: 8951.261 s. Mean Reward: 280.000. Std of Reward: 10.000. Training.

[INFO] Evacuation. Step: 770000. Time Elapsed: 9064.491 s. Mean Reward: 280.500. Std of Reward: 4.500. Training.

[INFO] Evacuation. Step: 780000. Time Elapsed: 9180.653 s. Mean Reward: 266.500. Std of Reward: 3.500. Training.

[INFO] Evacuation. Step: 790000. Time Elapsed: 9296.067 s. Mean Reward: 143.750. Std of Reward: 47.804. Training.

[INFO] Evacuation. Step: 800000. Time Elapsed: 9408.810 s. Mean Reward: 237.000. Std of Reward: 0.000. Training.

[INFO] Evacuation. Step: 810000. Time Elapsed: 9524.103 s. Mean Reward: 180.667. Std of Reward: 102.925. Training.

[INFO] Evacuation. Step: 820000. Time Elapsed: 9637.791 s. Mean Reward: 270.000. Std of Reward: 8.000. Training.

[INFO] Evacuation. Step: 830000. Time Elapsed: 9753.491 s. Mean Reward: 110.667. Std of Reward: 98.191. Training.

[INFO] Evacuation. Step: 840000. Time Elapsed: 9865.220 s. Mean Reward: 245.000. Std of Reward: 44.000. Training.

[INFO] Evacuation. Step: 850000. Time Elapsed: 9982.004 s. Mean Reward: 225.000. Std of Reward: 51.000. Training.

[INFO] Evacuation. Step: 860000. Time Elapsed: 10093.954 s. Mean Reward: 217.500. Std of Reward: 26.500. Training.

[INFO] Evacuation. Step: 870000. Time Elapsed: 10209.347 s. Mean Reward: 251.000. Std of Reward: 15.000. Training.

[INFO] Evacuation. Step: 880000. Time Elapsed: 10325.585 s. Mean Reward: 238.500. Std of Reward: 35.500. Training.

[INFO] Evacuation. Step: 890000. Time Elapsed: 10440.796 s. Mean Reward: 273.000. Std of Reward: 13.000. Training.

[INFO] Evacuation. Step: 900000. Time Elapsed: 10554.113 s. Mean Reward: 224.500. Std of Reward: 4.500. Training.

[INFO] Exported results\TrainingEvacuation3\Evacuation\Evacuation-899960.onnx

[INFO] Evacuation. Step: 910000. Time Elapsed: 10669.686 s. Mean Reward: 223.000. Std of Reward: 56.000. Training.

[INFO] Evacuation. Step: 920000. Time Elapsed: 10785.483 s. Mean Reward: 283.500. Std of Reward: 3.500. Training.

[INFO] Evacuation. Step: 930000. Time Elapsed: 10897.808 s. Mean Reward: 260.500. Std of Reward: 18.500. Training.

[INFO] Evacuation. Step: 940000. Time Elapsed: 11014.524 s. Mean Reward: 224.333. Std of Reward: 46.686. Training.

[INFO] Evacuation. Step: 950000. Time Elapsed: 11130.447 s. Mean Reward: 280.000. Std of Reward: 0.000. Training.

[INFO] Evacuation. Step: 960000. Time Elapsed: 11242.858 s. Mean Reward: 215.000. Std of Reward: 63.013. Training.

[INFO] Evacuation. Step: 970000. Time Elapsed: 11359.260 s. Mean Reward: 116.000. Std of Reward: 101.181. Training.

[INFO] Evacuation. Step: 980000. Time Elapsed: 11470.591 s. Mean Reward: 255.500. Std of Reward: 35.500. Training.

[INFO] Evacuation. Step: 990000. Time Elapsed: 11585.616 s. Mean Reward: 251.000. Std of Reward: 0.000. Training.

[INFO] Evacuation. Step: 1000000. Time Elapsed: 11700.404 s. Mean Reward: 222.000. Std of Reward: 67.000. Training.

[INFO] Evacuation. Step: 1010000. Time Elapsed: 11816.884 s. Mean Reward: 278.500. Std of Reward: 0.500. Training.

[INFO] Evacuation. Step: 1020000. Time Elapsed: 11931.935 s. Mean Reward: 285.500. Std of Reward: 5.500. Training.

[INFO] Evacuation. Step: 1030000. Time Elapsed: 12048.586 s. Mean Reward: 207.000. Std of Reward: 75.419. Training.

[INFO] Evacuation. Step: 1040000. Time Elapsed: 12165.631 s. Mean Reward: 297.500. Std of Reward: 1.500. Training.

[INFO] Evacuation. Step: 1050000. Time Elapsed: 12280.072 s. Mean Reward: 230.000. Std of Reward: 0.000. Training.

