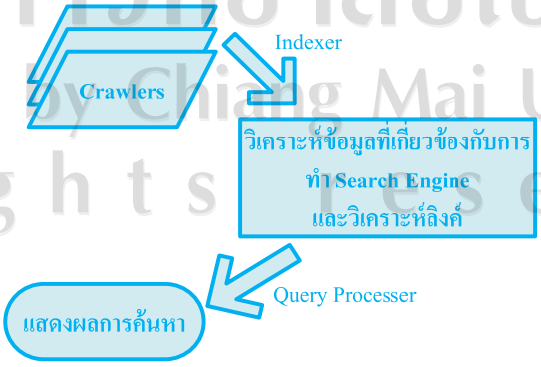
บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

การจัดลำดับเว็บเพจ (Web Page Ranking) บนระบบแบบรวมศูนย์ (Centralized System) ทำให้เกิดปัญหาความล่าช้าของการคำนวณและการใช้ทรัพยากร (Resource) ที่สูง เนื่องมาจากจำนวนของเว็บเพจ (Web Page) ที่เพิ่มขึ้น ทำให้โครงสร้างของเว็บลิงค์กราฟ (Web Link Graph) ที่นำมาคำนวณมีความซับซ้อนมากขึ้น ใช้ขนาดของหน่วยความจำ (Memory) ในการประมวลผลมากขึ้น รวมทั้งการใช้เนื้อที่ของฮาร์ดดิสก์ (Harddisk) ในการเก็บข้อมูลมากขึ้นตามไปด้วย

ในวิทยานิพนธ์ฉบับนี้ได้ศึกษาวิธีการจัดลำดับเว็บเพจบนเครือข่ายเพียร์ทูเพียร์ (Peer to Peer Network) ซึ่งเป็นการจัดลำดับเว็บเพจบนระบบกระจายศูนย์ (Distributed System) โดยนำเสนอวิธีการจัดลำดับเว็บเพจแบบเพิ่มขึ้น (Incremental Web Page Ranking) เพื่อเพิ่มประสิทธิภาพในแง่เวลาในการจัดลำดับเว็บเพจ

2.1 กูเกิล (Google)

กูเกิล [1] แบ่งการทำงานเป็น 3 ขั้นตอนหลักๆ ดังแสดงการทำงานของกูเกิลในรูปที่ 2.1 โดยจะเริ่มจากการค้นหาเว็บไซต์บนอินเตอร์เน็ต จากนั้นนำเว็บไซต์ทั้งหมดที่ค้นหาได้มาทำการวิเคราะห์ข้อมูล และสุดท้ายแสดงผลการสอบถาม (Query) ตามคำค้น (Keyword) ของผู้ใช้งาน



รูปที่ 2.1 ขั้นตอนการทำงานของกูเกิล

การค้นหาเว็บไซต์บนอินเตอร์เน็ต กูเกิลจะใช้เว็บคลอร์เลอร์ (Web Crawler) หรือเว็บสไปเดอร์ (Web Spider) ซึ่งเป็นโปรแกรมประยุกต์สำหรับค้นหาเว็บไซต์ทั้งหมดบนอินเตอร์เน็ต และนำกลับมาเก็บไว้ในในสโตร์เซิร์ฟเวอร์ (Store Server)

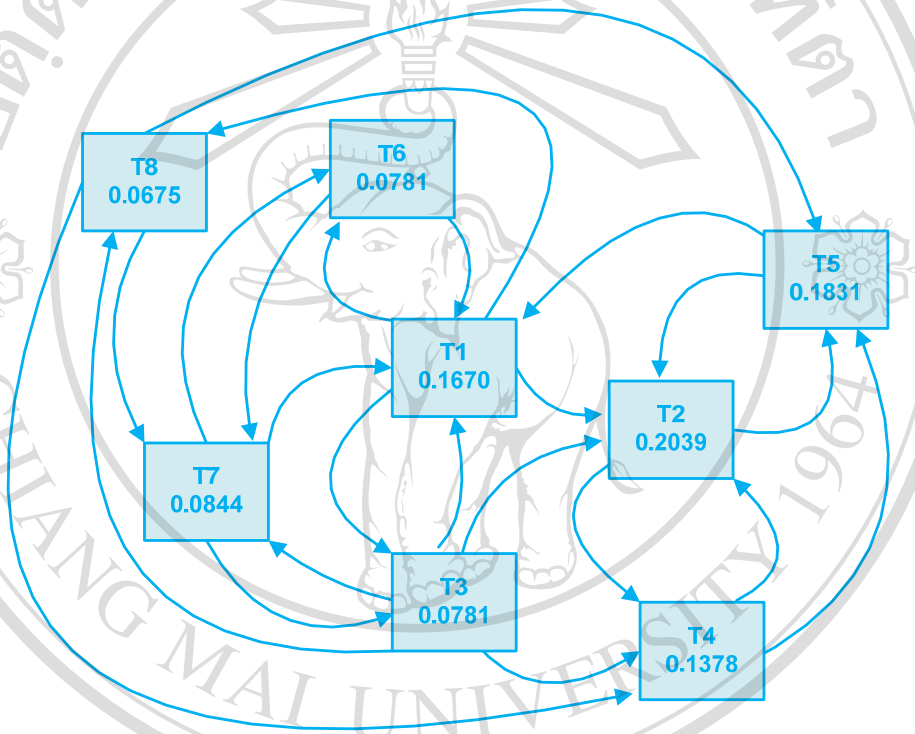
สำหรับแต่ละเว็บเพจที่ได้จากเว็บคลอร์เลอร์ จะถูกนำมาแยกเอาข้อมูลที่จำเป็นสำหรับการพัฒนาเสิร์ชเอ็นจินออกมาวิเคราะห์ด้วยกระบวนการต่างๆ ซึ่งกระบวนการหนึ่งที่เป็นพื้นฐานสำคัญของการพัฒนาเสิร์ชเอ็นจินคือการแยกลิงค์ (Link) ออกจากเว็บเพจเพื่อจัดลำดับเว็บเพจ (Webpage Ranking) ซึ่งกูเกิลเรียกขั้นตอนนี้ว่าการคำนวณค่าเพจแรงค์ (Page Rank) โดยจะพิจารณาจากลิงค์เข้า (Back Link) และลิงค์ออก (Forward Link) ของเว็บเพจ

การแสดงผลการสอบถามเป็นส่วนติดต่อกับผู้ใช้งาน โดยกูเกิลจะรับคำค้นจากผู้ใช้งานผ่านทางเว็บบราวเซอร์ (Web Browser) จากนั้นตัวประมวลผลการสอบถาม (Query Processer) จะทำการสอบถาม และแสดงผลกลับไปยังเว็บบราวเซอร์ของผู้ใช้งานที่ทำการค้นหา ดังแสดงส่วนต่อประสานกับผู้ใช้ (User Interface) ของกูเกิลในรูปที่ 2.2

รูปที่ 2.2 แสดงอินเตอร์เฟสของกูเกิล

(http://www.google.com/)

โครงสร้างของลิงค์เข้าและลิงค์ออกของเว็บเพจทั้งหมดที่ได้จากการวิเคราะห์ลิงค์ สามารถแทนได้ด้วยกราฟแบบมีทิศทาง (Directed Graph) โดยกำหนดให้แต่ละเว็บเพจแทนด้วยโหนด (Node) ของกราฟ และกำหนดให้แต่ละลิงค์ที่เชื่อมกันระหว่างเว็บเพจแทนด้วยเอดจ์ (Edges) ของกราฟ ซึ่งมีทิศทางจากเว็บเพจหนึ่งไปยังอีกเว็บเพจหนึ่ง เรียกกราฟของเว็บเพจที่มีความสัมพันธ์กันนี้ว่าเว็บลิงค์กราฟ ซึ่งเว็บลิงค์กราฟดังกล่าวสามารถนำไปคำนวณค่าเพจแรงค์ของแต่ละเว็บเพจได้ ดังแสดงตัวอย่างของเว็บลิงค์กราฟและค่าเพจแรงค์ในรูปที่ 2.3



รูปที่ 2.3 ตัวอย่างของเว็บลิงค์กราฟและค่าเพจแรงค์

เมื่อต้องการหาค่าเพจแรงค์ของแต่เว็บเพจในเว็บลิงค์กราฟ ทำได้ด้วยการคำนวณแบบขั้นตอนวิธีการทำซ้ำ (Iterative Algorithm) และมีผลลัพธ์ของสมการลู่เข้าหาค่าใดค่าหนึ่ง (Converge) ซึ่งจำนวนรอบของการปรับค่าเพจแรงค์เพื่อให้เข้าใกล้ค่าที่เหมาะสมที่สุดนั้น ขึ้นอยู่กับขนาดและโครงสร้างของเว็บลิงค์กราฟ

เนื่องจากกูเกิลเป็นเสิร์ชเอนจินบนระบบแบบรวมศูนย์ ทำให้ประสิทธิภาพในด้านเวลาการคำนวณ (Computational Time) ลดลง เมื่อขนาดของเว็บลิงค์กราฟมีขนาดใหญ่ขึ้นและมีโครงสร้างของเว็บลิงค์กราฟที่ซับซ้อนมากขึ้น

2.2 เพจแรงค์

ขั้นตอนวิธีการคำนวณค่าเพจแรงค์เป็นการวิเคราะห์ความสัมพันธ์ของลิงค์ ที่พัฒนาจากงานวิจัยของบริษัทกูเกิล [1, 3] โดยหลักการจะพิจารณาจากลิงค์เข้าและลิงค์ออก สำหรับในหนึ่งเว็บเพจจะพิจารณาค่าเพจแรงค์จากเว็บเพจอื่น ที่มีลิงค์เข้ามายังเว็บเพจที่กำลังพิจารณา แต่ในความเป็นจริงลิงค์ที่เข้ามายังเว็บเพจที่กำลังพิจารณานี้ไม่ได้ให้ค่าเพจแรงค์ทั้งหมดแก่เว็บเพจที่กำลังพิจารณา เนื่องจากเว็บเพจที่ลิงค์เข้ามานั้นอาจไม่ได้ลิงค์มาที่เว็บเพจที่กำลังพิจารณาเพียงจุดเดียว แต่ยังมีลิงค์ไปเว็บเพจอื่นๆ ด้วยทำให้ต้องมีการแบ่งค่าเพจแรงค์ตามจำนวนของลิงค์ที่ออกจากเว็บเพจนั้นด้วย

จากหลักการของการคำนวณเพจแรงค์ที่กล่าวมา สามารถเขียนได้ตามสมการที่ (2.1) และสมการที่ (2.2) ดังต่อไปนี้

$$PR_A = (1\text{-}d) + d\sum_{i=1}^{N}\left(\frac{PR(Ti)}{Out(Ti)}\right) \quad \ldots (2.1)$$

