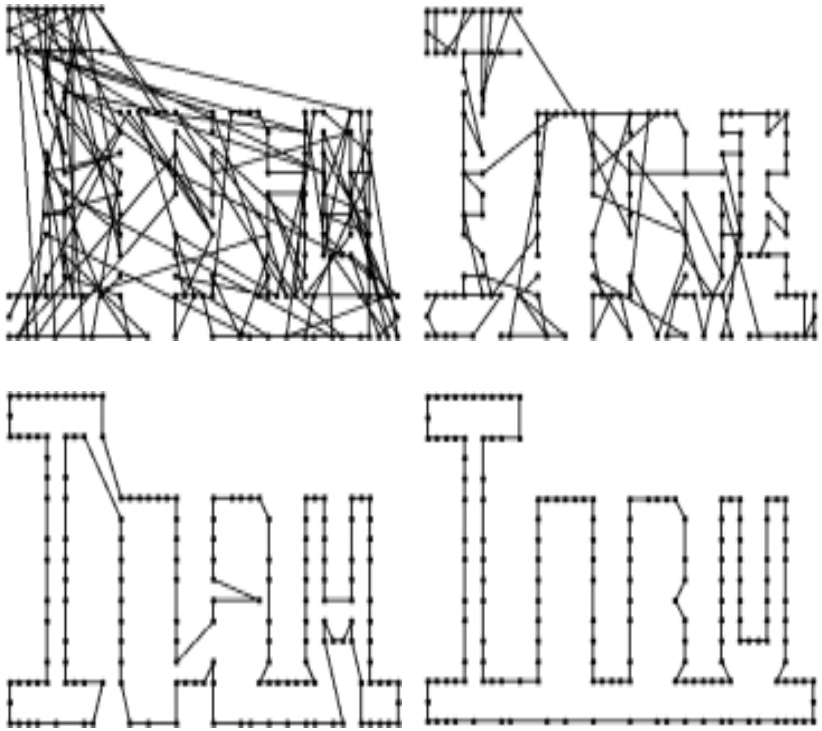
Tabu Search



Boonserm Kijsirikul
Dept. of Computer Engineering,
Chulalongkorn University

แนวคิดของ Tabu Search

- Tabu – ต้องห้าม
- หน่วยย่อยที่ต้องห้ามในการค้นหา Tabu ถูกกำหนดสถานะโดยขึ้นกับหน่วยความจำ (memory) (ซึ่งจะเปลี่ยนแปลงตามเวลาและสภาพแวดล้อม)
- Tabu search ได้จากแนวคิดที่ว่าการค้นหาแบบฉลาดจะต้องพิจารณาถึง

  (1) หน่วยความจำปรับตัว (adaptive memory)
  - สำหรับการค้นหาอย่างมีประสิทธิภาพ

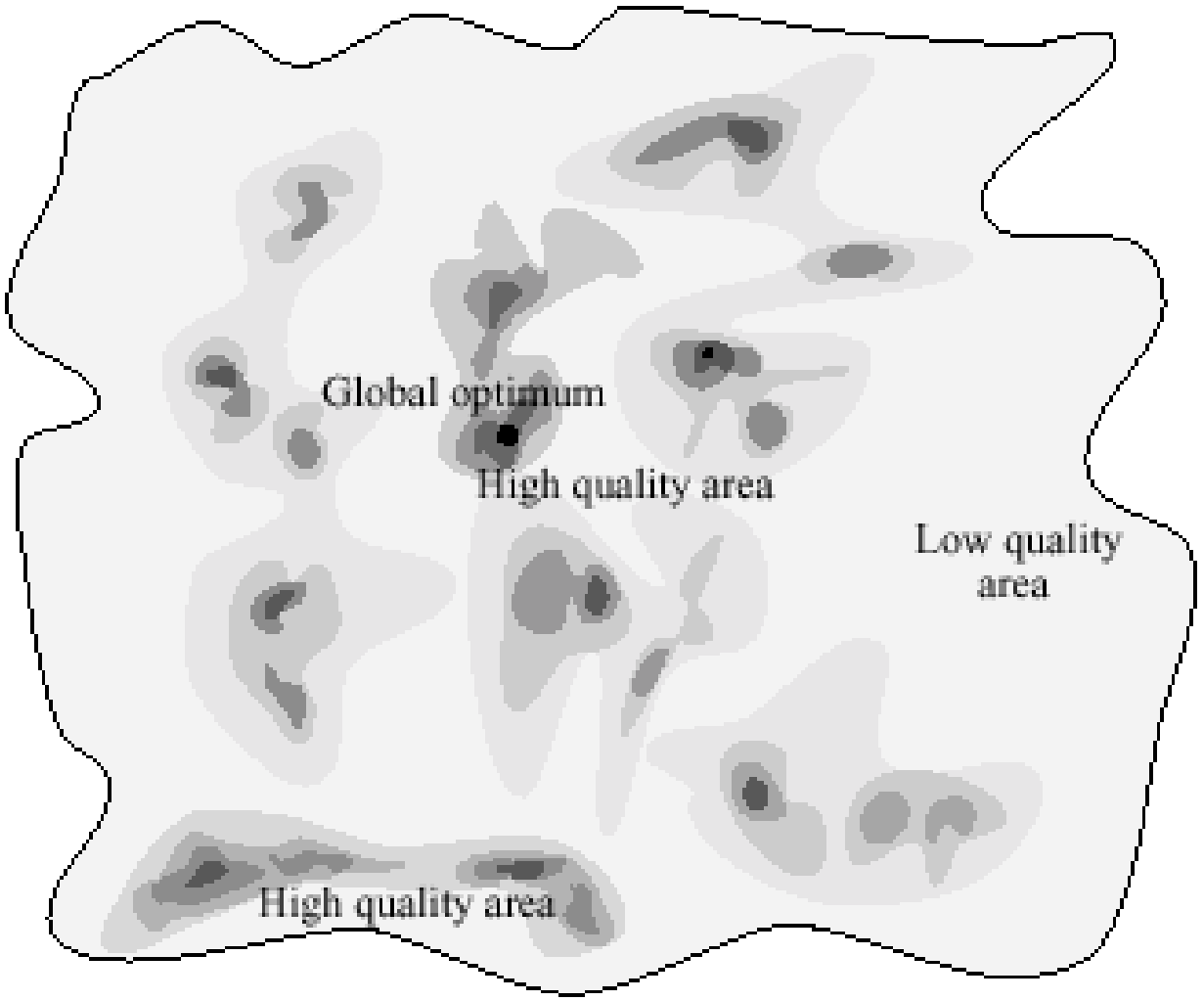
  (2) การสำรวจแบบตอบสนอง (responsive exploration)
  - ได้แนวคิดจากที่ว่าในบางครั้งเส้นทางที่เลวให้ข้อมูลมากกว่าเส้นทางที่ดี เพื่อหาเส้นทางใหม่ที่ดีขึ้น
  - การสำรวจอย่างถี่ถ้วนในเส้นทางที่ดี

