



รายงานการวิจัย

วิธีการทางวิศวกรรมความรู้เพื่อการรู้จำบริเวณกำหนดรหัสทางพันธุกรรม (A Knowledge Engineering Approach to the Recognition of Genomic Coding Regions)



ได้รับทุนอุดหนุนการวิจัยจาก
มหาวิทยาลัยเทคโนโลยีสุรนารี

ผลงานวิจัยเป็นความรับผิดชอบของหัวหน้าโครงการวิจัยแต่เพียงผู้เดียว



รายงานการวิจัย

วิธีการทางวิศวกรรมความรู้เพื่อการรู้จำบริเวณกำหนดรหัสทางพันธุกรรม (A Knowledge Engineering Approach to the Recognition of Genomic Coding Regions)

ผู้วิจัย

รองศาสตราจารย์ ดร.นิตยา เกิดประสพ

รองศาสตราจารย์ ดร.กิตติศักดิ์ เกิดประสพ

สาขาวิชาวิศวกรรมคอมพิวเตอร์

สำนักวิชาวิศวกรรมศาสตร์

หัวหน้าโครงการ

ผู้วิจัยร่วม

ได้รับทุนอุดหนุนการวิจัยจากมหาวิทยาลัยเทคโนโลยีสุรนารี ปีงบประมาณ พ.ศ. 2556 2557 และ 2558

ผลงานวิจัยเป็นความรับผิดชอบของหัวหน้าโครงการวิจัยแต่เพียงผู้เดียว

กันยายน 2560

กิตติกรรมประกาศ

คณะผู้วิจัยขอขอบคุณมหาวิทยาลัยเทคโนโลยีสุรนารี และสำนักงานคณะกรรมการวิจัยแห่งชาติ ที่สนับสนุนโครงการวิจัยนี้ด้วยการจัดสรรงบประมาณให้อย่างพอเพียงและต่อเนื่อง ตั้งแต่ปีงบประมาณ พ.ศ.2556 2557 และ 2558 รวมถึงขอขอบคุณผู้ทรงคุณวุฒิทั้งภายนอกและภายในมหาวิทยาลัย ที่ได้เสียสละเวลาทำหน้าที่ตรวจข้อเสนอโครงการวิจัยและร่างรายงานการวิจัยฉบับสมบูรณ์ ข้อเสนอแนะจากผู้ทรงคุณวุฒิทุกท่านเป็นประโยชน์อย่างมากต่อคณะผู้วิจัยในการปรับปรุงการออกแบบ และขั้นตอนการดำเนินงานของโครงการวิจัย งานวิจัยนี้สำเร็จได้อย่างดีด้วยการมีส่วนร่วมจากนักศึกษาทั้งในระดับปริญญาโทบัณฑิตและปริญญาตรีบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ ที่ได้ทำหน้าที่เป็นผู้ช่วยวิจัยในโครงการวิจัยนี้



บทคัดย่อภาษาไทย

นับตั้งแต่การประกาศเริ่มต้นโครงการจีโนมมนุษย์ในปี ค.ศ. 1990 จนกระทั่งการถอดรหัสโครโมโซมทั้งหมดเสร็จสมบูรณ์ในปี ค.ศ. 2003 ทำให้เกิดการถอดรหัสเซลล์ของสิ่งมีชีวิตและสร้างข้อมูลจีโนมเพิ่มขึ้นอย่างรวดเร็วมากในแต่ละปี แต่การตีความเพื่อใช้ประโยชน์ข้อมูลจีโนมเหล่านี้ทำได้ช้ามากทำให้เกิดงานด้านชีวสารสนเทศศาสตร์ ที่เน้นการนำเทคโนโลยีคอมพิวเตอร์มาช่วยในการรู้จำรหัสพันธุกรรมและการชี้ตำแหน่งโครงสร้างหลักในสายดีเอ็นเอ เช่นการระบุตำแหน่งผู้ให้และผู้รับในสายดีเอ็นเอที่เป็นขั้นตอนเริ่มต้นของการสร้างโปรตีน ขั้นตอนวิธีที่มักจะใช้ในการรู้จำรหัสพันธุกรรมประกอบด้วยเทคนิคฮิดเดนมาร์คอฟโมเดล โครงข่ายแบบเบย์ส์ และกำหนดการพลวัต ในระยะหลังเริ่มมีการใช้เทคนิคอัจฉริยะเช่น โครงข่ายประสาทเทียม ซัพพอร์ตเวกเตอร์แมชชีน และจีเนติกอัลกอริทึม มาช่วยเพิ่มความแม่นยำของการรู้จำรหัสพันธุกรรม

ในโครงการวิจัยนี้ผู้วิจัยกำหนดขอบเขตการศึกษาการรู้จำตำแหน่งพันธุกรรมในสายดีเอ็นเอที่เกี่ยวข้องกับการสังเคราะห์โปรตีนของสิ่งมีชีวิตชั้นสูง การรู้จำในงานวิจัยนี้เน้นการจำแนกตำแหน่งเชื่อมต่อระหว่างส่วนอินทรอนและเอ็กซอนในสายดีเอ็นเอขนาดสั้น เทคนิคการรู้จำเป็นการประยุกต์กระบวนการวิศวกรรมความรู้ร่วมกับขั้นตอนวิธีทางปัญญาประดิษฐ์ โดยในกระบวนการดังกล่าวจะรวมขั้นตอนการคัดเลือกฟีเจอร์ การสร้างโมเดล การประเมินความถูกต้องของโมเดล และการแปลงโมเดลเป็นกฎเพื่อแสดงผลลัพธ์ในลักษณะของฐานความรู้ การนำเสนอโมเดลในรูปแบบของฐานความรู้จะช่วยให้ผู้ใช้งานโปรแกรมทำความเข้าใจได้ง่าย นอกจากนี้ยังช่วยให้ปรับเปลี่ยนโมเดลได้สะดวกเมื่อข้อมูลในอนาคตมีการเปลี่ยนแปลง โปรแกรมที่ออกแบบยังสามารถทำงานในลักษณะยืดหยุ่นในกรณีที่มีข้อมูลไม่สมบูรณ์ทำให้โมเดลเป็นการค้นหาความรู้โดยประมาณ การพัฒนาโปรแกรมของงานวิจัยนี้ใช้ภาษาเอแอล และเขียนโปรแกรมให้ทำงานได้ทั้งในแบบการโปรแกรมแบบลำดับและการโปรแกรมแบบขนาน โปรแกรมทั้งสองแบบนี้เปิดเผยซอร์สโค้ดเพื่อให้ นักวิจัยที่สนใจสามารถพัฒนาต่อยอดงานวิจัยได้

บทคัดย่อภาษาอังกฤษ

Since the announcement of the human genome project in 1990 up to the successful sequencing of all the human chromosomes in the year 2003, the amount of available genome data has been increasing exponentially each year. Unfortunately, genomic interpretation cannot keep pace with such tremendous raw sequenced data. Computational methods to gene recognition and identification of its structural elements such as donor and acceptor splice sites are thus important to the success of bioinformatics. The widely used methods for gene recognition include hidden Markov model, Bayesian network, and dynamic programming. Recent advances in gene prediction tools apply computational intelligent methods such as artificial neural network, support vector machines, and genetic algorithms to produce a more accurate model.