หรือ

$$PR_A = \left(\frac{1-d}{N}\right) + d\sum_{i=1}^{N}\left(\frac{PR(Ti)}{Out(Ti)}\right) \quad \ldots (2.2)$$

โดย

d คือ ค่า damping factor จะกำหนดให้ใช้ค่า 0.85 [8]
N คือ จำนวนเว็บเพจทั้งหมดในเว็บลิงค์กราฟ
Ti คือ เว็บลิงค์ที่เข้ามายังเพจ A
Out(Ti) คือจำนวนลิงค์ที่ออกจากเพจ Ti

สมการที่ (2.1) เป็นสมการแรกที่ถูกคิดขึ้นโดยบริษัทกูเกิล [1] ซึ่งนำหลักการของแรนดอมเซอร์เฟอร์โมเดล (Random Surfer Model) มาประยุกต์ใช้โดยมีแนวคิดว่า ในการเยี่ยมชมเว็บเพจใดๆ ก็ตาม หากผู้เยี่ยมชมต้องการเปลี่ยนไปเยี่ยมชมเว็บเพจอื่น สามารถเลือกคลิกเปลี่ยนเพจได้แค่ครั้งเดียว ดังนั้นโอกาสที่ลิงค์ที่อยู่ภายในเว็บเพจที่กำลังถูกเยี่ยมชมจะถูกคลิกจะเป็น $\frac{1}{\text{out(P)}}$ โดยที่ out(p) คือจำนวนลิงค์ออก เช่นเว็บเพจ A มีลิงค์ออก 20 ลิงค์หากปัจจุบันผู้เยี่ยมชมกำลังเยี่ยมชมเว็บเพจ A และต้องการเปลี่ยนไปเว็บเพจอื่น โอกาสที่ลิงค์แต่ละลิงค์ในเว็บเพจ A จะถูกผู้เยี่ยมชมคลิกจะเป็น $\frac{1}{20}$

อย่างไรก็ตามในปัจจุบันบริษัทกูเกิลใช้สมการที่ (2.2) มาใช้ในการคำนวณค่าเพจแรงค์ ซึ่งเป็นสมการที่ปรับปรุงใหม่ของบริษัทกูเกิล [3] เพื่อให้ถูกต้องตามหลักการของความน่าจะเป็น

เนื่องจากแรนดอมเซอร์เฟอร์โมเดลเป็นแนวคิดที่เกิดจากความน่าจะเป็นในการเปลี่ยนสถานะ (Transition Probability) ที่แก้ปัญหาโดยใช้กฎลูกโซ่มาร์คอฟ (Markov Chain Rule) ผลรวมของค่าเว็บเพจทั้งหมดในเว็บลิงค์กราฟควรจะเป็น 1 ดังนั้นกูเกิล จึงนำสมการที่ (2.1) มาปรับปรุงใหม่ให้กลายเป็นสมการที่ (2.2) โดยสร้างพจน์ $\frac{(1-d)}{N}$ แทนที่พจน์ของ (1-d)

เมื่อต้องการนำสมการที่ (2.1) หรือสมการที่ (2.2) ไปคำนวณค่าเพจแรงค์จะทำได้ด้วยการคำนวณแบบขั้นตอนวิธีการทำซ้ำที่มีผลลัพธ์ลู่เข้าหาค่าใดค่าหนึ่ง ซึ่งในแต่ละรอบของการทำซ้ำค่าเพจแรงค์จะถูกปรับให้เข้าใกล้ค่าที่เหมาะสมที่สุด ซึ่งจำนวนรอบของการทำซ้ำจะขึ้นอยู่กับขนาดของเว็บลิงค์กราฟ โครงสร้างของเว็บลิงค์ และการเซตค่าเพจแรงค์เริ่มต้นในแต่ละเว็บเพจ

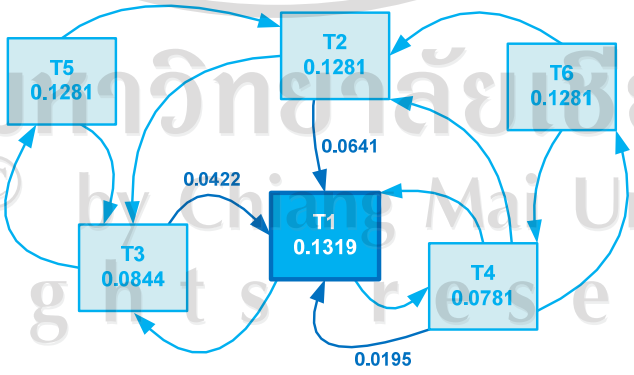
ตัวอย่างการคำนวณค่าเพจแรงค์ของเว็บลิงค์กราฟในเวลาหนึ่ง หากกำหนดให้จำนวนเว็บเพจทั้งหมดในเว็บลิงค์กราฟมีจำนวน 6 เว็บเพจ พิจารณาค่าเพจแรงค์ที่เว็บเพจ T1 ซึ่งได้รับค่าเพจแรงค์จาก เว็บเพจ T2, T3 และ T4 ดังรูปที่ 2.4 ซึ่งค่าเพจแรงค์ของเว็บเพจ T1 ได้จากการแทนค่าในสมการที่ (2.2) ดังแสดงวิธีการคำนวณดังต่อไปนี้

$$PR_{T1} = \left(\frac{1-d}{N}\right) + d\sum_{i=1}^{N}\left(\frac{PR(Ti)}{Out(Ti)}\right)$$

$$= \left(\frac{1-0.85}{6}\right) + 0.85\left(\left(\frac{PR(T2)}{2}\right) + \left(\frac{PR(T3)}{2}\right) + \left(\frac{PR(T4)}{4}\right)\right)$$

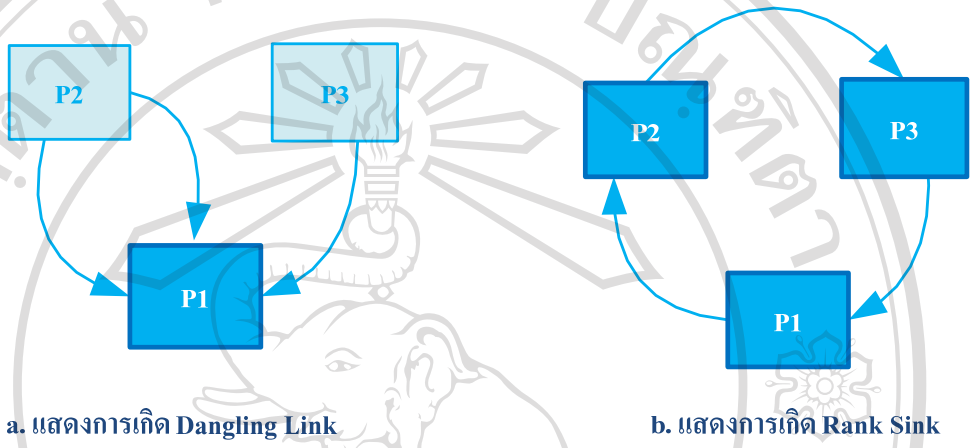
$$= 0.0250 + 0.85\ (0.0641+0.0422+0.0195)$$

$$= 0.1319$$



รูปที่ 2.4 แสดงตัวอย่างค่าเพจแรงค์ของ T1

สำหรับค่าคงที่ d เป็นค่าคงที่สำหรับสร้างความสมดุลให้สมการในกรณีการเกิด Dangling Link และ Rank Sink ซึ่งทั้งสองกรณีนี้จะส่งผลกระทบต่อค่าเพจแรงค์ในแต่ละรอบของการคำนวณ ทำให้การจัดลำดับเว็บเพจมีความผิดพลาดได้ ดังแสดงตัวอย่างในรูปที่ 2.5 แสดงตัวอย่างการเกิด Dangling Link และ Rank Sink



รูปที่ 2.5 ตัวอย่างการเกิด Dangling Link และ Rank Sink

จากรูปที่ 2.5a แสดงการเกิด Dangling Link ที่เว็บเพจ P1 ซึ่งเป็นเว็บเพจที่มีแต่ลิงค์เข้า ไม่มีลิงค์ออก ในกรณีดังกล่าวจะทำให้เกิดความผิดพลาดของค่าเพจแรงค์ในแต่ละรอบของการคำนวณซ้ำ เนื่องจากที่เว็บเพจ P1 จะไม่มีกระจายค่าเพจแรงค์ต่อไปยังเว็บเพจอื่นๆ ทำให้ค่าเพจแรงค์มีการสูญหายในระหว่างการคำนวณในแต่ละรอบของการคำนวณซ้ำ และส่งผลกระทบต่อการจัดลำดับเว็บเพจในที่สุด

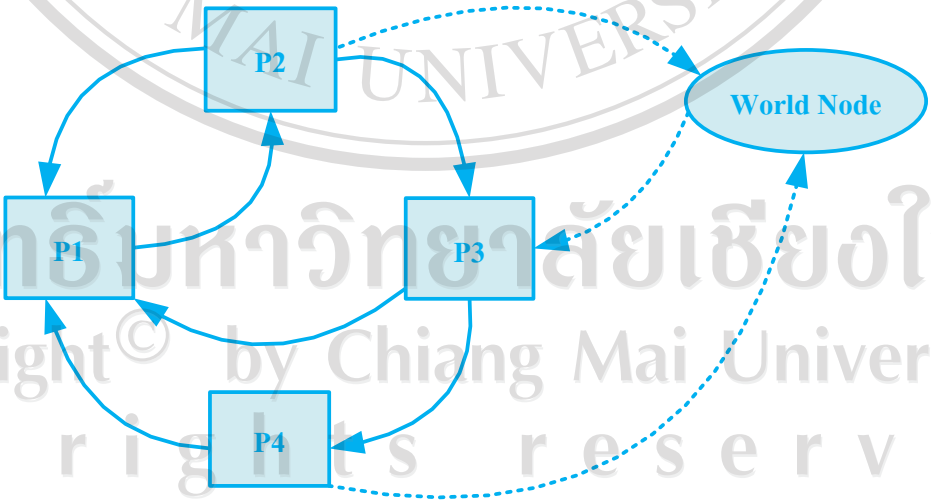
ในทำนองเดียวกัน ผลกระทบต่อการจัดลำดับเว็บเพจ ที่เกิดจากความผิดพลาดของค่าเพจแรงค์ในแต่ละรอบของการคำนวณซ้ำ สามารถเกิดขึ้นจากกรณีของ Rank Sink ได้เช่นเดียวกัน ซึ่งกรณีดังกล่าว เป็นการเชื่อมโยงของเว็บลิงค์ในลักษณะของการวนกลับมาที่เว็บเพจเดิม ดังแสดงตัวอย่างดังรูปที่ 2.5b จะเห็นได้ว่า จากเว็บเพจ P1 มีลิงค์ออกไปยังเว็บเพจ P2 และจากเว็บเพจ P2 มีลิงค์ออกไปยังเว็บเพจ P3 และจากเว็บเพจ P3 มีลิงค์ออกไปยังเว็บเพจ P1 ซึ่งถือว่าเป็นการเชื่อมโยงกลับมาตำแหน่งเดิม ปัญหาของ Rank Sink นี้จะทำให้ค่าเพจแรงค์เพิ่มขึ้นเรื่อยๆ