แนวคิดของ Tabu Search (ต่อ)

- หน่วยความจำ 2 ชนิด :
  - หน่วยความจำตามเวลา (recency-base memory)
  - หน่วยความจำตามความถี่ (frequency-base memory)
- ความละเอียดและความหลากหลาย

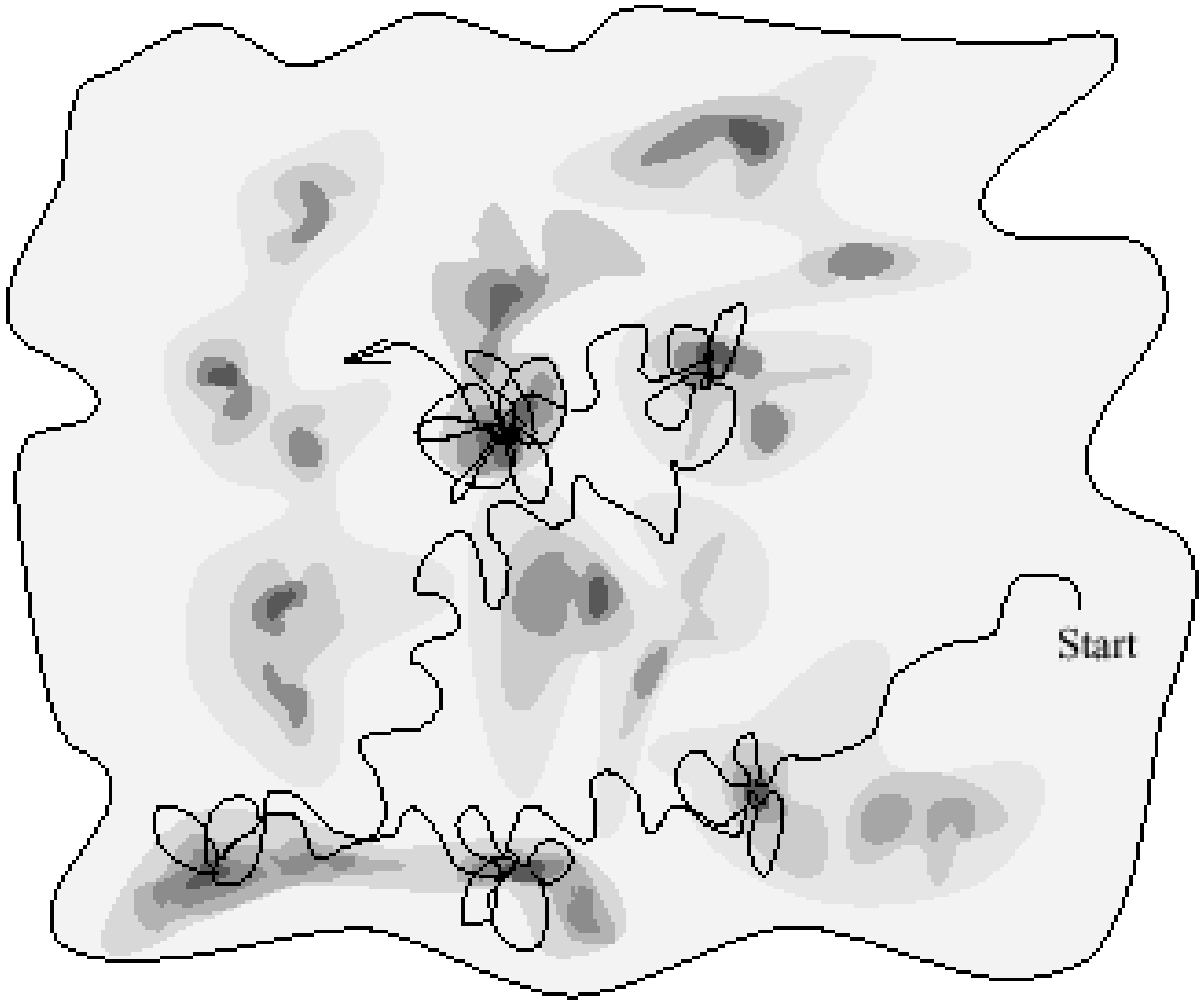
  (intensification and diversification)
  - เพิ่มความละเอียดในการค้นหาบริเวณใกล้เคียงกับคำตอบที่ดีที่เคยพบ
  - ค้นหาเพิ่มเติมในบริเวณที่แตกต่างจากคำตอบที่ดีที่เคยพบ