In this project, we consider the problem of recognizing coding regions for protein biosynthesis in eukaryotes. The recognition task is to separate coding and non-coding regions, and to identify the boundaries of intron and exon parts in the unknown DNA sequences. We tackle the problem with the knowledge engineering approach in which not only the machine learning techniques are employed, but also the whole process of knowledge discovery including feature selection, data modeling, model validation, and rule extraction is to be designed and developed. The advantages of the proposed knowledge engineering approach are the ease of use, the automatic generation of informative and comprehensible model, and the adaptation on new information. The induced prediction model is also expected to work well with approximate, incomplete, and uncertain data due to ambiguity in DNA sequencing. We developed the DNA coding region recognition program with the Erlang programming language in both sequential and parallel modes. The program is open source in such a way that the source code is publicly available for further improvement by interesting researchers.

สารบัญ

	หน้า
กิตติกรรมประกาศ	ก
บทคัดย่อภาษาไทย	ข
บทคัดย่อภาษาอังกฤษ	ค
สารบัญ	ง
สารบัญตาราง	ช
สารบัญภาพ	ซ
บทที่ 1 บทนำ	
1.1 ความสำคัญและที่มาของปัญหาการวิจัย	1
1.2 งานวิจัยที่เกี่ยวข้อง	6
1.3 วัตถุประสงค์ของโครงการวิจัย	8
1.4 ขอบเขตของการวิจัย	8
1.5 ประโยชน์ที่ได้รับ	8
บทที่ 2 การออกแบบและพัฒนาโปรแกรม	
2.1 กรอบแนวคิด	10
2.2 การออกแบบและพัฒนาโปรแกรม	11
2.3 การแปลงโมเดลเป็นฐานความรู้	15
2.4 การจัดการความไม่แน่นอนด้วยวิธีการค้นพบโดยประมาณ	18
2.5 การเพิ่มประสิทธิภาพโปรแกรมด้วยการทำงานแบบขนาน	20
บทที่ 3 ผลการทดสอบโปรแกรม	
3.1 ข้อมูลที่ใช้ในการทดสอบ	22
3.2 มาตรฐานประสิทธิภาพโมเดลในการทำนายจุดเชื่อมต่อในยีน	24
3.3 ผลการทดสอบโปรแกรมการค้นพบรูปแบบที่ปรากฏบ่อยในข้อมูลดีเอ็นเอ	28
3.4 ผลการทดสอบวิธีการค้นพบโดยประมาณ	31
3.5 ผลการทดสอบวิธีการค้นพบรูปแบบที่ปรากฏบ่อยแบบขนาน	33
บทที่ 4 บทสรุป	
4.1 สรุปผลการวิจัย	34
4.2 ข้อจำกัดของโปรแกรมและข้อเสนอแนะ	36
บรรณานุกรม	37

สารบัญ(ต่อ)

	หน้า
ภาคผนวก ก บทความวิจัยตีพิมพ์	41
<i>International Journal Articles</i>	
1. N. Kerdprasop, K. Kerdprasop (2015). Constraint-based system for genomic analysis. International Journal of Information and Education Technology, vol.5, no.2, February, pp.119-123. (indexing: INSPEC, ISSN: 2010-3689)	42
2. N. Kerdprasop, K. Kerdprasop (2014). Visual knowledge mining and utilization in the inductive expert system. International Journal of Computers, vol.8, pp.157-165. (ISSN 1998-4308)	47
3. N. Kerdprasop, K. Kerdprasop (2014). The discovery of top-k DNA frequent patterns with approximate method. Malaysian Journal of Computing, vol.2, no.2, November/December, Article No.3. 12pp. (ISSN 2231-7473)	56
4. K. Kerdprasop, N. Kerdprasop (2013). Concurrent data mining and genetic computing implemented with Erlang language. International Journal of Software Engineering and Its Applications, vol.7, no.3, May, pp.63-75. (indexing: Scopus, ISSN: 1738-9984)	68
5. N. Kerdprasop, F. Koongaew, K. Kerdprasop (2013). Building and querying a decision tree model with constraint logic programming. International Journal of Software Engineering and Its Applications, vol.7, no.5, September, pp.269-282. (indexing: Scopus, ISSN: 1738-9984)	81
6. N. Kerdprasop, K. Kerdprasop (2013). Knowledge engineering process for a rapid prototyping of inductive expert system. International Journal of Computer Science Issues (IJCSI), vol.10, issue 2, no.3, March, pp.408-414.	95
<i>Refereed International Conference Proceedings</i>	
7. N. Kerdprasop, K. Kerdprasop (2014). Visual data mining and the creation of inductive knowledge base. Proceedings of the 5th International Conference on Circuits, Systems, Control, Signals (CSCS'14), Salerno, Italy, 3-5 June, pp.181-186.	102
8. N. Kerdprasop, K. Kerdprasop (2013). Knowledge mining and its application to support computational health informatics. Proceedings of the International Conference on Education, Language, Society, Science and Engineering in ASEAN and its Neighbors, Kunming, People's Republic of China, 23-28 February, pp.274-280.	108

สารบัญ(ต่อ)

	หน้า
ภาคผนวก ข รหัสต้นฉบับของโปรแกรม	115
1. โปรแกรมการค้นพบรูปแบบที่ปรากฏบ่อยในข้อมูลดีเอ็นเอ (ลิขสิทธิ์เลขที่ ว1. 5346 ออกให้ ณ วันที่ 8 เมษายน พ.ศ. 2558)	116
2. โปรแกรมแบบขนานเพื่อค้นหาความสัมพันธ์ (ลิขสิทธิ์เลขที่ ว1. 5347 ออกให้ ณ วันที่ 8 เมษายน พ.ศ. 2558)	121
ประวัติผู้วิจัย	127



สารบัญตาราง

	หน้า
ตารางที่ 3.1 การกระจายของข้อมูลในแต่ละคลาสของชุดข้อมูลฝึกและข้อมูลทดสอบ	24
ตารางที่ 3.2 โครงสร้างคอนฟิวชันเมตริกซ์สำหรับข้อมูลการทำนายจุดเชื่อมต่อในสายดีเอ็นเอ	25
ตารางที่ 3.3 การเปรียบเทียบเทคนิคการทำนายจุดเชื่อมต่อ exon/intron	28
ตารางที่ 3.4 การเปรียบเทียบเทคนิคการทำนายจุดเชื่อมต่อ intron/exon	29
ตารางที่ 3.5 การเปรียบเทียบเทคนิคการทำนายดีเอ็นเอกรณีไม่ปรากฏจุดเชื่อมต่อ	30
ตารางที่ 3.6 การเปรียบเทียบจำนวนรูปแบบที่ค้นพบโดยวิธีปกติและโดยวิธีประมาณ	32
ตารางที่ 3.7 ผลการวิเคราะห์จำนวนรูปแบบที่ตรงกันระหว่างการค้นพบแบบปกติและ แบบประมาณ	33