2.3 ขั้นตอนวิธี JXP (JXP Algorithm)

จากงานวิจัย [2, 4] เป็นแนวคิดของการจัดลำดับเว็บเพจบนเครือข่ายเพียร์ทูเพียร์ โดยเสนอขั้นตอนวิธี JXP สำหรับพิจารณาค่าน้ำหนักของเว็บเพจ (Page Weight) บนโลคอลเว็บลิงค์กราฟ (Local Web Link Graph) ที่ถูกเก็บไว้ในแต่ละเพียร์ ให้เสมือนว่ากำลังพิจารณาค่าน้ำหนักของเว็บเพจบนโกลบอลเว็บลิงค์กราฟ (Global Web Link Graph)

เนื่องจากเป็นการประมวลผลบนระบบกระจายศูนย์ ทำให้ขั้นตอนวิธี JXP มีจุดเด่นในด้านประสิทธิภาพของเวลาในการคำนวณค่าเพจแรงค์ และการใช้ทรัพยากรที่มีอยู่จำกัดอย่างมีประสิทธิภาพ ทั้งด้านการประมวลผลและจัดเก็บข้อมูล เมื่อเว็บลิงค์กราฟมีขนาดใหญ่มากขึ้น ขั้นตอนวิธี JXP ถือเป็นทางเลือกที่ได้เปรียบสำหรับการจัดลำดับเว็บเพจ เมื่อเทียบกับการประมวลผลบนระบบแบบรวมศูนย์ อย่างไรก็ตามขั้นตอนวิธีดังกล่าวจะต้องทำการคำนวณค่าเพจแรงค์ใหม่ทุกครั้งเมื่อโครงสร้างของเว็บลิงค์กราฟในแต่ละเพียร์ถูกเปลี่ยนแปลง

สำหรับในขั้นตอนวิธี JXP จะมีการกำหนดเวิลด์โหนด (World Node) เพิ่มขึ้นมา ซึ่งเป็นโหนดพิเศษที่ใช้เก็บเว็บเพจที่อยู่ภายนอกเพียร์ (External Pages) ทั้งหมดบนเครือข่ายเพียร์ทูเพียร์ โดยเวิลด์โหนดจะมีลิงค์เชื่อมโยงระหว่างเว็บเพจที่อยู่ภายนอกเพียร์ และเว็บเพจที่อยู่ภายในเพียร์ (Internal Pages) ดังรูปที่ 2.6 แสดงตัวอย่างการเชื่อมต่อโลคอลเว็บลิงค์กราฟและเวิลด์โหนด



รูปที่ 2.6 ตัวอย่างการเชื่อมต่อโลคอลเว็บลิงค์กราฟและเวิลด์โหนด

ค่าน้ำหนักของเวิลด์โหนดสามารถคำนวณได้จากผลรวมของค่าน้ำหนักของเว็บเพจที่อยู่นอกเพียร์ทั้งหมด ดังสมการที่ (2.3)

$$\text{PageWeight(W)} = \sum_{i \in External}(PageWeight(i)) = 1 - \sum_{i \in Internal}(PageWeight(i)) \quad \ldots (2.3)$$

และค่าน้ำหนักของลิงค์จากเวิลด์โหนดที่ให้ค่าเว็บเพจที่อยู่ภายในเพียร์ สามารถคำนวณได้จากสมการที่ (2.4)

$$\text{PageWeight(W-->a)} = \frac{1}{PageWeight(W)}\left[\sum_{i \in know \& i \rightarrow a} \frac{PageWeight(i)}{Out(i)}\right] \quad \ldots (2.4)$$

โดย

Internal = $\{int_1, int_2, int_3, \ldots, int_n\}$ คือเซตของเว็บเพจที่อยู่ภายในเพียร์

External = $\{ext_1, ext_2, ext_3, \ldots, ext_{(N-n)}\}$ คือเซตของเว็บเพจที่อยู่ภายนอกเพียร์

know คือซับเซตของ External ที่มีลิงค์เชื่อมโยงไปยังเว็บเพจ a

a คือ เว็บเพจที่เป็นซับเซตของ Internal ที่มีลิงค์เข้ามาจาก know

ซึ่งขั้นตอนวิธี JXP สามารถแบ่งการทำงานของขั้นตอนวิธีออกเป็น 2 ส่วนคือ การคำนวณค่าน้ำหนักของเว็บเพจบนโลคอลเว็บลิงค์กราฟ (Local PageWeight Computation) และการจับคู่ระหว่างเพียร์ (Peer Meeting) ซึ่งการทำงานในแต่ละส่วนมีรายละเอียดของขั้นตอนดังต่อไปนี้

2.3.1 การคำนวณค่าน้ำหนักของเว็บเพจบนโลคอลเว็บลิงค์กราฟในแต่ละเพียร์

ก่อนการคำนวณค่าน้ำหนักของเว็บเพจบนโลคอลเว็บลิงค์กราฟ จะต้องมีการกำหนดค่าเริ่มต้นของตัวแปรต่างๆ โดยใช้ขั้นตอนวิธีในรูปที่ 2.7 แสดงขั้นตอนวิธีการคำนวณค่าน้ำหนักของเว็บเพจบนโลคอลเว็บลิงค์กราฟในแต่ละเพียร์

Algorithm1 Initial PageWeight Computation

1: input: local graph G and est. size of global graph N

2: n <-- size(G)

3: Create world node W

4: PageWeight(p|p $\in$ G) <-- $\frac{1}{N}$

5: PageWeight(W) <-- $\frac{N-n}{N}$

6: add W to G

7: run PageRank power iteration on G

8: Update(PageWeights)

รูปที่ 2.7 ขั้นตอนวิธีการคำนวณค่าน้ำหนักของเว็บเพจบนโลคอลเว็บลิงค์กราฟ

จากขั้นตอนวิธีในรูปที่ 2.7 มีรายละเอียดการทำงานดังต่อไปนี้

บรรทัดที่ 1 กำหนดให้โลคอลเว็บลิงค์กราฟ G และ ขนาดของโกลบอลเว็บลิงค์กราฟ N เป็นข้อมูลเข้าของขั้นตอนวิธี

บรรทัดที่ 2 กำหนดให้ n คือขนาดของโลคอลเว็บลิงค์กราฟ G

บรรทัดที่ 3 สร้างเวิลด์โหนดซึ่งเป็นโหนดที่เก็บเว็บเพจที่อยู่ภายนอกเพียร์

บรรทัดที่ 4,5 กำหนดค่าเริ่มต้นของค่าน้ำหนักของเว็บเพจให้กับแต่ละเว็บเพจโดยเว็บเพจที่อยู่ภายในเพียร์เป็น $\frac{1}{N}$ และค่าน้ำหนักของเวิลด์โหนดเป็น $\frac{N-n}{N}$

บรรทัดที่ 6 เชื่อมต่อ เวิลด์โหนดกับโลคอลเว็บลิงค์กราฟ G

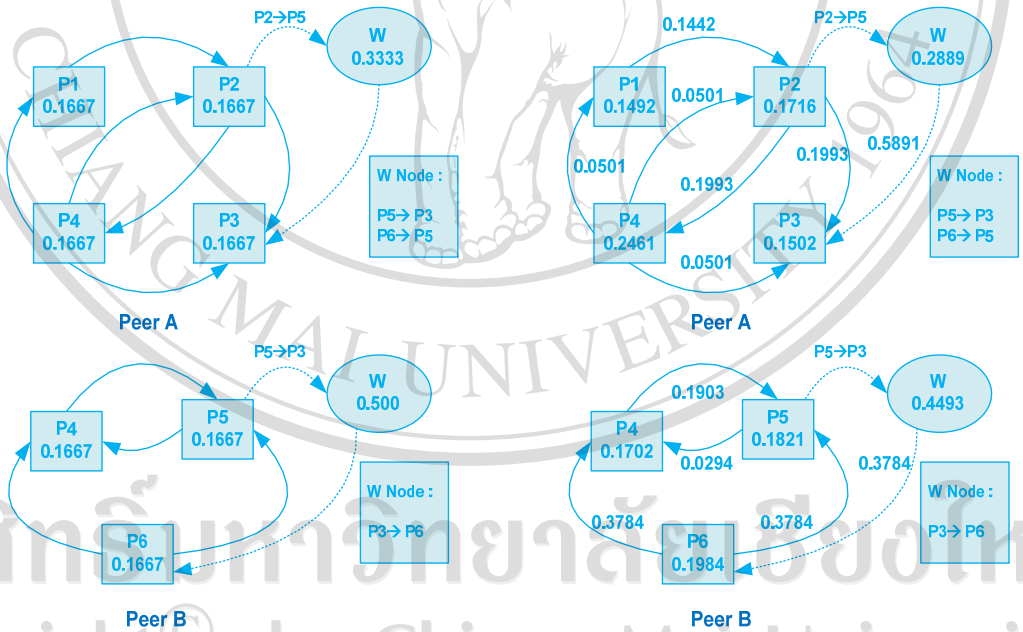
บรรทัดที่ 7 คำนวณค่าน้ำหนักของเว็บเพจเพื่อปรับค่าน้ำหนักของเว็บเพจในโลคอลเว็บลิงค์กราฟ G ให้เข้าใกล้ค่าที่เหมาะสม

บรรทัดที่ 8 บันทึกค่า

จากการทำงานของขั้นตอนวิธีในรูปที่ 2.7 ที่กล่าวมา สามารถแสดงตัวอย่างให้เห็นดังรูปที่ 2.8 แสดงโลคอลเว็บลิงค์กราฟก่อนและหลังการปรับค่าน้ำหนักของเว็บเพจให้เข้าใกล้ค่าที่เหมาะสม โดยกำหนดให้ขนาดของโกลบอลเว็บลิงค์กราฟมีโหนดทั้งหมด 6 โหนด และมีเพียร์ทั้งหมดในเครือข่าย 2 เพียร์คือ เพียร์ A และ เพียร์ B

รูปที่ 2.8a คือโลคอลเว็บลิงค์กราฟเริ่มต้น ซึ่งแต่ละเว็บเพจที่ถูกเก็บไว้ในเพียร์ A และ เพียร์ B ถูกเซตค่าเริ่มต้นเป็น 0.1667 เท่ากันหมด และเวิลด์โหนดของเพียร์ A ถูกกำหนดให้เป็น 0.3333 และเวิลด์โหนดของเพียร์ B ถูกกำหนดให้เป็น 0.5000 ตามลำดับ