[INFO] Exported results\TrainingEvacuation3\Evacuation\Evacuation-1049985.onnx

[INFO] Evacuation. Step: 1060000. Time Elapsed: 12397.121 s. Mean Reward: 171.000. Std of Reward: 100.257. Training.

[INFO] Evacuation. Step: 1070000. Time Elapsed: 12513.859 s. Mean Reward: 186.500. Std of Reward: 44.500. Training.

[INFO] Evacuation. Step: 1080000. Time Elapsed: 12631.776 s. Mean Reward: 221.333. Std of Reward: 67.460. Training.

[INFO] Evacuation. Step: 1090000. Time Elapsed: 12753.182 s. Mean Reward: 293.000. Std of Reward: 0.000. Training.

[INFO] Evacuation. Step: 1100000. Time Elapsed: 12871.792 s. Mean Reward: 188.000. Std of Reward: 106.518. Training.

[INFO] Evacuation. Step: 1110000. Time Elapsed: 12993.089 s. Mean Reward: 171.333. Std of Reward: 74.906. Training.

[INFO] Evacuation. Step: 1120000. Time Elapsed: 13109.679 s. Mean Reward: 167.500. Std of Reward: 77.500. Training.

[INFO] Evacuation. Step: 1130000. Time Elapsed: 13225.839 s. Mean Reward: 292.000. Std of Reward: 3.000. Training.

[INFO] Evacuation. Step: 1140000. Time Elapsed: 13345.249 s. Mean Reward: 208.000. Std of Reward: 79.049. Training.

[INFO] Evacuation. Step: 1150000. Time Elapsed: 13462.660 s. Mean Reward: 285.500. Std of Reward: 5.500. Training.

[INFO] Evacuation. Step: 1160000. Time Elapsed: 13578.650 s. Mean Reward: 238.000. Std of Reward: 0.000. Training.

[INFO] Evacuation. Step: 1170000. Time Elapsed: 13696.321 s. Mean Reward: 247.000. Std of Reward: 22.450. Training.

[INFO] Evacuation. Step: 1180000. Time Elapsed: 13814.632 s. Mean Reward: 256.500. Std of Reward: 9.500. Training.

[INFO] Evacuation. Step: 1190000. Time Elapsed: 13932.691 s. Mean Reward: 286.000. Std of Reward: 0.000. Training.

[INFO] Evacuation. Step: 1200000. Time Elapsed: 14051.309 s. Mean Reward: 194.000. Std of Reward: 77.644. Training.

[INFO] Exported results\TrainingEvacuation3\Evacuation\Evacuation-1199990.onnx

[INFO] Evacuation. Step: 1210000. Time Elapsed: 14167.572 s. Mean Reward: 191.333. Std of Reward: 117.284. Training.

[INFO] Evacuation. Step: 1220000. Time Elapsed: 14286.044 s. Mean Reward: 169.333. Std of Reward: 109.180. Training.

[INFO] Evacuation. Step: 1230000. Time Elapsed: 14402.445 s. Mean Reward: 280.500. Std of Reward: 6.500. Training.

[INFO] Evacuation. Step: 1240000. Time Elapsed: 14522.583 s. Mean Reward: 231.000. Std of Reward: 40.000. Training.

[INFO] Evacuation. Step: 1250000. Time Elapsed: 14643.517 s. Mean Reward: 223.333. Std of Reward: 51.526. Training.

[INFO] Evacuation. Step: 1260000. Time Elapsed: 14763.516 s. Mean Reward: 115.000. Std of Reward: 102.853. Training.

[INFO] Evacuation. Step: 1270000. Time Elapsed: 14882.272 s. Mean Reward: 269.000. Std of Reward: 26.495. Training.

[INFO] Evacuation. Step: 1280000. Time Elapsed: 15000.654 s. Mean Reward: 295.000. Std of Reward: 0.000. Training.

[INFO] Evacuation. Step: 1290000. Time Elapsed: 15117.938 s. Mean Reward: 274.000. Std of Reward: 19.000. Training.

[INFO] Evacuation. Step: 1300000. Time Elapsed: 15236.767 s. Mean Reward: 220.000. Std of Reward: 91.699. Training.

[INFO] Evacuation. Step: 1310000. Time Elapsed: 15355.143 s. Mean Reward: 200.000. Std of Reward: 81.000. Training.

[INFO] Evacuation. Step: 1320000. Time Elapsed: 15471.993 s. Mean Reward: 272.000. Std of Reward: 23.000. Training.

[INFO] Evacuation. Step: 1330000. Time Elapsed: 15591.408 s. Mean Reward: 118.400. Std of Reward: 109.870. Training.

[INFO] Evacuation. Step: 1340000. Time Elapsed: 15710.445 s. Mean Reward: 217.000. Std of Reward: 53.000. Training.