สารบัญภาพ

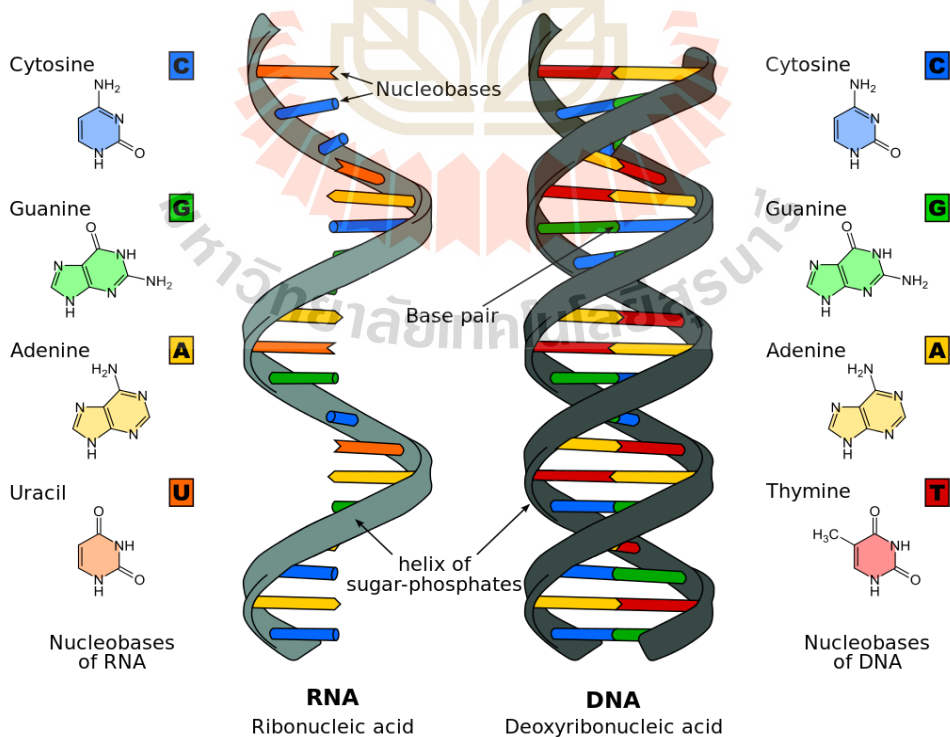
	หน้า
รูปที่ 1.1 โครงสร้างสายเดี่ยวของอาร์เอ็นเอและเกลียวคู่ของดีเอ็นเอ	1
รูปที่ 1.2 รหัสโคดอนบนสายอาร์เอ็นเอและตารางแสดงความเชื่อมโยงระหว่างกรดอะมิโน และรหัสโคดอน	2
รูปที่ 1.3 ขั้นตอนเชื่อมต่อเพื่อตัดส่วนอินทรอนออกจากสายอาร์เอ็นเอ	4
รูปที่ 1.4 รวมขั้นตอนการคัดลอกรหัส การเชื่อมต่อส่วนเอ็กซอนและการแปลรหัสในยูคาริโอต	4
รูปที่ 1.5 โครงสร้างอินทรอน และการตัดทิ้งส่วนอินทรอนและเชื่อมต่อส่วนเอ็กซอน ในขั้นตอนการถอดรหัสคำสั่งจากดีเอ็นเอไปยังเอ็มอาร์เอ็นเอ	5
รูปที่ 2.1 แนวคิดของโปรแกรม assoDNA สำหรับทำนายจุดเชื่อมต่อในดีเอ็นเอ	10
รูปที่ 2.2 ขั้นตอนวิธีของโปรแกรม assoDNA	12
รูปที่ 2.3 การทำงานของโปรแกรม assoDNA แสดงในลักษณะ flowchart	13
รูปที่ 2.4 ฟังก์ชันหลักและฟังก์ชันค้นหาความสัมพันธ์ของโปรแกรม assoDNA	14
รูปที่ 2.5 ตัวอย่างจอภาพที่เป็นผลการทำงานของโปรแกรม assoDNA	14
รูปที่ 2.6 การแปลงโมเดลเป็นฐานความรู้ด้วยโปรแกรมเชิงภาพ VisiRule	16
รูปที่ 2.7 ชุดคำสั่งภาษาเชิงตรรกะที่ได้จากการประมวลผลภาพในรูปที่ 2.6	16
รูปที่ 2.8 การโต้ตอบกับฐานความรู้แล้วให้ผลวินิจฉัยเป็นรูปแบบเชื่อมต่อ exon-intron	17
รูปที่ 2.9 การโต้ตอบกับฐานความรู้แล้วให้ผลวินิจฉัยเป็นรูปแบบเชื่อมต่อ intron-exon	17
รูปที่ 2.10 การโต้ตอบกับฐานความรู้แล้วให้ผลวินิจฉัยว่าไม่ปรากฏรูปแบบเชื่อมต่อ	18
รูปที่ 2.11 กรอบแนวคิดของการสุ่มเลือกข้อมูลตามความหนาแน่นเพื่อบันทึกไว้ในแหล่งกักเก็บ	19
รูปที่ 2.12 ขั้นตอนวิธีของการสุ่มเลือกข้อมูลตามความหนาแน่นด้วยการเลื่อนกรอบหน้าต่างต่าง	19
รูปที่ 2.13 ชุดคำสั่งภาษาเอแอลสำหรับประมวลผลโปรแกรม assoDNA แบบขนาน	20
รูปที่ 2.14 จอภาพแสดงการประมวลผลโปรแกรม assoDNA แบบขนาน	21
รูปที่ 3.1 โครงสร้างของยีนที่ประกอบด้วยส่วน exon และ intron	23
รูปที่ 3.2 ตัวอย่างข้อมูลที่ใช้ในการทดสอบโปรแกรม	23
รูปที่ 3.3 การเปรียบเทียบค่า F-measure และ FP rate ของโมเดลตรวจจับ exon/intron	28
รูปที่ 3.4 การเปรียบเทียบค่า F-measure และ FP rate ของโมเดลตรวจจับ intron/exon	29
รูปที่ 3.5 การเปรียบเทียบค่า F-measure และ FP rate ของโมเดลกรณีไม่ปรากฏจุดเชื่อมต่อ	30
รูปที่ 3.6 การเปรียบเทียบค่า F-measure และ FP rate ของโมเดลโดยเฉลี่ยของทุกคลาส	31
รูปที่ 3.7 เปรียบเทียบเวลาในการประมวลผลโปรแกรมแบบลำดับและแบบขนาน	33