หลังจากใช้สูตรของการคำนวณค่าน้ำหนักของเว็บเพจ ในสมการที่ (2.2) คำนวณซ้ำไปเรื่อยๆ จนค่าน้ำหนักของเว็บเพจเข้าใกล้ค่าที่เหมาะสม การคำนวณจะสิ้นสุดลง ซึ่งแสดงผลดังรูปที่ 2.8b ซึ่งเป็นค่าน้ำหนักของเว็บเพจในโลคอลเว็บลิงค์กราฟ หลังจากปรับค่าน้ำหนักของเว็บเพจให้เข้าใกล้ค่าที่เหมาะสมเรียบร้อยแล้ว



a.โลคอลเว็บลิงค์กราฟเริ่มต้น b.โลคอลเว็บลิงค์กราฟหลังจากปรับค่าน้ำหนักของเว็บเพจ

รูปที่ 2.8 แสดงโลคอลเว็บลิงค์กราฟก่อนและหลังการปรับค่าน้ำหนักของเว็บเพจ

อย่างไรก็ตามการทำงานที่เป็นอิสระต่อกันของเว็บคลอร์เลอร์บนเครือข่ายเพียร์ทูเพียร์ส่งผลให้สามารถเก็บข้อมูลเว็บเพจหนึ่งเว็บเพจไว้ในเพียร์มากกว่าหนึ่งเพียร์ได้ การคำนวณน้ำหนักของเว็บเพจเฉพาะบนโลคอลเว็บลิงค์กราฟ ทำให้น้ำหนักของเว็บเพจเดียวกันที่ถูกเก็บไว้ต่างเพียร์มี

ค่าไม่เท่ากัน และมีผลกระทบต่อการจัดลำดับของเว็บเพจ จึงต้องมีขั้นตอนวิธีการจับคู่ระหว่างเพียร์ ในหัวข้อที่ 2.3.2 ต่อไป

2.3.2 การจับคู่ระหว่างเพียร์

หลังจากคำนวณน้ำหนักของเว็บเพจบนโลคอลเว็บลิงค์กราฟเรียบร้อยแล้ว จากนั้นจะเป็นการจับคู่ระหว่างเพียร์ เพื่อทำให้เว็บเพจเดียวกันที่ถูกเก็บไว้ต่างเพียร์มีค่าเท่ากัน โดยมีขั้นตอนวิธีดังรูปที่ 2.9 แสดงขั้นตอนวิธีการจับคู่ระหว่างเพียร์

Algorithm2 the JXP Algorithm

1: input: local graph G_A, world node W_A, score list L_A

2: repeat

3: Contact a random peer B in the network and exchange information

4: G_M <-- mergeGraphs(G_A,G_B)

5: W_M <-- mergeWorldNodes(W_A,W_B)

6: G'_M <-- ($G_M + W_M$)

7: L_M <-- combineLists(L_A,L_B)

8: PR <-- PageRank(G'_M)

9: L'_M <-- updateScoresList(L_M, PR)

10: update(L_A)

11: update(W_A)

12: Discard(G_M,W_M,L_M,G_B,W_B,L_B)

รูปที่ 2.9 แสดงขั้นตอนวิธีการจับคู่ระหว่างเพียร์

จากขั้นตอนวิธีในรูปที่ 2.9 มีกระบวนการทำงานดังนี้

บรรทัดที่ 1 เลือกเพียร์ขึ้นมา 1 เพียร์สำหรับเป็นเพียร์หลักในการจับคู่ ให้ชื่อว่าโลคอลเว็บลิงค์กราฟ G_A

บรรทัดที่ 3 สุ่มเพียร์ที่เหลือขึ้นมา 1 เพียร์ให้ชื่อว่าว่าโลคอลเว็บลิงค์กราฟ G_B เพื่อนำมาจับคู่กับว่าโลคอลเว็บลิงค์กราฟ G_A

บรรทัดที่ 4,5,6,7 รวมโลคอลเว็บลิงค์กราฟ G_A และ โลคอลเว็บลิงค์กราฟ G_B เข้าด้วยกัน ให้ชื่อเว็บลิงค์กราฟกราฟใหม่นี้ว่าเว็บลิงค์กราฟ G'_M

บรรทัดที่ 8 คำนวณค่าน้ำหนักของเว็บเพจที่เว็บลิงค์กราฟ G'_M ให้เข้าใกล้ค่าที่เหมาะสม

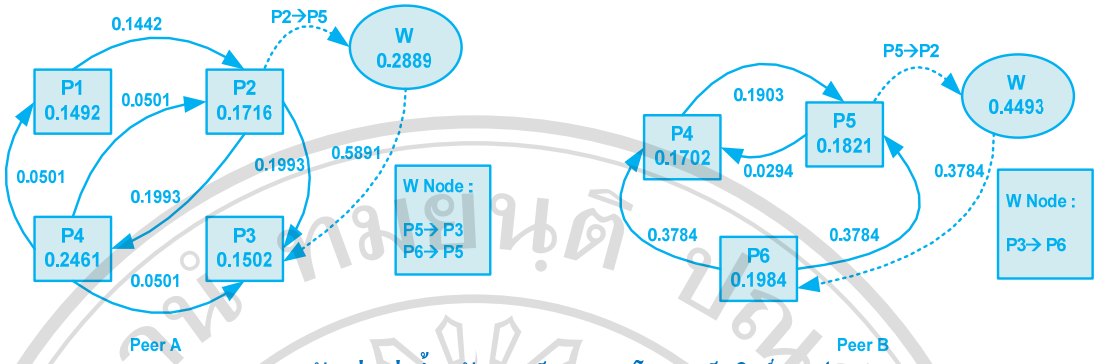
บรรทัดที่ 9, 10, 11 บันทึกข้อมูล

บรรทัดที่ 12 แยกเว็บลิงค์กราฟ G'_M ให้เป็นโลคอลเว็บลิงค์กราฟ G_A และโลคอลเว็บลิงค์กราฟ G_B

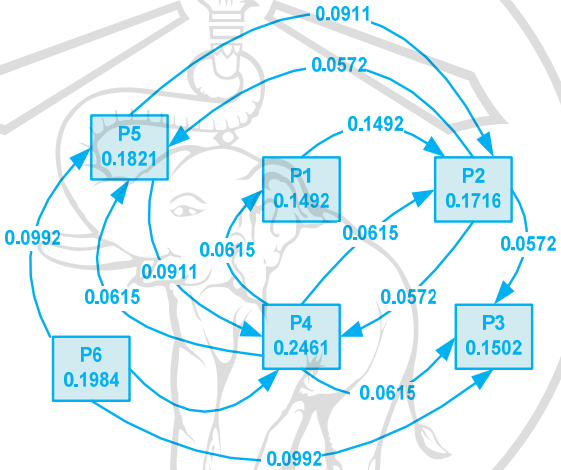
ทำซ้ำจนกระทั่งจับคู่ครบทุกเพียร์

จากการทำงานของขั้นตอนวิธีในรูปที่ 2.9 จะทำให้ค่าน้ำหนักของเว็บเพจเดียวกันที่ถูกเก็บไว้ในต่างเพียร์มีค่าเท่ากัน ซึ่งขั้นตอนดังกล่าวจะทำให้เสมือนว่ากำลังพิจารณาค่าน้ำหนักของเว็บเพจบนโกลบอลเว็บลิงค์กราฟ ดังแสดงตัวอย่างในรูปที่ 2.10 ซึ่งเป็นผลที่ได้จากขั้นตอนวิธีการจับคู่ระหว่างเพียร์

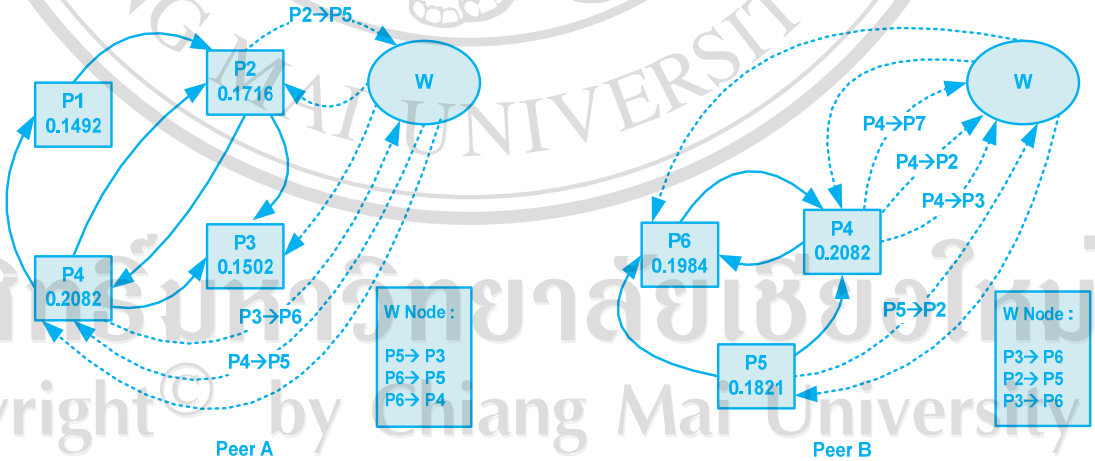
จากรูปที่ 2.10a แสดงน้ำหนักของเว็บเพจบนโลคอลเว็บลิงค์กราฟ จะเห็นได้ว่าที่เว็บเพจ P4 ของเพียร์ A และ เพียร์ B ซึ่งเป็นเว็บเพจเดียวกัน มีค่าน้ำหนักไม่เท่ากัน เมื่อทำการจับคู่ระหว่างเพียร์แล้ว โลคอลเว็บลิงค์กราฟที่ถูกเก็บไว้ในเพียร์ A และเพียร์ B ถูกนำมารวมกันเป็นเว็บลิงค์กราฟใหม่ และเริ่มคำนวณน้ำหนักของเว็บเพจในกราฟใหม่นี้อีกครั้ง ดังแสดงในรูปที่ 2.10b เมื่อค่าน้ำหนักของเว็บเพจเข้าใกล้ค่าที่เหมาะสมแล้ว กราฟดังกล่าวจะถูกแยกออกเป็นโลคอลเว็บลิงค์กราฟเดิม ก่อนการรวมกราฟ ดังแสดงในรูปที่ 2.10c จะเห็นได้ว่า ที่เว็บเพจ P4 ของทั้งเพียร์ A และ เพียร์ B มีค่าเท่ากัน



a. แสดงตัวอย่างค่าน้ำหนักของเว็บเพจบนโลคอลเว็บลิงค์กราฟ



b. แสดงตัวอย่างเว็บลิงค์กราฟที่เกิดจากการรวมเพียร์ A และเพียร์ B เข้าด้วยกัน



c. แสดงตัวอย่างค่าน้ำหนักของเว็บเพจบนโลคอลเว็บลิงค์กราฟหลังจาผ่านการจับคู่ระหว่างเพียร์