แนวคิดของ Tabu Search (ต่อ)



a) Solution space model

แนวคิดของ Tabu Search (ต่อ)



d) Tabu search

Tabu Search & Short Term Memory

- ปัญหาการค้นหา : ต้องการหาค่า x ที่ทำให้ฟังก์ชัน f(x) มีค่าต่ำสุด โดยกำหนดให้ N(x) เป็นจุดข้างเคียง (neighborhood) ของ x ใน search space **X**
- การค้นหาแบบ hill-climbing

Hill-climbing algorithm

1) Choose $x \in \mathbf{X}$ to start the process
2) Find $x' \in N(x)$ such that $f(x') < f(x)$
3) If no such x' can be found, x is the local optimum and the method stops.
4) Otherwise, designate x' to be the new x and go to 2).

Tabu Search & Short Term Memory (ต่อ)

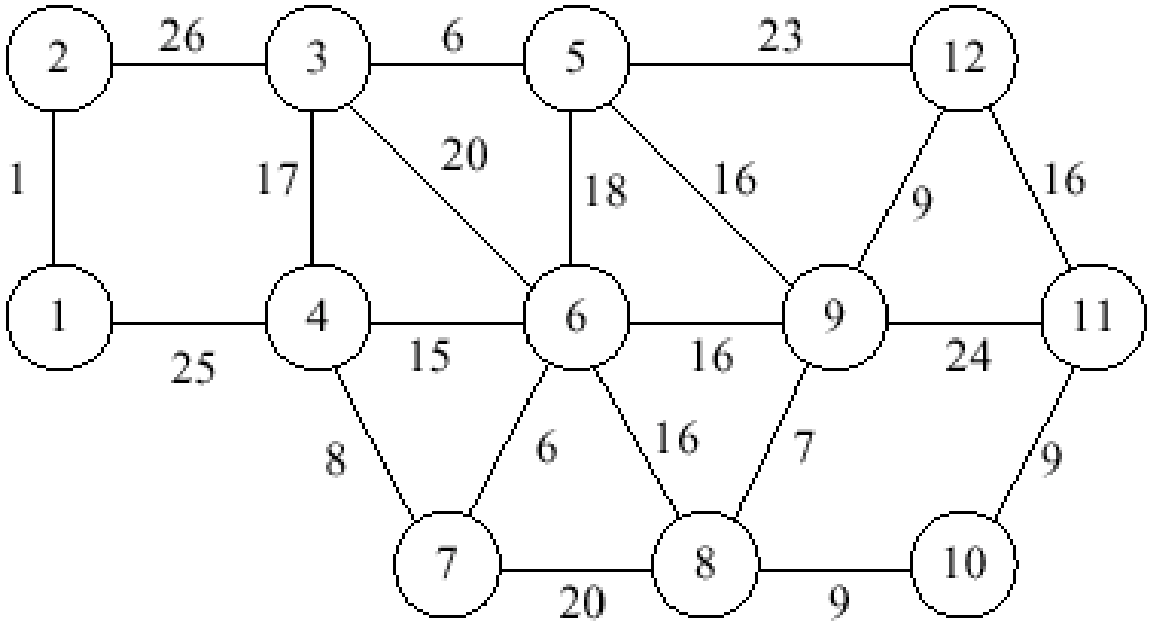
- Tabu Search ใช้หน่วยความจำ 2 แบบเพื่อเปลี่ยนค่า N(x)
  - short term memory สำหรับเป็นหน่วยความจำตามเวลา
  - long term memory หน่วยความจำตามความถี่
- ให้ N*(x) แทนจุดข้างเคียงของ x ที่เปลี่ยนค่าแล้ว
- สำหรับ short term memory
  - N*(x) เป็นเซตย่อยของ N(x)
  - สถานะของ tabu (tabu status) ใช้สำหรับตัดสมาชิกบางตัวของ N(x) ออก
- สำหรับ long term memory
  - N*(x) อาจเป็นซุปเปอร์เซตของ N(x)