บทที่ 1

บทนำ

1.1 ความสำคัญและที่มาของปัญหาการวิจัย

การสร้างโปรตีนมีความสำคัญต่อการดำรงชีพของสิ่งมีชีวิตทั้งหลาย ทั้งนี้เนื่องจากโปรตีนเป็นส่วนประกอบที่สำคัญในร่างกาย นับตั้งแต่กระดูก เนื้อเยื่อ เอ็นไซม์ รวมถึงสารประกอบอื่น ๆ ที่มีความสำคัญต่อกระบวนการทางชีวเคมี เช่น ฮีโมโกลบินในเซลล์เม็ดเลือดแดง เป็นต้น กระบวนการสร้างโปรตีนในสิ่งมีชีวิต จะถูกควบคุมด้วยชุดคำสั่งที่ถูกเก็บอยู่ในรูปแบบของรหัสพันธุกรรมในสายดีเอ็นเอ (DNA – deoxyribonucleic acid) โดยดีเอ็นเอจะถูกบรรจุอยู่ในโครโมโซม และโครโมโซมจะอยู่ภายในนิวเคลียสของเซลล์ เมื่อเริ่มกระบวนการสร้างโปรตีน รหัสพันธุกรรมในดีเอ็นเอที่มีโครงสร้างเป็นเกลียวคู่ จะถูกคัดลอกให้อยู่ในโครงสร้างที่มีขนาดเล็กลงในลักษณะโครงสร้างสายเดี่ยวของอาร์เอ็นเอ (RNA – ribonucleic acid) ในรูปที่ 1.1 แสดงโครงสร้างสายเดี่ยวของอาร์เอ็นเอ (ซ้ายมือ) เปรียบเทียบกับโครงสร้างเกลียวคู่ของดีเอ็นเอ (ขวามือ) หน่วยย่อยสุดของรหัสพันธุกรรมในสายดีเอ็นเอ ประกอบด้วยนิวคลีโอไทด์ 4 ชนิด คือ Cytosine, Guanine, Adenine, Thymine หรือเรียกชื่อย่อว่า C, G, A, T โครงสร้างในอาร์เอ็นเอ ประกอบด้วยนิวคลีโอไทด์ที่คล้ายดีเอ็นเอ ได้แก่ Cytosine, Guanine, Adenine, Uracil หรือเรียกชื่อย่อว่า C, G, A, U

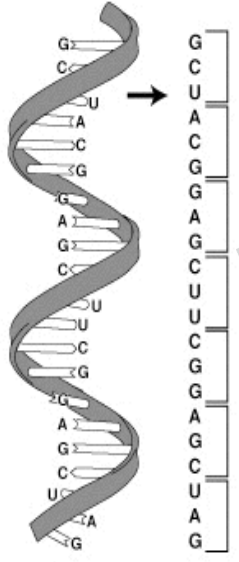


รูปที่ 1.1 โครงสร้างสายเดี่ยวของอาร์เอ็นเอและเกลียวคู่ของดีเอ็นเอ

(ที่มา https://commons.wikimedia.org/wiki/File%3ADifference_DNA_RNA-EN.svg)

โครงสร้างของอาร์เอ็นเอเป็นสายเดี่ยวทำให้มีขนาดเล็กกว่าดีเอ็นเอ และการมีขนาดเล็กนี้ช่วยให้อาร์เอ็นเอสามารถเคลื่อนที่ผ่านรูพรุนของเยื่อหุ้มนิวเคลียสได้ ในกระบวนการถ่ายทอดคำสั่งเพื่อสร้างโปรตีน จึงเริ่มต้นด้วยขั้นตอนการคัดลอกรหัสพันธุกรรมจากดีเอ็นเอไปสู่อาร์เอ็นเอเรียกว่า transcription โดยอาร์เอ็นเอที่ได้เรียกว่า messenger RNA หรือ mRNA ที่มีขนาดเล็กพอที่จะเคลื่อนที่ผ่านรูพรุนบนเยื่อหุ้มนิวเคลียสได้ เมื่อเอ็มอาร์เอ็นเอเคลื่อนที่ออกจากนิวเคลียสเข้าสู่ส่วนที่เรียกว่าไซโตพลาสซึม รหัสพันธุกรรมในสายอาร์เอ็นเอจะถูกถอดรหัสโดยไรโบโซม (ribosome) ในขั้นตอนที่เรียกว่าการแปลรหัส หรือ translation

การถอดรหัสคำสั่งจะใช้รหัสสี่นิวคลีโอไทด์ 3 ตำแหน่งที่ปรากฏต่อเนื่องกันบนสายเอ็มอาร์เอ็นเอ เรียกว่า โคดอน (codon) รหัสสามตำแหน่งหรือโคดอนนี้เป็นชุดข้อมูลสำหรับกำหนดรูปแบบการสร้างกรดอะมิโน (รูปที่ 1.2) โคดอน AUG เป็นรหัสกำหนดการเริ่มสร้างกรดอะมิโน และโคดอน UAG/UGA/UAA เป็นรหัสพิเศษทำหน้าที่กำหนดการหยุดสร้างกรดอะมิโน สายของกรดอะมิโนที่ถูกสร้างขึ้น จะถูกม้วนพับและแปลงรูปร่างต่อไปเป็นโครงสร้างขั้นปฐมภูมิ ทุติยภูมิ และตติยภูมิของโปรตีนแต่ละชนิด ความสัมพันธ์จากดีเอ็นเอไปเป็นอาร์เอ็นเอและสุดท้ายนำไปสู่การสังเคราะห์โปรตีนนี้เป็นหลักการพื้นฐานของชีววิทยาระดับโมเลกุล เรียกว่า ความเชื่อหลัก หรือ central dogma (Crick, 1970)