รูปที่ 2.10 แสดงขั้นตอนวิธีการจับคู่ระหว่างเพียร์

2.4 ขั้นตอนวิธี IPR (IPR Algorithm)

แนวคิดของงานวิจัย [5] เป็นการจัดลำดับเว็บเพจโดยพิจารณาเฉพาะส่วนที่เพิ่มเข้ามาใหม่ของเว็บลิงค์กราฟ โดยได้เสนอขั้นตอนวิธี IPR สำหรับแบ่งโหนดของกราฟออกเป็น 2 ส่วน คือ ส่วนแรกเป็นส่วนโหนดของเว็บลิงค์กราฟที่ไม่เปลี่ยนแปลง และส่วนที่สองเป็นส่วนโหนดของเว็บลิงค์กราฟที่จะนำมาคำนวณน้ำหนักของเว็บเพจ โดยขั้นตอนวิธี IPR จะคำนวณค่าน้ำหนักของเว็บเพจเฉพาะส่วนที่สองเท่านั้น ดังแสดงขั้นตอนวิธี IPR ในรูปที่ 2.11

Algorithm IPR(G , V_u, V_c)

1 Initialize the list V_Q

2 Pop a Vertex N from V_c

3 For all the children of N

4 if children of N list V_u

5 Remove them from V_u

6 Push them in V_c

7 Push N in V_Q and repeat step 2 till queue V_c is empty

8 For each element in list V_u

9 Take the element and scale the previous pagerank value to get new pagerank value.

10 Look up whether any of the children, element of V_u belong to V_Q, if so remove element of V_u, copy it in V_b.

11 Scale Border Nodes in V_b for stochastic property

12 Perform Original PageRank (V_Q U V_b)

รูปที่ 2.11 แสดงขั้นตอนวิธี IPR

จากรูปที่ 2.11 แสดงขั้นตอนวิธี IPR ซึ่งรับข้อมูลเข้าเป็นเว็บลิงค์กราฟ G ลิสต์ (List) ของโหนดในเว็บลิงค์กราฟที่ไม่เปลี่ยนแปลง (V_u) และลิสต์ของโหนดในเว็บลิงค์กราฟที่เปลี่ยนแปลงจากเดิม (V_c) อธิบายรายละเอียดการทำงานของขั้นตอนวิธี IPR ได้ดังต่อไปนี้

บรรทัดที่ 1 เป็นขั้นตอนวิธีการสร้างลิสต์ V_Q สำหรับเก็บโหนดของเว็บลิงค์กราฟที่จะนำมาคำนวณค่าน้ำหนักของเว็บเพจ

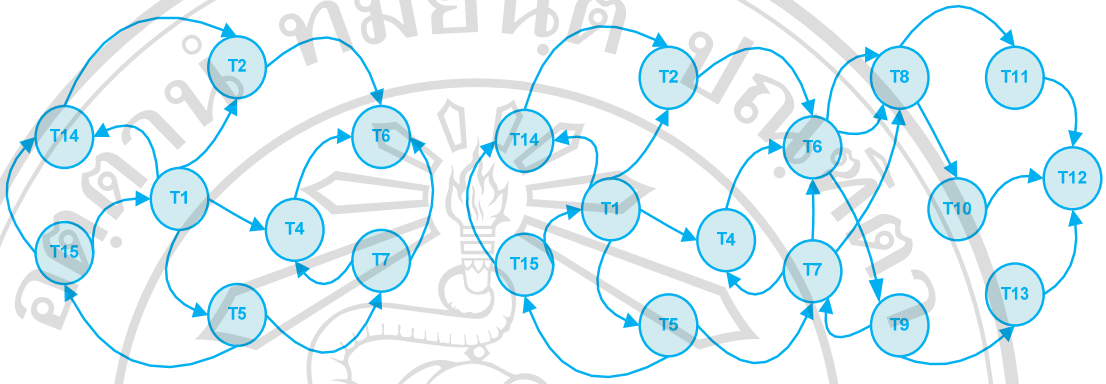
บรรทัดที่ 2-7 เป็นขั้นตอนวิธีการเลือกโหนดของเว็บลิงค์กราฟ G จาก ลิสต์ของโหนดใน V_u และ ลิสต์ของโหนดใน V_c มาเก็บไว้ในลิสต์ V_Q โดยทำการเลือกโหนดจาก V_c ทีละโหนดไปเก็บไว้ในลิสต์ V_Q หากโหนดลูก (Child Node) ของ V_c เป็นโหนดในลิสต์ V_u แล้วให้นำโหนดลูกนั้น มาเก็บไว้ในลิสต์ V_c การทำงานของ Step2 นี้จะหยุดลงเมื่อลิสต์ของ V_c ไม่มีโหนดเหลืออยู่ เมื่อทำขั้นตอนวิธีใน Step2 เสร็จสิ้นแล้ว จะได้ลิสต์ของโหนดในเว็บลิงค์กราฟสองส่วนคือส่วนลิสต์ของโหนดในเว็บลิงค์กราฟที่ไม่เปลี่ยนแปลง และส่วนลิสต์ของโหนดในเว็บลิงค์กราฟที่จะนำมาคำนวณค่าน้ำหนักของเว็บเพจ

บรรทัดที่ 8-10 เป็นขั้นตอนวิธีการหาโหนดขอบ (Border Nodes) ที่แบ่งระหว่างส่วนของโหนดในเว็บลิงค์กราฟที่ไม่เปลี่ยนแปลง กับส่วนของโหนดในกราฟที่จะนำมาคำนวณค่าน้ำหนักของเว็บเพจ โดยโหนดขอบหาได้จาก ทุกๆ โหนดในลิสต์ของโหนดที่ไม่เปลี่ยนแปลงที่มีลิงค์ออกไปหาโหนดในลิสต์ของโหนดที่จะนำมาคำนวณค่าน้ำหนักของเว็บเพจ โหนดขอบเป็นโหนดที่มีผลต่อการคำนวณค่าน้ำหนักของเว็บเพจ เนื่องจากเป็นโหนดที่ให้ค่าน้ำหนักของเว็บเพจในส่วนของโหนดที่ไม่เปลี่ยนแปลง กับส่วนของโหนดที่จะนำมาคำนวณค่าน้ำหนักของเว็บเพจ

บรรทัดที่ 11-12 เป็นขั้นตอนวิธีการคำนวณค่าน้ำหนักของเว็บเพจในเว็บลิงค์กราฟ G โดยคำนวณค่า น้ำหนักของเว็บเพจเฉพาะลิสต์ของโหนดที่จะนำมาคำนวณค่าน้ำหนักของเว็บเพจ และโหนดขอบเท่านั้น

ดังนั้นข้อได้เปรียบของขั้นตอนวิธี IPR เมื่อโครงสร้างเว็บลิงค์กราฟมีการเปลี่ยนแปลงไปขั้นตอนวิธี IPR ไม่จำเป็นต้องคำนวณค่าน้ำหนักเว็บเพจในทุกๆ โหนดของเว็บลิงค์กราฟ ทำให้เพิ่มประสิทธิภาพของเวลาการคำนวณค่าน้ำหนักของเว็บเพจให้มากขึ้น

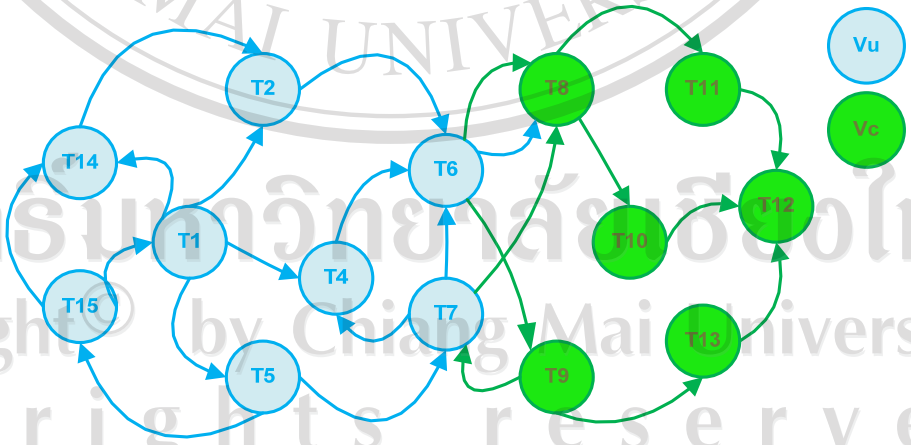
ตัวอย่างการทำงานของขั้นตอนวิธี IPR กำหนดให้เว็บลิงค์กราฟ G เป็นเว็บลิงค์กราฟที่ต้องการพิจารณาค่าน้ำหนักของเว็บเพจ G_1 เป็นเว็บลิงค์กราฟเมื่อเวลา T_1 และ G_2 เป็นเว็บลิงค์กราฟเมื่อเวลา T_2 ซึ่งเกิดจากโครงสร้างที่เปลี่ยนแปลงไปของเว็บลิงค์กราฟ T_1 ดังแสดงในรูปที่ 2.12



a. เว็บลิงค์กราฟเมื่อเวลา T1 b. เว็บลิงค์กราฟเมื่อเวลา T2

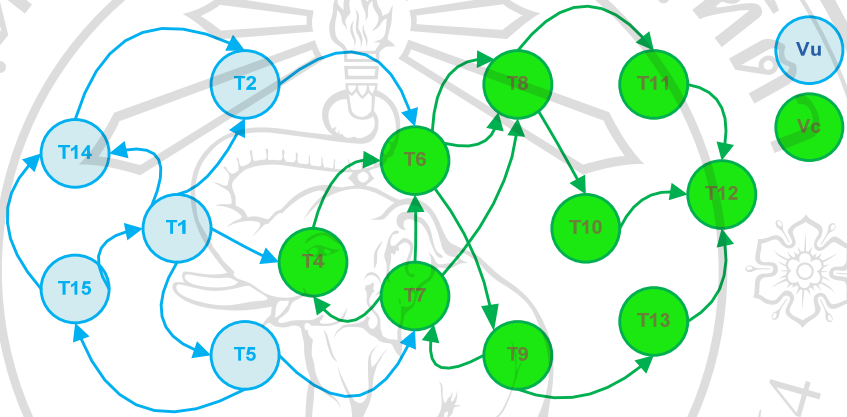
รูปที่ 2.12 เว็บลิงค์กราฟที่เวลา T_1 และ T_2