Tabu Search & Short Term Memory (ต่อ)

- N*(x) ขึ้นอยู่กับ short term memory และจะเปลี่ยนแปลงในแต่ละครั้งของการค้นหา
- short term memory ใช้เป็นหน่วยความจำตามเวลา
  - เก็บคำตอบหรือคุณสมบัติของคำตอบ (solution attribute) ในการค้นหาที่ผ่านมาเร็วๆนี้
  - คุณสมบัติที่ปรากฏในคำตอบที่พบเร็วๆนี้จะถูกกำหนดให้สถานะเป็น tabu-active
  - คำตอบอื่นๆที่จะพบในอนาคตที่มีคุณสมบัติแบบ tabu-active ก็จะมีสถานะเป็น tabu-active ด้วย
  - ป้องกันการสร้างคำตอบบางตัวไม่ให้อยู่ใน N*(x)

ตัวอย่าง Minimum k-Tree Problem

- Minimum k-tree problem : ให้หาต้นไม้ที่มี k กิ่งจากกราฟ โดยให้ผลรวมของน้ำหนักของกิ่งน้อยที่สุด (ในตัวอย่าง k=4)
  - ต้นไม้คือเซตของกิ่งที่ไม่มีลูป



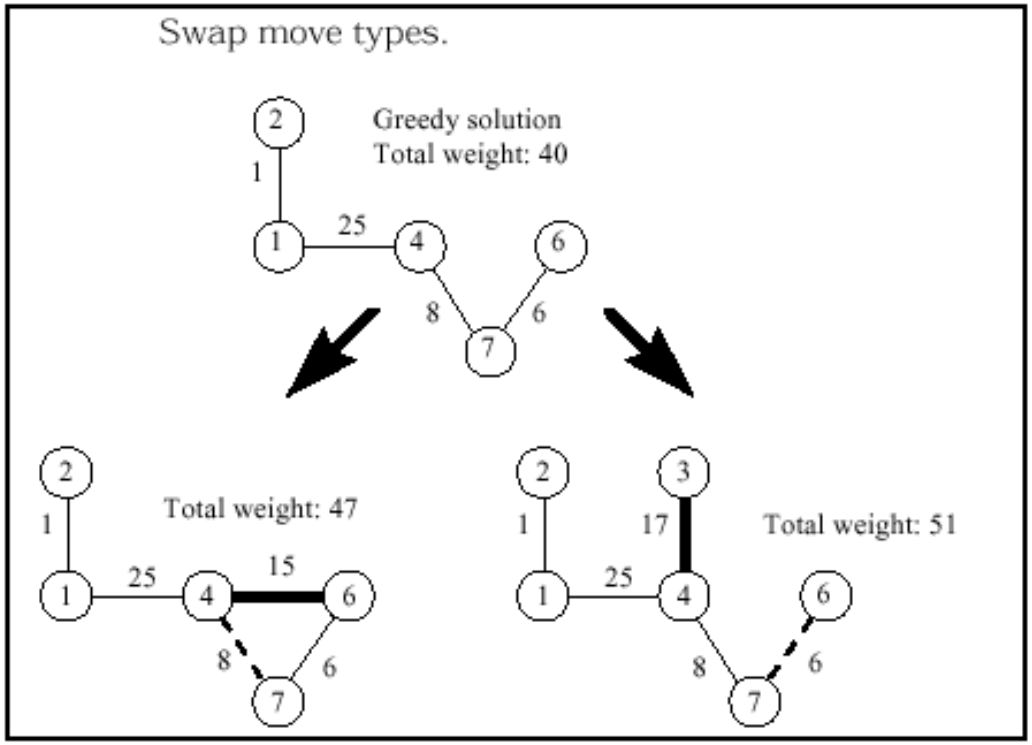
ตัวอย่าง Minimum k-Tree Problem (ต่อ)

- สมมติว่าจากคำตอบ x ใดๆ คำตอบอื่นๆที่อยู่ใน N(x) ได้จากการสลับกิ่ง
- ให้คำตอบแรกได้จากอัลกอริทึม greedy ซึ่งหาเซตของกิ่งเริ่มจากโหนดแรก และให้ผลรวมน้ำหนักในแต่ละครั้งที่เพิ่มกิ่งใหม่น้อยที่สุด

Greedy construction.			
Step	Candidates	Selection	Total Weight
1	(1,2)	(1,2)	1
2	(1,4), (2,3)	(1,4)	26
3	(2,3), (3,4), (4,6), (4,7)	(4,7)	34
4	(2,3), (3,4), (4,6), (6,7), (7,8)	(6,7)	40

ตัวอย่าง Minimum k-Tree Problem (ต่อ)

- เมื่อได้คำตอบเริ่มต้นแล้ว เราสร้างคำตอบใหม่โดยแทนที่กิ่ง 1 กิ่งในต้นไม้ด้วยกิ่งใหม่ โดยที่ผลที่ได้ต้องเป็นต้นไม้
- รูปด้านล่างแสดงการแทนที่ 2 แบบ