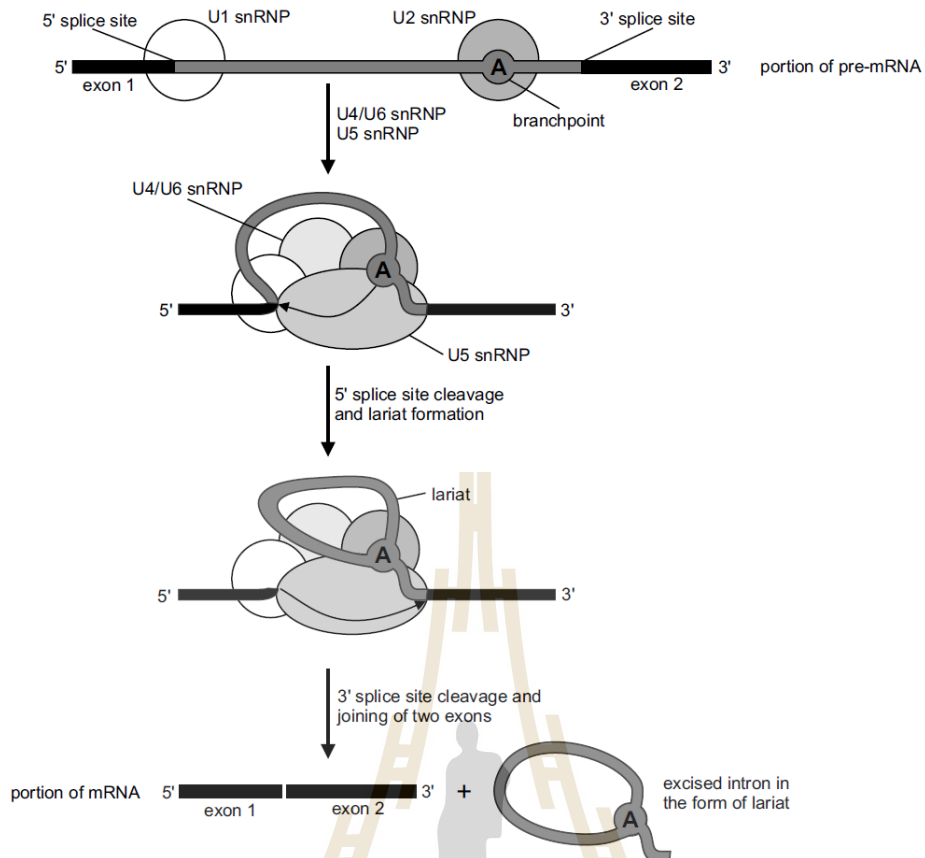
Amino acid	mRNA codons	Amino acid	mRNA codons
Alanine	GCU, GCC, GCA, GCG	Leucine	UUA, UUG, CUU, CUC, CUA, CUG
Arginine	CGU, CGC, CGA, CGG, AGA, AGG	Lysine	AAA, AAG
Asparagine	AAU, AAC	Methionine	AUG
Aspartate	GAU, GAC	Phenylalanine	UUU, UUC
Cysteine	UGU, UGC	Proline	CCU, CCC, CCA, CCG
Glutamine	CAA, CAG	Serine	UCU, UCC, UCA, UCG, AGU, AGC
Glutamate	GAA, GAG	Threonine	ACU, ACC, ACA, ACG
Glycine	GGU, GGC, GGA, GGG	Tryptophan	UGG
Histidine	CAU, CAC	Tyrosine	UAU, UAC
Isoleucine	AUU, AUC, AUA	Valine	GUU, GUC, GUA, GUG
START	AUG	STOP	UAG, UGA, UAA

รูปที่ 1.2 รหัสโคดอนบนสายอาร์เอ็นเอ (ซ้าย) และตารางแสดงความเชื่อมโยงระหว่างกรดอะมิโนและรหัสโคดอน (ขวา) (ที่มา https://en.wikipedia.org/wiki/Genetic_code)

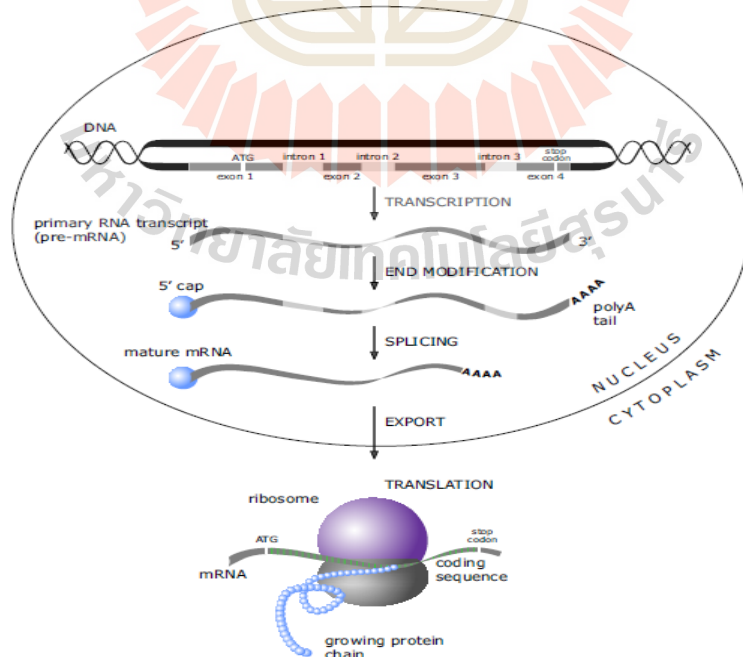
การศึกษาเซลล์ในทางชีววิทยาระดับโมเลกุลได้แบ่งสิ่งมีชีวิตเป็น 2 กลุ่มใหญ่คือ กลุ่มโพรคาริโอต (prokaryotes) และกลุ่มยูคาริโอต (eukaryotes) เซลล์ของสิ่งมีชีวิตในกลุ่มโพรคาริโอตซึ่งเป็นสิ่งมีชีวิตชั้นต่ำที่ยังไม่มีพัฒนาการซับซ้อนจะไม่มีนิวเคลียส ทำให้พบสารที่เป็นรหัสพันธุกรรมกระจายอยู่ทั่วเซลล์ รหัสพันธุกรรมนี้คือชุดของคำสั่งในรูปแบบของโคดอนที่สามารถถูกถอดรหัสเป็นกรดอะมิโน กลุ่มของโคดอนจะปรากฏอยู่ในสายของนิวคลีโอไทด์ โดยแบ่งเป็นส่วนย่อย ๆ เรียกว่าเอ็กซอน (exon - expressed sequences) ดังนั้นเอ็กซอนจึงเป็นส่วนสำคัญของดีเอ็นเอ ทำหน้าที่บันทึกรหัสที่สามารถนำไปใช้แปลเพื่อสร้างเป็นโปรตีน กระบวนการสังเคราะห์โปรตีนจึงเป็นถ่ายทอดรหัสคำสั่งเป็นลำดับจากดีเอ็นเอเป็นอาร์เอ็นเอและได้โปรตีนในขั้นตอนสุดท้าย

สิ่งมีชีวิตชั้นสูงในกลุ่มยูคาริโอต เช่น สัตว์เลี้ยงลูกด้วยนม จะมีความซับซ้อนมากกว่าโพรคาริโอต ภายในเซลล์จะพบนิวเคลียสทำหน้าที่ห่อหุ้มโครโมโซมและสายของดีเอ็นเอไว้ภายใน เมื่อเริ่มกระบวนการสร้างโปรตีน รหัสพันธุกรรมในดีเอ็นเอจะถูกคัดลอกเป็นสายอาร์เอ็นเอระดับปฐมภูมิ เรียกว่า primary RNA transcript หรือ pre-mRNA สายอาร์เอ็นเอที่คัดลอกมาจากดีเอ็นเอนี้ภายในจะปรากฏส่วนของนิวคลีโอไทด์ที่บรรจุรหัสในการสร้างโปรตีนหรือส่วนเอ็กซอน สลับด้วยส่วนของนิวคลีโอไทด์ที่ไม่ได้ใช้ในการสร้างโปรตีน เรียกว่าส่วนอินทรอน (intron - intervening sequences) ลักษณะเช่นนี้แตกต่างจากสิ่งมีชีวิตในกลุ่มโพรคาริโอต ที่ปรากฏเฉพาะส่วนเอ็กซอนต่อเนื่องกันตลอดทั้งสายของดีเอ็นเอและอาร์เอ็นเอ โดยไม่ปรากฏส่วนอินทรอน