[INFO] Evacuation. Step: 1350000. Time Elapsed: 15826.968 s. Mean Reward: 286.000. Std of Reward: 12.000. Training.

[INFO] Exported results\TrainingEvacuation3\Evacuation\Evacuation-1349977.onnx

[INFO] Evacuation. Step: 1360000. Time Elapsed: 15944.122 s. Mean Reward: 278.000. Std of Reward: 0.000. Training.

[INFO] Evacuation. Step: 1370000. Time Elapsed: 16061.706 s. Mean Reward: 285.500. Std of Reward: 4.500. Training.

[INFO] Evacuation. Step: 1380000. Time Elapsed: 16178.704 s. Mean Reward: 232.000. Std of Reward: 10.000. Training.

[INFO] Evacuation. Step: 1390000. Time Elapsed: 16295.398 s. Mean Reward: 277.000. Std of Reward: 14.000. Training.

[INFO] Evacuation. Step: 1400000. Time Elapsed: 16411.483 s. Mean Reward: 262.000. Std of Reward: 0.000. Training.

[INFO] Evacuation. Step: 1410000. Time Elapsed: 16529.716 s. Mean Reward: 282.500. Std of Reward: 12.500. Training.

[INFO] Evacuation. Step: 1420000. Time Elapsed: 16649.024 s. Mean Reward: 253.333. Std of Reward: 29.010. Training.

[INFO] Evacuation. Step: 1430000. Time Elapsed: 16768.686 s. Mean Reward: 253.000. Std of Reward: 29.000. Training.

[INFO] Evacuation. Step: 1440000. Time Elapsed: 16882.442 s. Mean Reward: 283.500. Std of Reward: 9.500. Training.

[INFO] Evacuation. Step: 1450000. Time Elapsed: 16994.461 s. Mean Reward: 254.000. Std of Reward: 0.000. Training.

[INFO] Evacuation. Step: 1460000. Time Elapsed: 17110.110 s. Mean Reward: 250.333. Std of Reward: 28.709. Training.

[INFO] Evacuation. Step: 1470000. Time Elapsed: 17226.183 s. Mean Reward: 178.000. Std of Reward: 28.000. Training.

[INFO] Evacuation. Step: 1480000. Time Elapsed: 17342.298 s. Mean Reward: 280.500. Std of Reward: 14.500. Training.

[INFO] Evacuation. Step: 1490000. Time Elapsed: 17455.359 s. Mean Reward: 277.500. Std of Reward: 16.500. Training.

[INFO] Evacuation. Step: 1500000. Time Elapsed: 17572.857 s. Mean Reward: 106.500. Std of Reward: 110.780. Training.

[INFO] Exported results\TrainingEvacuation3\Evacuation\Evacuation-1499965.onnx

[INFO] Exported results\TrainingEvacuation3\Evacuation\Evacuation-1500029.onnx

[INFO] Copied results\TrainingEvacuation3\Evacuation\Evacuation-1500029.onnx to results\TrainingEvacuation3\Evacuation.onnx.

บรรณานุกรม

1. SRT. *History*. 2021 23 November 2022]; Available from: https://www.railway.co.th/AboutUs/Blog_detail?value1=00DE5502B5AA7B42A92BE9FF953D8EBD010000009195357A2E56E2C33B0733673D37F9CC71C2BC7787F26521F0B819D15572186E&value2=00DE5502B5AA7B42A92BE9FF953D8EBD010000006CFD6D939058106716B128C328D64C4824C6E86643978CA5957CD0EDE6E8E575.
2. MRTA. *History*. 2021 23 November 2022]; Available from: https://www.mrta.co.th/th/about_mrta/history/.
3. BTS. *History*. 2021 23 November 2022]; Available from: https://www.bts.co.th/info/info-history.html.
4. MRTA. รถไฟฟ้ามหานคร สายเฉลิมรัชมงคล. 2021 23 November 2022]; Available from: https://www.mrta.co.th/th/services/bl/.
5. easternhsr. รถไฟฟ้าเชื่อมท่าอากาศยานสุวรรณภูมิ. 2022 23 November 2022]; Available from: https://www.easternhsr.com/hsr/airportraillink.
6. MRTA. รถไฟฟ้ามหานคร สายฉลองรัชธรรม. 2021 23 November 2022]; Available from: https://www.mrta.co.th/th/services/ppl/.
7. SRTET. รถไฟฟ้าชานเมืองสายสีแดง. 2022 23 November 2022]; Available from: https://www.srtet.co.th/th.
8. MRTA. โครงการรถไฟฟ้าสายสีชมพู. 2021 23 November 2022]; Available from: https://www.mrta.co.th/th/projectelectrictrain/bangkok-and-vicinities/pinkline/.
9. MRTA. โครงการรถไฟฟ้าสายสีเหลือง. 2021 23 November 2022]; Available from: https://www.mrta.co.th/th/projectelectrictrain/bangkok-and-vicinities/yellowline/.
10. MRTA. โครงการรถไฟฟ้าสายสีส้ม. 2021 23 November 2022]; Available from: https://www.mrta.co.th/th/projectelectrictrain/bangkok-and-vicinities/orangeline/.
11. SRT. สถานีกลางบางซื่อ. 2021 23 November 2022]; Available from: https://www.railway.co.th/More/Knowledge_Detail?value1=00DE5502B5AA7B42A92BE9FF953D8EBD0100000052302CDE8F6D01CBCFD7155C6522121FCCAD576ECE