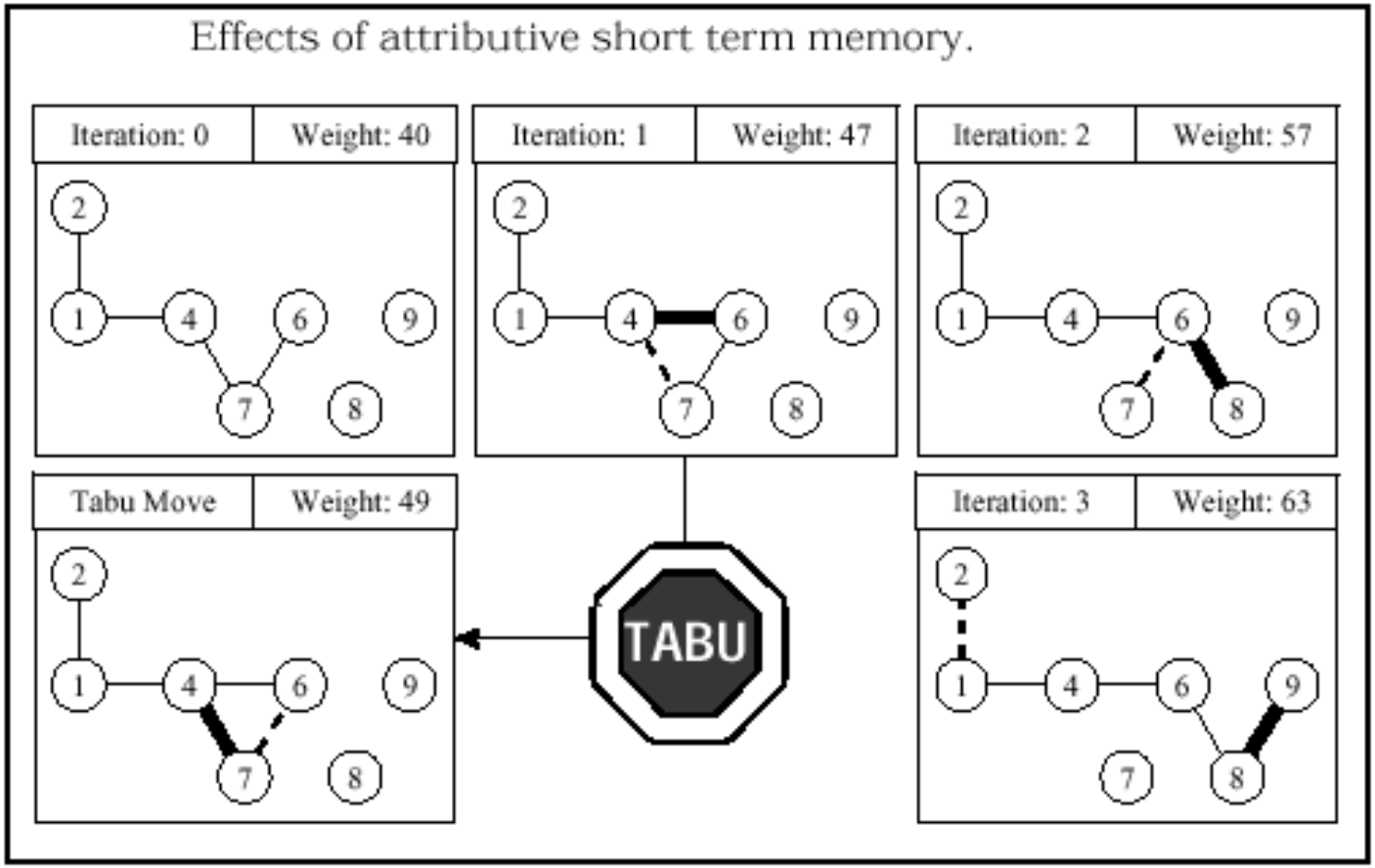
เส้นปะแสดงกิ่งที่ถูกลบ
เส้นหนาแสดงกิ่งที่เพิ่มเข้า

เลือกคุณสมบัติที่ใช้กำหนดสถานะของ tabu

- ในปัญหานี้กำหนดให้ added edge (กิ่งที่เพิ่มเข้า) และ dropped edge (กิ่งที่ลบออก) กำหนดสถานะของ tabu
- added edge หมายถึงคำตอบอื่นๆใน N(x) ที่ลบกิ่งนี้จะมีสถานะเป็น tabu-active (ไม่นำมาพิจารณา)
- dropped edge หมายถึงคำตอบอื่นๆใน N(x) ที่เพิ่มกิ่งนี้จะมีสถานะเป็น tabu-active (ไม่นำมาพิจารณา)
- tabu-tenure คือระยะเวลาที่ added edge หรือ dropped edge จะส่งผลต่อสถานะของ tabu
  - ให้ tabu-tenure ของ dropped edge เท่ากับ 2
  - ให้ tabu-tenure ของ added edge เท่ากับ 1