การที่เซลล์ในสิ่งมีชีวิตกลุ่มยูคาริโอตปรากฏส่วนของเอ็กซอน สลับด้วยส่วนของอินทรอน ทำให้ขั้นตอนของการคัดลอกรหัสพันธุกรรม (transcription) ที่เกิดขึ้นภายในนิวเคลียสของเซลล์ มีขั้นตอนย่อยเพิ่มขึ้น เรียกว่า ขั้นตอนเชื่อมต่อ (splicing) โดยในขั้นตอนนี้จะเกิดการโค้งงอในสายอาร์เอ็นเอเพื่อให้ส่วนปลายทั้งสองด้านของอินทรอนมาบรรจบกันและถูกตัดออกได้ (รูปที่ 1.3) ส่วนของเอ็กซอนที่เหลือ จะถูกเชื่อมต่อเข้าด้วยกันเป็นสายอาร์เอ็นเอสายยาวสายเดียวที่บรรจุเฉพาะรหัสสร้างโปรตีน สายอาร์เอ็นเอที่ได้นี้เรียกว่า messenger RNA หรือ mRNA สายเอ็มอาร์เอ็นเอจะถูกส่งออกจากนิวเคลียสไปยังส่วนไซโตพลาสซึมของเซลล์ เพื่อให้ส่วนประกอบสำคัญของเซลล์ที่เรียกว่าไรโบโซม ทำหน้าที่ถอดรหัสแต่ละโคดอนเป็นกรดอะมิโนแต่ละชนิด เชื่อมต่อเป็นสายของกรดอะมิโน และสายของกรดอะมิโนจะถูกปรับโครงสร้างต่อไปเป็นโปรตีนที่สมบูรณ์ในขั้นตอนสุดท้าย ขั้นตอนนี้แสดงได้ดังรูปที่ 1.4

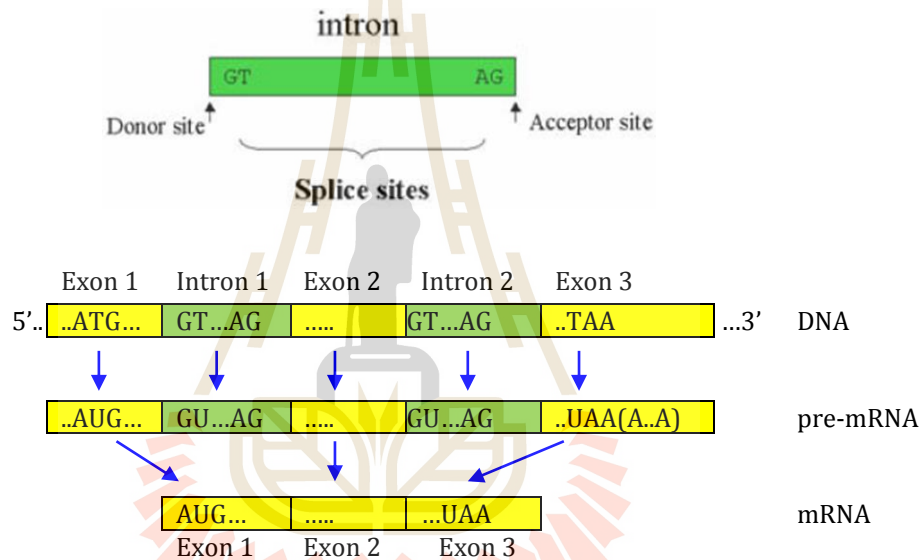


รูปที่ 1.3 ขั้นตอนเชื่อมต่อเพื่อตัดส่วนอินทรนออกจากสายอาร์เอ็นเอ (ที่มา S. Rogic, 2006)



รูปที่ 1.4 รวมขั้นตอนการคัดลอกรหัส การเชื่อมต่อส่วนเอ็กซอนและการแปลรหัสโคดอนในยูคาริโอต (ที่มา S. Rogic, 2006)

กระบวนการเชื่อมต่อเพื่อตัดส่วนอินทรอนและเชื่อมส่วนเอ็กซอนในสายอาร์เอ็นเอ จะดำเนินไปในทิศทางที่แน่นอนคือจากด้าน 5' ไปยังด้าน 3' การตัดและต่อสายอาร์เอ็นเอจะมีการตรวจสอบจุดเชื่อมต่อ (splice site) ซึ่งจะประกอบด้วยจุดเชื่อมระหว่างเอ็กซอนกับอินทรอน เรียกว่า donor site และจุดเชื่อมระหว่างอินทรอนกับเอ็กซอน เรียกว่า acceptor site จุดเชื่อมต่อทั้งสองแบบนี้จะปรากฏทั้งในสายดีเอ็นเอและถูกคัดลอกต่อมาในสายอาร์เอ็นเอ (รูปที่ 1.4) การวิเคราะห์โครงสร้างของสายดีเอ็นเอเพื่อค้นหาตำแหน่งที่แน่นอนของ donor sites และ acceptor sites มีความสำคัญต่อการทำนายโครงสร้างของเอ็มอาร์เอ็นเอ และการทราบโครงสร้างเอ็มอาร์เอ็นเอจะช่วยให้คาดหมายถึงสารประกอบโปรตีนที่จะได้ รวมถึงช่วยให้สามารถทำนายความผิดปกติของการสร้างโปรตีนและคาดคะเนความเสี่ยงที่จะกลายเป็นเซลล์มะเร็ง อันมีสาเหตุมาจากดีเอ็นเอมีรหัสพันธุกรรมที่บกพร่อง



รูปที่ 1.5 โครงสร้างอินทรอน (ภาพบน) และการตัดทั้งส่วนอินทรอนและเชื่อมต่อส่วนเอ็กซอนในขั้นตอนการถอดรหัสคำสั่งจากดีเอ็นเอไปยังเอ็มอาร์เอ็นเอ (ภาพล่าง)

นักวิจัยในสาขาชีววิทยาระดับโมเดลกุลเชิงคำนวณ ได้พยายามค้นหาเทคนิคที่จะช่วยให้สามารถสร้างโมเดลที่แม่นยำในการตรวจสอบและทำนาย donor และ acceptor sites ในสายดีเอ็นเอ เพราะโมเดลที่ได้จะเป็นความรู้พื้นฐานที่สำคัญในการวิเคราะห์ด้านอื่น ๆ ที่เกี่ยวข้องกับรหัสพันธุกรรม โดยในระยะแรกมีการใช้เทคนิคการวิเคราะห์ลำดับของนิวคลีโอไทด์ (sequence-based approach) เพื่อค้นหาารูปแบบที่จะช่วยให้ทำนาย donor และ acceptor sites ได้อย่างแม่นยำ แต่ในระยะหลังนักวิจัยเริ่มใช้เทคนิคของ machine learning ที่ให้ผลลัพธ์ที่รวดเร็วกว่า และความแม่นยำที่สูงขึ้น แต่จากการศึกษาเบื้องต้นกับข้อมูลดีเอ็นเอขนาดเล็ก (2,000 instances, 60 base pairs) ของผู้วิจัยพบว่าการใช้เทคนิค machine learning เช่น support vector machines และ

C4.5 ที่ถึงแม้จะให้โมเดลที่ค่าความแม่นยำในการทำนายโดยรวมมีค่าสูง แต่เมื่อวิเคราะห์โดยละเอียด จะพบว่าการทำนาย true splice sites จะมีกรณีของ false positive ปะปนอยู่ค่อนข้างมาก