จากนั้นหาส่วนต่างของเว็บลิงค์กราฟโดยการเปรียบเทียบเว็บลิงค์กราฟ G_2 และ G_1 เพื่อหาโครงสร้างของเว็บลิงค์กราฟที่เปลี่ยนแปลงไปจากเดิม ซึ่งการเปรียบเทียบดังกล่าวจะสามารถแบ่งโหนดในเว็บลิงค์กราฟ G_2 ออกเป็น 2 ส่วน คือ ส่วนของโหนดที่ไม่เปลี่ยนแปลงแทนด้วย V_u และส่วนของโหนดที่เปลี่ยนแปลงแทนด้วย V_c ดังแสดงในรูปที่ 2.13



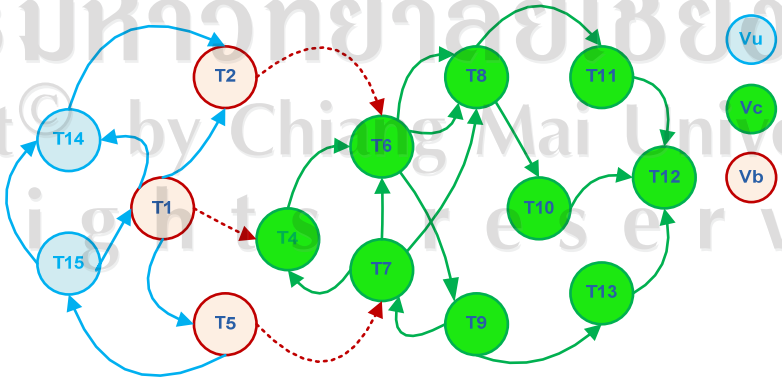
รูปที่ 2.13 เว็บลิงค์กราฟ G_2 ที่ถูกแบ่งกราฟเป็น 2 ส่วน

เมื่อแบ่งเว็บลิงค์กราฟ G_2 ออกเป็น 2 ส่วน คือส่วนลิสต์ของโหนดที่ไม่เปลี่ยนแปลง V_u และส่วนลิสต์ของโหนดที่เปลี่ยนแปลง V_c จากนั้นนำทั้งสองส่วนไปเป็นข้อมูลเข้าสำหรับขั้นตอนวิธี IPR ซึ่งในขั้นแรกเว็บลิงค์กราฟ G_2 จะถูกแบ่งลิสต์ของโหนดในเว็บลิงค์กราฟใหม่ เป็นส่วนที่ไม่เปลี่ยนแปลง และส่วนที่จะนำมาคำนวณค่าน้ำหนักของเว็บเพจ ดังแสดงในรูปที่ 2.14 ซึ่งที่โหนด T_1, T_2, T_5, T_{14}, T_{15} คือโหนดของกราฟที่ไม่เปลี่ยนแปลง และที่โหนด T_4, T_6, T_7, T_8, T_9, T_{10}, T_{11}, T_{12}, T_{13} คือส่วนที่จะนำมาคำนวณค่าน้ำหนักของเว็บเพจ



รูปที่ 2.14 แสดงเว็บลิงค์กราฟ G_2 ที่ถูกแบ่งเป็น 2 ส่วนหลังผ่านขั้นตอนวิธี IPR

จากนั้นขั้นตอนวิธี IPR จะหาโหนดขอบของกราฟ G_2 และจะคำนวณค่าน้ำหนักของเว็บเพจเฉพาะโหนดขอบในเว็บลิงค์กราฟ และโหนดที่จะนำมาคำนวณน้ำหนักของเว็บเพจเท่านั้น ดังแสดงในรูปที่ 2.15 จะคำนวณค่าน้ำหนักของเว็บเพจที่โหนด T_4, T_6, T_7, T_8, T_9, T_{10}, T_{11}, T_{12}, T_1, T_2, T_5 เท่านั้น

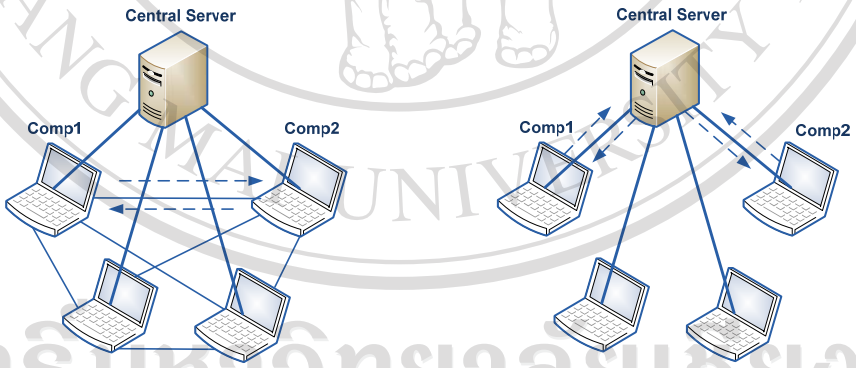


รูปที่ 2.15 เว็บลิงค์กราฟ G_2 ที่ถูกแบ่งส่วนของกราฟเป็น 3 ส่วนหลังผ่านขั้นตอนวิธี IPR

จากขั้นตอนวิธี IPR ที่กล่าวมาทำให้เพิ่มประสิทธิภาพในด้านเวลาการคำนวณค่าเพจแรงค์ลงได้ เนื่องจากเว็บลิงค์กราฟที่นำมาคำนวณค่าเพจแรงค์มีขนาดเล็กลงจากเดิม อย่างไรก็ตาม ขั้นตอนวิธี IPR ยังเป็นการคำนวณค่าเพจแรงค์บนระบบแบบรวมศูนย์ ซึ่งประสิทธิภาพในด้านเวลาการคำนวณจะลดลงเมื่อขนาดเว็บลิงค์กราฟมีขนาดใหญ่ขึ้น

2.5 เครือข่ายแบบเพียร์ทูเพียร์

เพียร์ทูเพียร์เป็นเครือข่ายคอมพิวเตอร์ (Computer Network) แบบหนึ่งที่ถูกออกแบบให้ทำงานแบบระบบกระจายศูนย์ ซึ่งแต่ละเพียร์สามารถประมวลผลด้วยตัวเองและทำงานเป็นอิสระจากเพียร์อื่นๆ ทำให้ลดปัญหาในด้านการขยายขนาดของเครือข่าย และมีความคงทน (Fault Tolerant) ซึ่งในกรณีที่บางเพียร์ถูกเพิ่มหรือออกจากเครือข่ายไปจะไม่มีผลกระทบกับเครือข่ายทั้งหมด แต่ละเพียร์สามารถจำลองตัวเองให้เป็นได้ทั้งเครื่องคอมพิวเตอร์ไคลเอนต์ (Client) และเครื่องคอมพิวเตอร์เซิร์ฟเวอร์ (Server) ทำให้แลกเปลี่ยนข้อมูลกันได้โดยตรง ระหว่างเครื่องคอมพิวเตอร์ไคลเอนต์กับเครื่องคอมพิวเตอร์ไคลเอนต์ด้วยกัน ซึ่งตรงข้ามกับการแลกเปลี่ยนข้อมูลแบบไคลเอนต์-เซิร์ฟเวอร์ (Client-Server) ที่เครื่องคอมพิวเตอร์ไคลเอนต์แต่ละเครื่อง ไม่สามารถที่จะแลกเปลี่ยนข้อมูลกันโดยตรงได้ต้องผ่านคอมพิวเตอร์เซิร์ฟเวอร์ก่อน



a. การแลกเปลี่ยนข้อมูลบนเครือข่ายแบบเพียร์ทูเพียร์ b. การแลกเปลี่ยนข้อมูลบนเครือข่ายแบบไคลเอนต์-เซิร์ฟเวอร์

รูปที่ 2.16 เปรียบเทียบการแลกเปลี่ยนข้อมูลบนเครือข่ายเพียร์ทูเพียร์ และไคลเอนต์-เซิร์ฟเวอร์

จากรูปที่ 2.16 เป็นการเปรียบเทียบการแลกเปลี่ยนข้อมูลบนเครือข่ายเพียร์ทูเพียร์และไคลเอนต์-เซิร์ฟเวอร์ ซึ่งจะเห็นได้ว่าในรูปที่ 2.16a คอมพิวเตอร์ Comp1 สามารถแลกเปลี่ยนข้อมูลกับคอมพิวเตอร์ Comp2 ได้โดยตรง แต่ในรูปที่ 2.16b คอมพิวเตอร์ Comp1 และคอมพิวเตอร์

Comp2 ซึ่งเป็นเครื่องคอมพิวเตอร์บนเครือข่ายแบบไคลเอนต์-เซิร์ฟเวอร์จะแลกเปลี่ยนข้อมูลกันได้ต้องผ่านเครื่องเซิร์ฟเวอร์ก่อน

ปัจจุบันมีหลายงานวิจัยได้พัฒนาโปรโตคอล (Protocol) สำหรับเครือข่ายเพียร์ทูเพียร์ ทั้งที่มีโครงสร้างเป็นแฮชเทเบิล (Hash Table) สกิปลิสต์ (Skip List) และ ทรี (Tree) ซึ่งมีตัวอย่างดังต่อไปนี้

2.5.1 CHORD จากงานวิจัย [6] ได้พัฒนาโปรโตคอล (Protocol) CHORD ซึ่งเป็นโปรโตคอลที่ทำงานบนเครือข่ายแบบเพียร์ทูเพียร์มีโครงสร้างเป็นแฮชเทเบิลแบบกระจาย (Distribute Hash Table) โดยมีคุณสมบัติที่น่าสนใจดังต่อไปนี้

1) ระบบเครือข่ายถูกออกแบบให้เป็นเพียร์ทูเพียร์แท้ (Pure P2P) คือไม่มีเพียร์ที่เป็นเพียร์แม่ข่าย (Central Server) หรือซุปเปอร์เพียร์ (Super Peer) แต่ละเพียร์มีความสำคัญเท่ากันหมด ทำให้เครือข่ายมีความเข้มแข็ง (Robust) หากเพียร์ใดเพียร์หนึ่งใช้งานไม่ได้ จะไม่มีผลกระทบต่อเพียร์อื่นๆ

2) เมื่อเครือข่ายเกิดปัญหาการเสียของเพียร์หรือมีการเพิ่มเพียร์ใหม่เข้ามาจะไม่มีผลกระทบกับเครือข่ายโดยรวมระบบจะทำงานได้เป็นปกติเสมอ