- ตารางและรูปด้านล่างแสดงการสร้างคำตอบเมื่อผ่านไป 3 รอบ

Iteration	Tabu-active net tenure		Add	Drop	Weight
	1	2			
1			(4,6)	(4,7)	47
2	(4,6)	(4,7)	(6,8)	(6,7)	57
3	(6,8), (4,7)	(6,7)	(8,9)	(1,2)	63



Aspiration Criteria (เกณฑ์ของความหวัง)

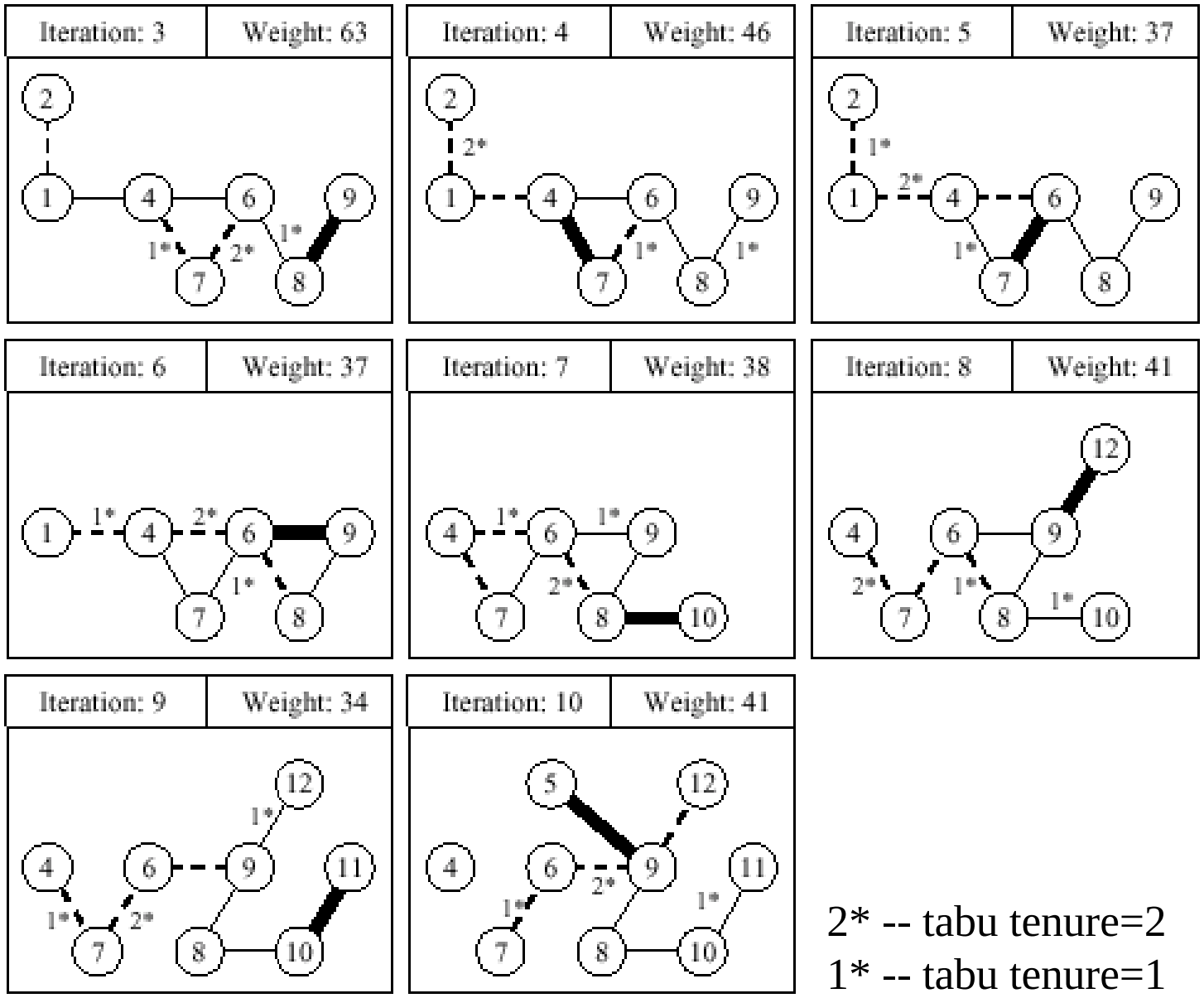
- Aspiration criteria คือเกณฑ์ที่ใช้สำหรับเปลี่ยนสถานะของ tabu สำหรับคำตอบใดๆจาก active เป็น non-active
  - เนื่องจากในบางครั้งคำตอบที่มีสถานะเป็น tabu-active เป็นคำตอบที่ดี
  - โดยทั่วไปจะใช้เกณฑ์ที่ว่า ถ้าเป็นคำตอบที่ให้ค่า f(x) น้อยที่สุดเท่าที่เคยมีมาจะยอมรับคำตอบนั้นได้ (เปลี่ยนสถานะจาก active เป็น non-active)