ดังนั้นผู้วิจัยจึงมีแนวคิดที่จะพัฒนาโมเดลของการตรวจจับ splice sites บนสายดีเอ็นเอ ที่ให้ผลการทำนายที่มีความแม่นยำและมีอัตรา false positive ต่ำ และรูปแบบของโมเดลที่ได้จะสามารถถูกแปลงเป็นฐานความรู้ด้วยขั้นตอนการแปลงที่เป็นกระบวนการอัตโนมัติ เพื่อให้ได้ระบบฐานความรู้สำหรับสนับสนุนงานด้านการทำนายยีน ระบบฐานความรู้ที่สร้างด้วยกระบวนการอัตโนมัติ นี้จะเป็นความก้าวหน้าใหม่ของการพัฒนาเครื่องมือสำหรับงานชีวสารสนเทศ

1.2 งานวิจัยที่เกี่ยวข้อง

ในช่วงต้นศตวรรษที่ 21 ได้เกิดปรากฏการณ์สำคัญทางด้านชีววิทยาระดับโมเลกุล เมื่อทีมวิจัย Human Genome Project ประกาศความสำเร็จของการถอดรหัสทั้งหมดของโครโมโซมในมนุษย์ ทำให้ได้ข้อมูลของสายดีเอ็นเอ ยาว 3,310 ล้านคู่เบส (Baldi & Brunak, 2001) และประกอบด้วยยีนประมาณ 20,000-25,000 หน่วย (Stein, 2004) การวิเคราะห์ข้อมูลทางชีววิทยาขนาดใหญ่เหล่านี้ ทำให้เกิดสาขาวิจัยใหม่ในชื่อของ ชีวสารสนเทศศาสตร์ หรือ bioinformatics ซึ่งเป็นศาสตร์ของการประยุกต์ใช้เทคนิคเชิงคำนวณต่าง ๆ ในสาขาสถิติ และสาขาคอมพิวเตอร์ด้านการรู้จำแบบ และการเรียนรู้ของเครื่อง เพื่อวิเคราะห์และประมวลผลข้อมูลดีเอ็นเอ (Ouzounis & Valencia, 2003; Cohen, 2004; Wodehouse, 2006)

วัตถุประสงค์ของการวิเคราะห์ข้อมูลดีเอ็นเอมีได้หลากหลาย ทั้งเพื่อศึกษาโครงสร้างและหน้าที่ของยีน (genomics) ศึกษาโครงสร้างและหน้าที่ของโปรตีน (proteomics) ที่เป็นผลผลิตของกระบวนการถอดรหัสดีเอ็นเอ รวมถึงเพื่อศึกษากระบวนการระหว่างกลางของการถอดรหัสจากดีเอ็นเอเป็นโปรตีน โครงการวิจัยนี้เน้นที่การศึกษาในส่วนที่บรรจุรหัสพันธุกรรมเพื่อการสังเคราะห์โปรตีน โดยกำหนดขอบเขตของงานเพื่อการสังเคราะห์โมเดลเชิงคำนวณให้สามารถรู้จำและจำแนกส่วนบรรจุรหัส (exon) ออกจากส่วนที่ไม่มีรหัสสำหรับสร้างโปรตีน (intron) รวมถึงความสามารถในการจำแนกจุดเชื่อมต่อ (splice sites) ของเอ็กซอนและอินทรอน งานในลักษณะนี้เรียกว่าการทำนายยีน (gene prediction) หรือการรู้จำรูปแบบของยีน (gene recognition)

การทำนายยีน หรือการรู้จำรูปแบบของยีน เป็นปัญหาวิจัยที่สำคัญในงานชีวสารสนเทศ เนื่องจากความรู้ที่ได้จะเป็นพื้นฐานสำคัญของการวิเคราะห์ด้านอื่น เช่น ชนิด ปริมาณ และความสมบูรณ์ของโปรตีนที่จะได้จากกระบวนการสังเคราะห์ นักวิจัยได้พยายามคิดค้นเทคนิคต่าง ๆ เพื่อการทำนายยีนที่แม่นยำ มานานกว่าสองทศวรรษ เทคนิคเหล่านี้แบ่งได้เป็นสองกลุ่มใหญ่ (Mathe et al., 2002; Do & Choi, 2006; Flicek, 2007; Gross et al., 2007; Bandyopadhyay et al., 2008) คือ intrinsic (หรือ *ab initio*) และ extrinsic (หรือ homology-based)

เทคนิคการทำนายยีนในแบบ intrinsic หรือ ab initio จะใช้ข้อมูลลำดับนิวคลีโอไทด์และโคดอน เฉพาะที่ปรากฏในสายดีเอ็นเอที่เป็นเป้าหมายของการสังเคราะห์โมเดล เพื่อการทำนายตำแหน่งของยีน (Fickett, 1982; Gribskov et al., 1984; Staden, 1984) เทคนิคการวิเคราะห์มักจะใช้วิธีการทางสถิติเช่น hidden Markov model ซอฟต์แวร์เพื่อการทำนายยีนที่จัดอยู่ในกลุ่ม ab initio ได้แก่ GenScan (Burge & Karlin, 1997), Augustus (Stanke & Waack, 2003), TigrScan/Genzilla (Majoros et al., 2004) และ CRAIG (Bernal et al., 2007) ซอฟต์แวร์ที่มีชื่อเสียงและเป็นที่ยอมรับในกลุ่มนี้คือ GenScan เนื่องจากใช้งานได้ง่ายและประมวลผลรวดเร็ว ถึงแม้ในระยะหลังจะมีซอฟต์แวร์เพื่อการทำนายยีนที่ให้ค่าความแม่นยำสูงกว่า แต่ GenScan ยังได้รับความนิยมใช้เป็นเกณฑ์พื้นฐาน ในการเปรียบเทียบความสามารถในการทำนายกับซอฟต์แวร์ที่ได้รับการพัฒนาขึ้นมาใหม่

เทคนิคการทำนายยีนในกลุ่มที่สองคือ extrinsic หรือ homology-based หรือ similarity-based มีการใช้ข้อมูลอื่นนอกเหนือจากดีเอ็นเอเป้าหมายประกอบการทำนายยีน ข้อมูลอื่นที่ใช้ได้แก่ ลำดับการเรียงตัวของกรดอะมิโนในโปรตีน ลำดับของนิวคลีโอไทด์ในสายของเอ็มอาร์เอ็นเอ และรูปแบบการเรียงตัวของสายดีเอ็นเอ ซอฟต์แวร์เพื่อการทำนายในกลุ่ม extrinsic ได้แก่ ROSETTA (Batzoglou et al., 2000), CEM (Bafna & Husan, 2000), TWINSCAN (Korf et al., 2001), SLAM (Alexandersson et al., 2003), SGP (Parra et al., 2003), EvoGene (Pederson & Hein, 2003), ExoniPhy (Siepel & Hausler, 2004), N-SCAN (Gross & Brent, 2005), DOGFISH (Carter & Durbin, 2006) และ CONTRAST (Gross et al., 2007)