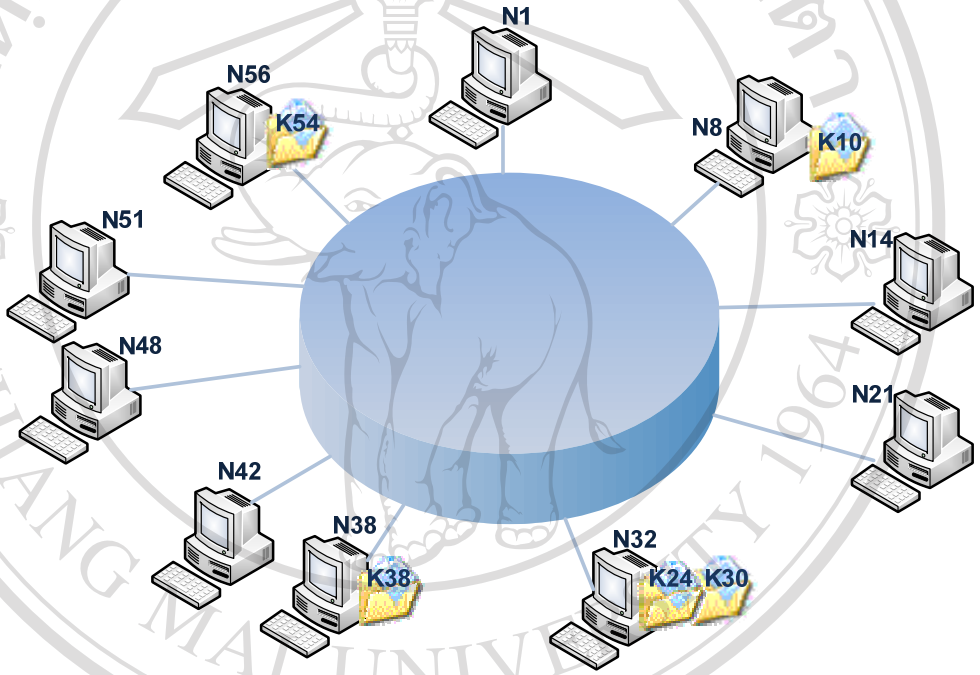
3) เมื่อเครือข่ายมีขนาดใหญ่มากขึ้น ระบบยังสามารถทำงานได้เป็นปกติ เนื่องจากเวลาในการหาค่าคีย์ (Key) เป็นแบบลอกกาลิทึม (Logarithm)

4) เนื่องจากมีโครงสร้างเป็นดีสทริบิวท์แฮชเทเบิล แฮชฟังก์ชั่น (Hash Function) จะทำหน้าที่กระจายค่าคีย์ไปเก็บไว้ในแต่ละเพียร์อย่างสมดุล

5) ไม่มีข้อกำหนดในการใช้ค่าคีย์จะใช้ค่าไหนก็ได้ ทำให้เปลี่ยนแปลงค่าคีย์ของเครือข่ายได้โดยไม่มีผลกระทบกับระบบ

สำหรับโปรโตคอลของ CHORD จะมีตัวกำหนดเอกลักษณ์ (Identifiers) เรียงต่อกันเป็นวงกลม เรียกว่าวงแหวนคอร์ด (Chord Ring) ซึ่งตัวกำหนดเอกลักษณ์ดังกล่าวจะเป็นตัวเลขตั้งแต่ 0 ถึง 2^{m-1} โดยที่ m อาจเป็นค่าคงที่ใดๆ ที่ไม่ทำให้วงแหวนคอร์ดมีขนาดเล็กเกินไปจนเกิดการชนกันของค่าคีย์สูง

หากต้องการกระจายค่าคีย์ไปเก็บไว้ในแต่ละเพียร์ แฮชฟังก์ชั่นจะสร้างตัวกำหนดเอกลักษณ์ขึ้นมาสองตัวคือ ตัวกำหนดเอกลักษณ์ของค่าคีย์ (Key Identifiers) โดยการแฮช (Hash) ค่าคีย์ที่ต้องการเก็บ และตัวกำหนดเอกลักษณ์ของโหนด (Node Identifiers) โดยการแฮชหมายเลยไอพี (IP Address) ของโหนด ซึ่งค่าคีย์จะถูกเก็บไว้ในโหนดที่มีตัวกำหนดเอกลักษณ์ของโหนดเท่ากับหรือมากกว่าตัวกำหนดเอกลักษณ์ของค่าคีย์นั้นโดยเรียกโหนดดังกล่าวว่า Successor Node ดังแสดงในรูปที่ 2.17 ตัวอย่างวงแหวนคอร์ดและการกระจายค่าคีย์ไปเก็บไว้ในแต่ละโหนดบนเครือข่ายเพียร์ทูเพียร์

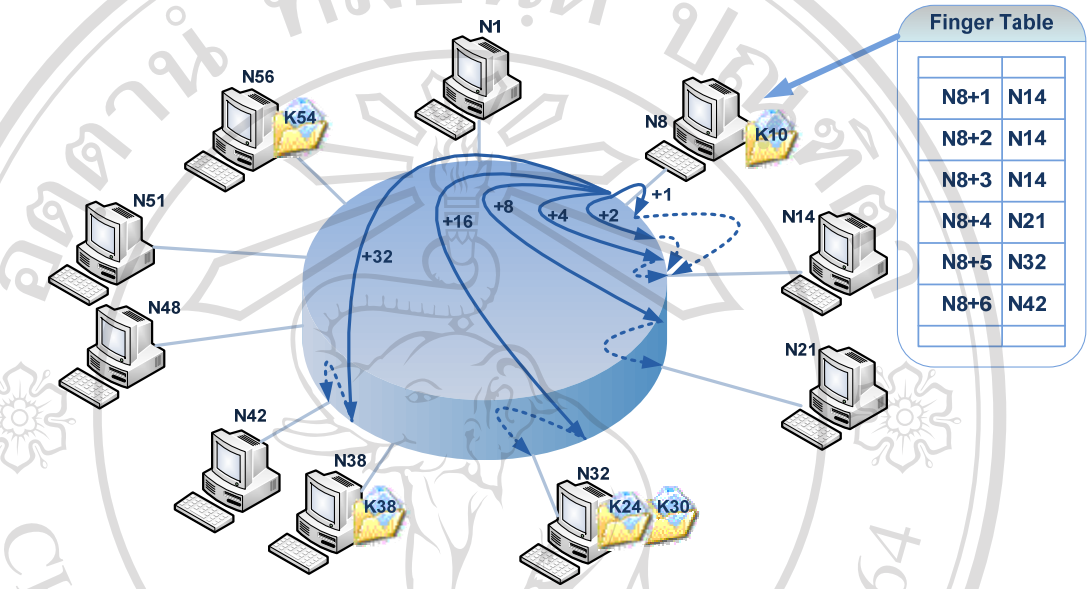


รูปที่ 2.17 เครือข่ายเพียร์ทูเพียร์ที่วงแหวนคอร์ดมีค่า m=6, 10 โหนด และ 5 คีย์

จากรูปที่ 2.17 เมื่อกำหนดให้ค่า m=6 ดังนั้นวงแหวนคอร์ดจะมีขนาด 64 บิต ซึ่งหมายถึงว่าตัวกำหนดเอกลักษณ์จะต้องมีค่าอยู่ระหว่าง 0 ถึง 63 นั่นก็คือบนเครือข่ายเพียร์ทูเพียร์นี้สามารถมีโหนดได้ทั้งหมดไม่เกิน 64 โหนด

เนื่องจากมีโหนดบนเครือข่ายอยู่เพียง 10 โหนด คือ N1, N8, N14, N21, N32, N38, N42, N48, N51 และ N56 เมื่อแต่ละโหนดถูกแฮชหมายเลขไอพีเพื่อสร้างตัวกำหนดเอกลักษณ์ของโหนด จะได้ตำแหน่งของแต่ละโหนดบนวงแหวนคอร์ดดังรูป 2.17

ในทำนองเดียวกันตัวกำหนดเอกลักษณ์ของค่าคีย์ที่ได้จากแฮชฟังค์ชัน ก็จะเป็นตัวระบุว่าแต่ละค่าคีย์มี Successor Node อยู่ตำแหน่งใดบนวงแหวนคอร์ด ดังเช่น K10 มี Successor Node อยู่ที่ N8, K24 และ K30 มี Successor Node อยู่ที่ N32, K38 มี Successor Node อยู่ที่ N38 และ K54 มี Successor Node อยู่ที่ N56



รูปที่ 2.18 ตารางเส้นทาง (Routing Table) ของโหนด N8

ในแต่ละโหนดบนเครือข่ายเพียร์ทูเพียร์ จะมีการเก็บค่าของตารางเส้นทางบนวงแหวนคอร์ด เรียกว่า Finger Table ดังตัวอย่างในรูปที่ 2.18 แสดงตารางเส้นทางของโหนด N8 โดยสามารถคำนวณหาแต่ละค่าใน Finger Table ได้จากสูตรในสมการที่ (2.5) ดังนี้

$$\text{Finger}(i) = \text{Successor}(n + 2^{i-1}) \quad \ldots (2.5)$$

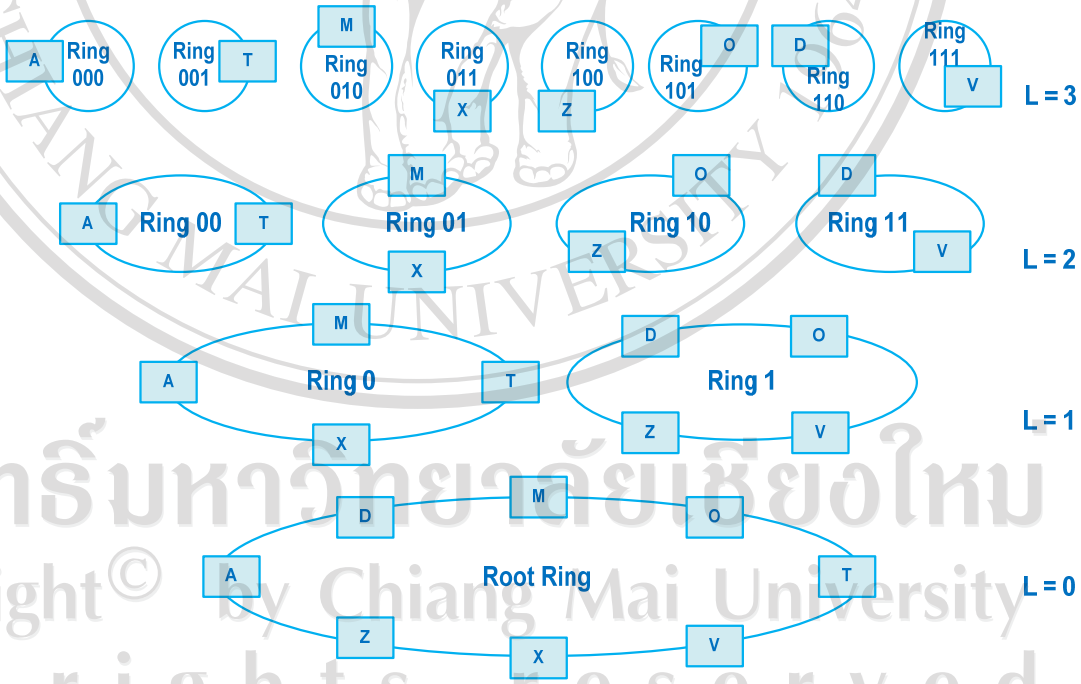
โดย n คือโหนดที่ต้องการหาค่าใน Finger Table และ i คือ Finger ของโหนด n