Tabu Search : Minimum k-Tree Problem

- กำหนดให้เงื่อนไขการหยุดของอัลกอริทึม tabu search คือจำนวนรอบเกินกว่าที่กำหนด (ในปัญหานี้เท่ากับ 10)

Iterations of a first level TS procedure.						
Iteration	Tabu-active net tenure		Add	Drop	Move Value	Weight
	1	2				
3	(6,8), (4,7)	(6,7)	(8,9)	(1,2)	6	63
4	(6,7), (8,9)	(1,2)	(4,7)	(1,4)	-17	46
5	(1,2), (4,7)	(1,4)	(6,7)	(4,6)	-9	37*
6	(1,4), (6,7)	(4,6)	(6,9)	(6,8)	0	37
7	(4,6), (6,9)	(6,8)	(8,10)	(4,7)	1	38
8	(6,8), (8,10)	(4,7)	(9,12)	(6,7)	3	41
9	(4,7), (9,12)	(6,7)	(10,11)	(6,9)	-7	34*
10	(6,7), (10,11)	(6,9)	(5,9)	(9,12)	7	41

Graphical representation of TS iterations.



Long Term Memory

- Long term memory ใช้สำหรับค้นหาคำตอบใหม่ที่แตกต่างจากเดิม โดยการหยุดกระบวนการค้นหาของ short term memory แล้วเริ่มต้นกระบวนการค้นหาที่จุดใหม่
- ใช้ long term memory เพื่อกำหนดว่าควรเลือกจุดใหม่ที่ใด
- long term memory มีได้หลายรูปแบบ รูปแบบหนึ่งที่นิยม คือ critical event memory (ความจำของเหตุการณ์สำคัญ)
- critical event memory จำเหตุการณ์สำคัญทั้งหมดที่ผ่านมา เป็นใช้เป็นข้อมูลสำหรับการสร้างสถานะของ tabu สำหรับจุดใหม่ที่จะใช้เป็นจุดเริ่มต้นของการค้นหาครั้งใหม่
- critical event memory ใช้กำหนด diversification

Long Term Memory (ต่อ)

- ในปัญหานี้กำหนดให้เหตุการณ์สำคัญคือ
  - จุดเริ่มต้นของแต่ละการรันครั้งใหม่
    (รอบที่ 0 ในกรณีของตัวอย่าง)
  - local optimum ที่เกิดขึ้นในแต่ละครั้งของการรัน ซึ่งให้ค่า f(x) น้อยกว่าจุดก่อนหน้าและจุดด้านหลังที่ติดกับจุดนั้น
    (รอบที่ 5, 6, 9 ในกรณีของตัวอย่าง)
- ในตัวอย่างรอบที่ 9 คือรอบที่ให้ค่าต่ำสุด ดังนั้นเราต้องการเริ่มการรันครั้งใหม่ก่อนรอบนี้ สมมติว่าเริ่มที่หลังรอบที่ 7

Long Term Memory (ต่อ)

- ในกรณีนี้เหตุการณ์สำคัญคือเหตุการณ์ที่เกิดในรอบที่ 0, 5, 6
- เรารวมข้อมูลทั้งหมดของทั้งสามรอบไว้ใน long term memory ซึ่งก็คือกิ่ง (1,2), (1,4), (4,7), (6,7), (6,8), (8,9) และ (6,9) (ในกรณีที่ใช้ frequency-based memory เป็น long term memory เราจะใช้จำนวนครั้งเพื่อคิด weight ของแต่ละกิ่งด้วย)
- จากนั้นใช้ข้อมูลที่ได้เป็นตัวป้องกันการสร้างจุดเริ่มต้นที่มีกิ่งเหมือนกับกิ่งในความจำนี้ ในแต่ละขั้นตอนของการสร้างจุดเริ่มต้นจุดใหม่นั้น (ในกรณีของเราใช้ greedy จะทำทั้งหมด 4 ขั้นตอน – มี 4 กิ่ง) จะป้องกันมากถ้าเป็นขั้นตอนต้นๆ และป้องกันน้อยถ้าเป็นขั้นตอนท้ายๆของการสร้างจุดเริ่มใหม่

Long Term Memory (ต่อ)

- ในกรณีของเรากำหนดให้การป้องกันเป็นดังต่อไปนี้
  - ใน 2 ขั้นตอนแรก ห้ามมีกิ่งในความจำเลย
  - หลังจากนั้นยอมให้มีกิ่งในความจำได้ (กิ่งทั้งหมดมีแค่ 4)