C1C5F9E6990CC4AAF8D58F&value2=00DE5502B5AA7B42A92BE9FF953D8EBD0100000002C64CFF1648BE932C1D444812350D0F16F5119458E8BF0F33F39FCA9A2F1C3D.

12. Association, N.F.P., *NFPA 130 Standard for Fixed Guideway Transit Systems.* National Fire Protection Association, Massachusetts, 1995.
13. Shiwakoti, N., et al., *Likely behaviours of passengers under emergency evacuation in train station.* Safety science, 2017. **91**: p. 40-48.
14. Helbing, D., I. Farkas, and T. Vicsek, *Simulating dynamical features of escape panic.* Nature, 2000. **407**(6803): p. 487-490.
15. Zhou, M., et al., *Optimization of crowd evacuation with leaders in urban rail transit stations.* IEEE transactions on intelligent transportation systems, 2019. **20**(12): p. 4476-4487.
16. Ming, R. and X. Peng. *Study on the social force model of personnel evacuation in large stadiums*. in *2017 International Conference on Service Systems and Service Management*. 2017. IEEE.
17. Yang, X., et al., *Guided crowd dynamics via modified social force model.* Physica A: Statistical Mechanics and its Applications, 2014. **411**: p. 63-73.
18. Zhou, M., et al., *Modeling and simulation of crowd evacuation with signs at subway platform: A case study of beijing subway stations.* IEEE Transactions on Intelligent Transportation Systems, 2020.
19. Wang, Q., et al., *Improved multi-agent reinforcement learning for path planning-based crowd simulation.* IEEE Access, 2019. **7**: p. 73841-73855.
20. Zheng, S. and H. Liu, *Improved multi-agent deep deterministic policy gradient for path planning-based crowd simulation.* IEEE Access, 2019. **7**: p. 147755-147770.
21. Schulman, J., et al., *Proximal policy optimization algorithms.* arXiv preprint arXiv:1707.06347, 2017.
22. Simonini, T. *An Introduction to Deep Reinforcement Learning*. Oct 9, 2020; Available from: https://thomassimonini.medium.com/an-introduction-to-deep-reinforcement-learning-17a565999c0c.

23. Majumder, A., *Deep Reinforcement Learning in Unity With Unity ML Toolkit* 2021.

24. Simonini, T. *An Introduction to Unity ML-Agents*. Jan 30,2020; Available from: https://towardsdatascience.com/an-introduction-to-unity-ml-agents-6238452fcf4c.

25. Helbing, D. and P. Molnar, *Social force model for pedestrian dynamics.* Physical review E, 1995. **51**(5): p. 4282.

26. Hart, P.E., N.J. Nilsson, and B. Raphael, *A formal basis for the heuristic determination of minimum cost paths.* IEEE transactions on Systems Science and Cybernetics, 1968. **4**(2): p. 100-107.

27. Borromeo, N.A., *Hads-On Unity 2020 Game Development*. July 2020, Livery Place, 35 Livery Street, Birmingham, B3 2PB, UK: Packt Publishing Ltd.

28. Technologies, U. *Inner Workings of the Navigation System*. 2020 2021-12-18]; Available from: https://docs.unity3d.com/Manual/nav-InnerWorkings.html.

29. Rafiq, A., T.A.A. Kadir, and S.N. Ihsan. *Pathfinding Algorithms in game development*. in *IOP Conference Series: Materials Science and Engineering*. 2020. IOP Publishing.

30. Qiang, W. and Z. Zhongli. *Reinforcement learning model, algorithms and its application*. in *2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*. 2011. IEEE.

ประวัติผู้เขียน

ชื่อ-สกุล	นายนิธินันท์ สินพันธุ์
วัน เดือน ปี เกิด	15 พฤศจิการยน 2540
สถานที่เกิด	โรงพยาบาลชัยนาทนเรนทร
วุฒิการศึกษา	ปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิชา วิศวกรรมเครื่องกล สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ที่อยู่ปัจจุบัน	276 ถนนชัยณรงค์ ตำบลในเมือง อำเภอเมืองชัยนาท จังหวัดชัยนาท 17000