จะเห็นได้ว่าแต่ละโหนดจะรู้จักเพียงโหนดที่อยู่ใกล้ กับตัวมันเองในลักษณะตามเข็มนาฬิกา ไม่ได้รู้จักทุกๆ โหนดที่อยู่บนวงแหวนคอร์ด การแลกเปลี่ยนข้อมูลจำเป็นต้องทำกับโหนดข้างเคียงแทน แต่สำหรับการค้นหาค่าคีย์ที่ถูกเก็บไว้ในแต่ละโหนด ไม่ได้ใช้เวลาการค้นหาแบบลิเนียร์ (Linear) เนื่องจากมีโครงสร้างแบบแฮชเทเบิลจึงใช้เวลาเป็น O(log N)

อย่างไรก็ตามบางค่า i เมื่อแทนค่าจากสมการที่ (2.5) ทำให้ได้ตำแหน่งของ Successor ที่ไม่มีโหนดอยู่จริงบนเครือข่าย ดังตัวอย่างของ i=3 จากรูปที่ 2.18 จะได้ตำแหน่งของ Successor เป็นโหนด N11 ซึ่งไม่มีอยู่จริง หากเกิดกรณีดังกล่าวให้กำหนด Successor เป็นโหนดที่อยู่ใกล้ที่สุดซึ่งในที่นี้คือโหนด N14

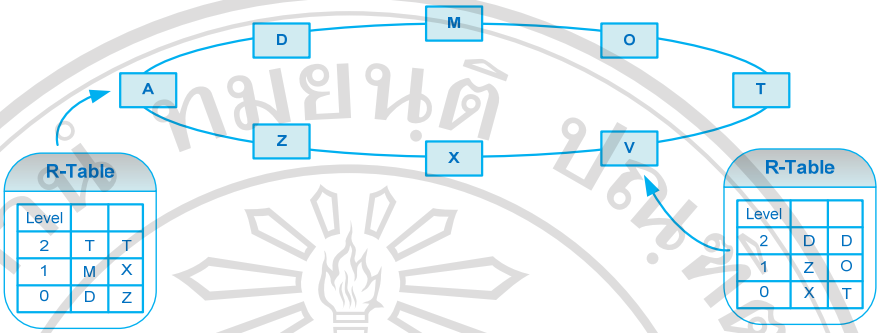
2.5.2 SkipNet จากงานวิจัย [9] ได้พัฒนาโปรโตคอล SkipNet เพื่อแก้ปัญหาการไม่มีอยู่จริงของโหนดในตารางเส้นทางบนเครือข่ายเพียร์ทูเพียร์ มีโครงสร้างเป็นสคริปลิสต์ทำให้เวลาในการค้นหาค่าคีย์เป็น O(log N)

สำหรับโครงสร้างของ SkipNet จะคล้ายๆกับวงแหวนคอร์ด แต่วงแหวน (Ring) ของ SkipNet จะแบ่งเป็นระดับ (Level) โดยในระดับที่ 0 จะเรียกว่าวงแหวนราก (Root Ring) ซึ่งจะเก็บทุกๆโหนดในเครือข่ายไว้ ส่วนวงแหวนในชั้นที่สูงขึ้นไปจะถูกแบ่งครึ่งจากวงแหวนชั้นก่อนหน้า โดยแต่ละวงแหวนที่ถูกแบ่งครึ่งจะเก็บโหนดที่ถัดจากตัวเองไป 2 โหนด ดังแสดงในรูปที่ 2.19 ตัวอย่างวงแหวนของ SkipNet ในระดับต่างๆ



รูปที่ 2.19 แสดงตัวอย่างวงแหวนของ SkipNet ในระดับต่างๆ

และในแต่ละโหนดของ SkipNet จะเก็บตารางเส้นทางที่เรียกว่า R-Table ตามระดับต่างๆ ดังแสดงในรูปที่ 2.20 ตัวอย่างตารางเส้นทางของโหนด A และ โหนด V



รูปที่ 2.20 ตัวอย่างตารางเส้นทางของโหนด A และ V

2.5.3 BATON จากงานวิจัย [7] ได้พัฒนาโปรโตคอล BATON เพื่อช่วยสนับสนุนการสอบถามแบบช่วง (Range Query) ให้มีประสิทธิภาพมากขึ้น โดย BATON จะมีโครงเป็นเป็นทรีที่มีความสมดุล (Balanced Tree Structure) เช่นเดียวกับเอวีแอลทรี (AVL Tree) หากพิจารณาการเชื่อมโยงของโหนดใน BATON จะเห็นได้ว่า ใน 1 โหนด สามารถเชื่อมโยงไปยังโหนดอื่นๆในทรีได้ 4 การเชื่อมโยง ดังรูปที่ 2.21 ตังอย่างการเชื่อมโยงของโหนด f ซึ่งมีการเชื่อมโยงดังต่อไปนี้

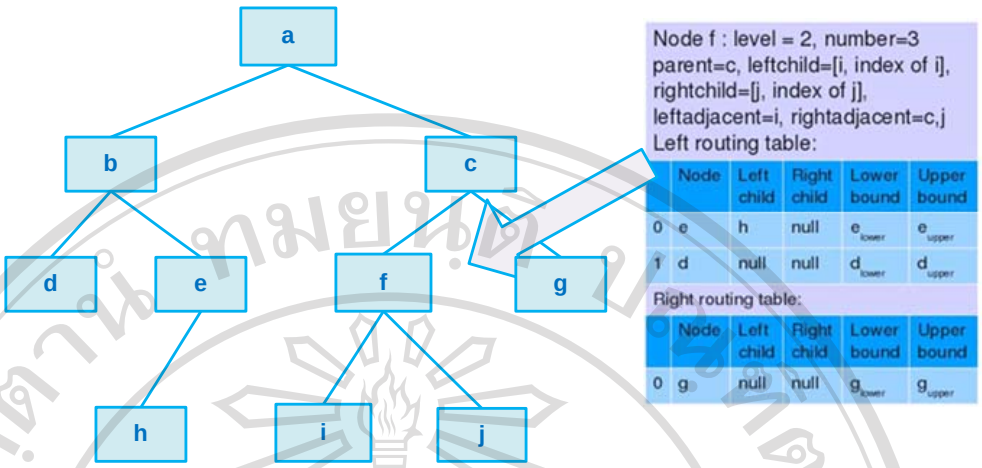
1) เชื่อมโยงไปยังโหนดพาเร้นท์(Parent) ซึ่งโหนดแม่ของ f คือ c

2) เชื่อมโยงไปยังโหนดลูกทั้งสอง คือโหนด i และ j

3) เชื่อมโยงไปยังโหนดที่อยู่ติดกันด้านซ้ายและโหนดที่อยู่ติดกันด้านขวา (Left Adjacent and Right Adjacent) ที่เกิดจากการท่องแบบอินออร์เดอร์ (In-order Traversal) คือโหนด i และ j

4) เชื่อมโยงไปยังเส้นทางตารางด้านซ้ายและด้านขวา ซึ่งมีที่มาจาก $N-2^{j-1}$ สำหรับโหนดติดกันด้านซ้าย และ$N+2^{j-1}$ สำหรับโหนดที่ติดกันด้านขวา โดยที่ N จำนวนของโหนดที่อยู่ในระดับ (Level) เดียวกัน และ j คือจำนวนที่น้อยกว่าระดับของโหนดที่กำลังพิจารณานั้น คือโหนด e, d และ g

นอกจากนี้ในแต่ละโหนดของ BATON จะเก็บตารางเส้นทาง ดังแสดงในรูปที่ 2.21 แสดงตารางเส้นของโหนด f



รูปที่ 2.21 โครงสร้างของ BATON และตารางเส้นทางของโหนด f

ในวิทยานิพนธ์ฉบับนี้ ผู้วิจัยได้ศึกษางานวิจัย [3] ซึ่งเป็นงานวิจัยที่เสนอแนวคิดของการจัดลำดับเว็บเพจบนเครือข่ายเพียร์ทูเพียร์ โดยพัฒนาขั้นตอนวิธี JXP สำหรับจัดลำดับเว็บเพจบนระบบกระจายศูนย์ ให้เสมือนว่ากำลังจัดลำดับเว็บเพจบนระบบแบบรวมศูนย์ซึ่งได้อธิบายการทำงานของขั้นตอนวิธี JXP ไว้ในหัวข้อ 2.3 หากเว็บลิงค์กราฟขยายขนาดใหญ่ขึ้น ขั้นตอนวิธี JXP จะมีข้อได้เปรียบในด้านประสิทธิ์ภาพของเวลาการคำนวณค่าเพจแรงค์ และประสิทธิภาพของการใช้ทรัพยากรในคอมพิวเตอร์

โดยในขั้นตอนวิธี JXP จะพิจารณาค่าเพจแรงค์ในแต่ละโลคอลเว็บลิงค์กราฟบนเครือข่าย ให้เสมือนกับว่ากำลังพิจารณาค่าเพจแรงค์บนโกลบอลเว็บลิงค์กราฟ อย่างไรก็ตามค่าเพจแรงค์ที่ถือว่ามีประสิทธภาพและถูกต้องที่สุด คือค่าเพจแรงค์ที่ได้จากการพิจารณาบนโกลบอลเว็บลิงค์กราฟ ซึ่งในความเป็นจริงไม่มีความเป็นไปได้ที่จะได้ค่าดังกล่าว เนื่องจากสามารถพิจารณาได้แค่ในโลคอลเว็บลิงค์กราฟเท่านั้น แต่ขั้นตอนวิธี JXP สามารถที่จะพิจารณาค่าเพจแรงค์จากโลคอลเว็บลิงค์กราฟ ให้มีค่าใกล้เคียงกับค่าเพจแรงค์จากโกลบอลเว็บลิงค์กราฟได้

เนื่องจากเว็บลิงค์กราฟมีโครงสร้างที่เปลี่ยนแปลงตลอดเวลา ในขั้นตอนวิธี JXP จะมีการคำนวณค่าเพจแรงค์ใหม่ ในทุกๆเว็บเพจของเว็บลิงค์กราฟในแต่ละเพียร์ ซึ่งมีผลต่อประสิทธิภาพของเวลาการคำนวณ ดังนั้นผู้วิจัยจึงได้นำแนวคิดในขั้นตอนวิธี IPR เพื่อใช้ในการแก้ปัญหา ซึ่งอธิบายไว้ในหัวข้อที่ 2.4