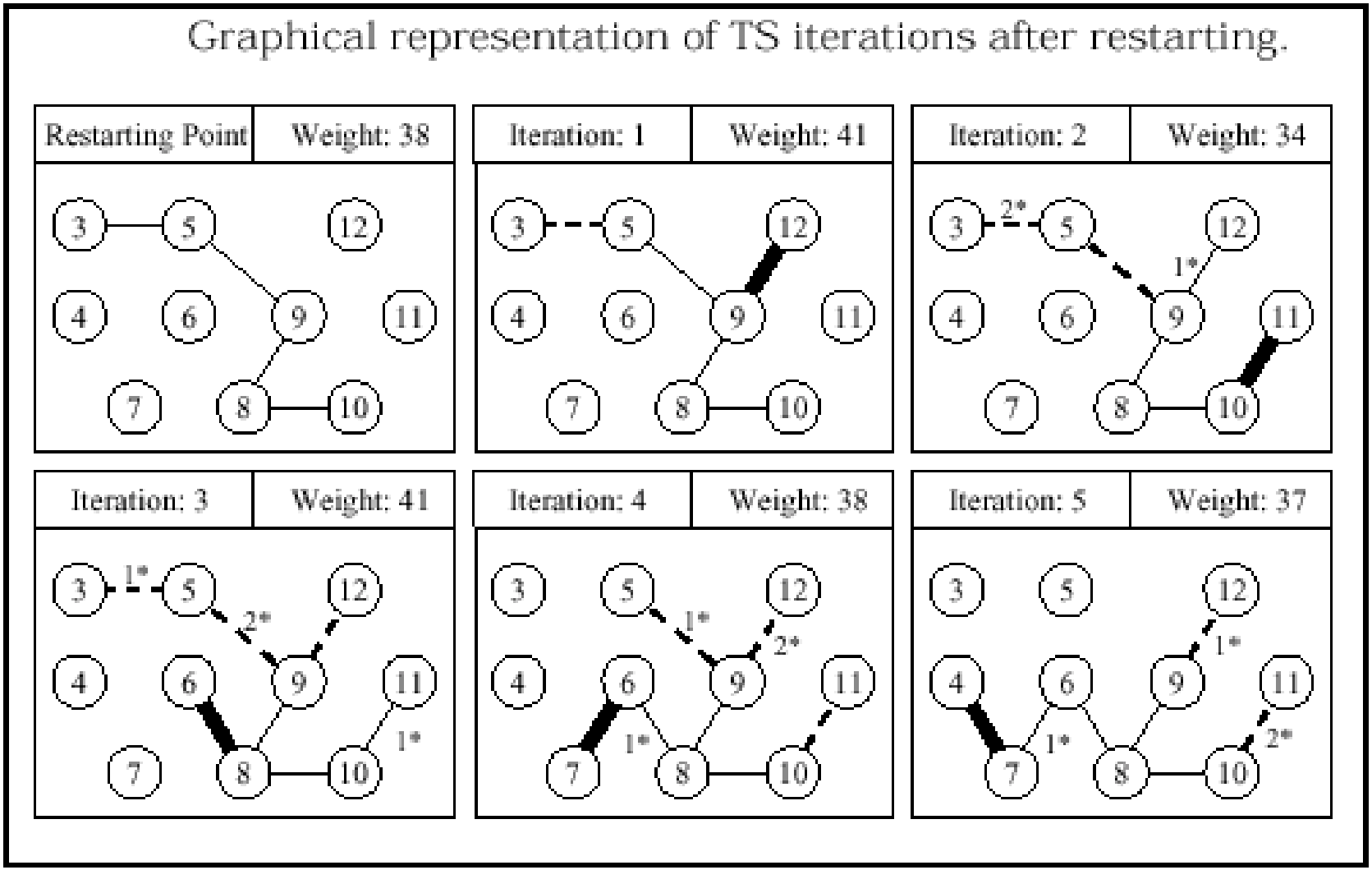
Restarting procedure.			
Step	Candidates	Selection	Total Weight
1	(3,5)	(3, 5)	6
2	(2,3), (3,4), (3,6), (5,6), (5,9), (5,12)	(5, 9)	22
3	(2,3), (3,4), (3,6), (5,6), (5,12), (6,9), (8,9), (9,12)	(8, 9)	29
4	(2,3), (3,4), (3,6), (5,6), (5,12), (6,8), (6,9), (7,8), (8,10), (9,12)	(8, 10)	38

Restarting Procedure

- จากจุดเริ่มต้นใหม่ เราใช้กระบวนการค้นหาเดิม ผลได้ดังตารางและรูปต่อไปนี้

TS iterations following restarting.						
Iteration	Tabu-active net tenure		Add	Drop	Move Value	Weight
	1	2				
1			(9,12)	(3,5)	3	41
2	(9,12)	(3,5)	(10,11)	(5,9)	-7	34*
3	(3,5), (10,11)	(5,9)	(6,8)	(9,12)	7	41
4	(5,9), (6,8)	(9,12)	(6,7)	(10,11)	-3	38
5	(9,12), (6,7)	(10,11)	(4,7)	(8,10)	-1	37

Restarting Procedure (ต่อ)



Tabu Search Algorithm

Choose an initial (possibly random) solution $x \in \mathbf{X}$
x* := x, k := 1,
Initialize tabu short-term memory and long-term memory
while the stopping condition is not met **do**
  k := k+1
  Generate a candidate set N*(x) including x'' with tabu-active
  which satisfies aspiration criteria
  Find a best x' $\in$ N*(x)
  if local optimum reached **then begin**
    if no improvement made over a period **then**
      Apply long term memory to restart the process,
      and find a new solution x'
  end if
  x := x'
  Update tabu memory and adjust search parameters
  if f(x')<f(x*) **then** x* := x'
end while