ซอฟต์แวร์เพื่อการทำนายยีนที่เกิดขึ้นในระยะหลัง เช่น CONTRAST แสดงให้เห็นว่า การใช้ข้อมูลอื่น เช่น รูปแบบการเรียงตัวของสายดีเอ็นเอ ประกอบการทำนายตำแหน่งของเอ็กซอน จะให้ผลการทำนายที่แม่นยำมากขึ้น ข้อมูลที่ใช้ประกอบการทำนาย นอกจากรูปแบบการเรียงตัวของสายดีเอ็นเอแล้ว ยังพบว่าโครงสร้างของอินทรอน และข้อมูลเกี่ยวกับโครงสร้างของอาร์เอ็นเอทั้งในระดับปฐมภูมิและทุติยภูมิสามารถนำมาช่วยในการทำนายยีนได้เป็นผลดี (Sparks & Brendel, 2005; Marashi et al., 2006; Rogic, 2006; Dogan et al., 2007)

โครงการวิจัยนี้มีวัตถุประสงค์ประสงค์ในการทำนายยีนด้วยวิธีการของ machine learning โดยใช้เทคนิคการค้นพบความสัมพันธ์ (association mining) ใช้ข้อมูลของ intron และ exon ในโครงสร้างของ RNA ประกอบการทำนาย splice sites ของยีน ข้อมูลประกอบการทำนายจะถูกออกแบบโครงสร้างและจัดรูปแบบการแทนข้อมูลให้เหมาะสมกับการทำหน้าที่เป็นข้อมูลประกอบการทำนาย โดยโมเดลที่ได้สามารถถูกแปลงให้อยู่ในกรอบของระบบฐานความรู้ที่จะง่ายต่อการใช้งาน นอกจากนี้การมี inference engine แยกจากส่วนฐานความรู้จะทำให้สะดวกต่อการปรับปรุงระบบเมื่อมีความรู้ใหม่เกิดขึ้นในอนาคต

ปัจจุบันระบบฐานความรู้ในงาน genomics ยังมีเพียงระบบ FIGENIX (Gouret et al., 2005) และ CASSIOPE (Rascol et al., 2009) ที่ใช้ในการเปรียบเทียบจีโนมของหลายดีเอ็นเอเพื่อค้นหาบริเวณที่มีลำดับนิวคลีโอไทด์ตรงกัน และระบบ AlexSys (Aniba et al., 2010) ที่ใช้ในการเปรียบเทียบและวิเคราะห์โปรตีน ระบบฐานความรู้เพื่อการทำนายยีนที่พัฒนาขึ้นนี้จึงจัดเป็นความก้าวหน้าใหม่สำหรับงานวิจัยด้าน genomics

1.3 วัตถุประสงค์ของโครงการวิจัย

- ค้นคว้าความรู้พื้นฐานเกี่ยวกับโครงสร้างอาร์เอ็นเอ อินทรอน และกระบวนการเชื่อมต่ออาร์เอ็นเอ ที่เหมาะสมสำหรับการพัฒนาเป็นความรู้พื้นฐานในการสร้างโมเดลเพื่อตรวจจับ splice sites บนสายดีเอ็นเอ รวมถึงออกแบบรูปแบบของการแทนความรู้ที่เหมาะสมกับงานพัฒนาโมเดลเพื่อการทำนายยีน
- พัฒนาโมเดลสำหรับการทำนายตำแหน่งเชื่อมต่อระหว่างส่วนเอ็กซอน-อินทรอน บนสายดีเอ็นเอที่ให้ค่าความแม่นยำในการทำนายสูง และให้ค่า false positive ต่ำ โดยรูปแบบของโมเดลที่ได้ จะต้องเหมาะสมสำหรับการแปลงเป็นฐานความรู้สำหรับงานด้านชีวสารสนเทศ

1.4 ขอบเขตของการวิจัย

งานวิจัยนี้เป็นการพัฒนาโมเดลเชิงคำนวณ และออกแบบระบบฐานความรู้สำหรับงานด้านการทำนายยีน เน้นที่การทำตำแหน่งเอ็กซอน-อินทรอน และขอบเขตระหว่างส่วนของเอ็กซอนและอินทรอน โดยในเบื้องต้นนี้ยังไม่รวมงานด้าน gene annotation ที่เป็นการหาจำนวนยีนและกำหนดตำแหน่งที่แน่นอนของยีนบนโครโมโซม โมเดลที่สร้างขึ้นจะพิจารณาโครงสร้างของดีเอ็นเอและอาร์เอ็นเอเป็นหลัก โดยยังไม่พิจารณาหน้าที่ หรือ function ของรหัสพันธุกรรม การทดสอบความแม่นยำของโมเดลจะใช้เทคนิคของการตรวจสอบด้วยข้อมูลทดสอบ ตามวิธีการปกติของ machine learning โดยจะไม่ใช้วิธีการตรวจสอบทางห้องปฏิบัติการชีววิทยาระดับโมเลกุล

1.5 ประโยชน์ที่ได้รับ

งานวิจัยนี้เป็นการพัฒนาวิธีการเพื่อให้ได้องค์ความรู้ใหม่ในด้านระบบฐานความรู้เพื่อการทำนายยีนหรือรหัสทางพันธุกรรม ประโยชน์ที่ได้รับโดยตรงคือเทคนิคและอัลกอริทึมใหม่ที่สามารถตีพิมพ์ผลงานวิจัยที่ประกอบด้วยผลการออกแบบระบบ การพัฒนาเทคนิค และอัลกอริทึมของการประมวลผล โดยตลอดระยะเวลาสามปีของการดำเนินงานโครงการวิจัยนี้ สามารถตีพิมพ์เผยแพร่

ผลงานวิจัยในวารสารวิชาการ 6 บทความ และนำเสนอผลงานวิจัยและตีพิมพ์ใน International Conference Proceedings อีก 2 บทความ

การทดสอบระบบที่ออกแบบและพัฒนาขึ้นในลักษณะ prototype ทำให้ได้โปรแกรมต้นแบบเพื่อการทำนายเงินที่สามารถจดลิขสิทธิ์ได้ 2 โปรแกรม ได้แก่ โปรแกรมการค้นพบรูปแบบที่ปรากฏบ่อยในข้อมูลดีเอ็นเอ (ลิขสิทธิ์เลขที่ ว1. 5346 ออกให้ ณ วันที่ 8 เมษายน 2558) และโปรแกรมแบบขนานเพื่อค้นหาความสัมพันธ์ (ลิขสิทธิ์เลขที่ ว1. 5347 ออกให้ ณ วันที่ 8 เมษายน 2558) นอกจากนี้ผู้ช่วยวิจัยที่เป็นนักศึกษาในระดับบัณฑิตศึกษา ทั้งระดับปริญญาโทและปริญญาเอก ได้มีโอกาสมีส่วนร่วมในโครงการวิจัยนี้เพื่อพัฒนาความสามารถในการทำงานวิจัยระดับสูง

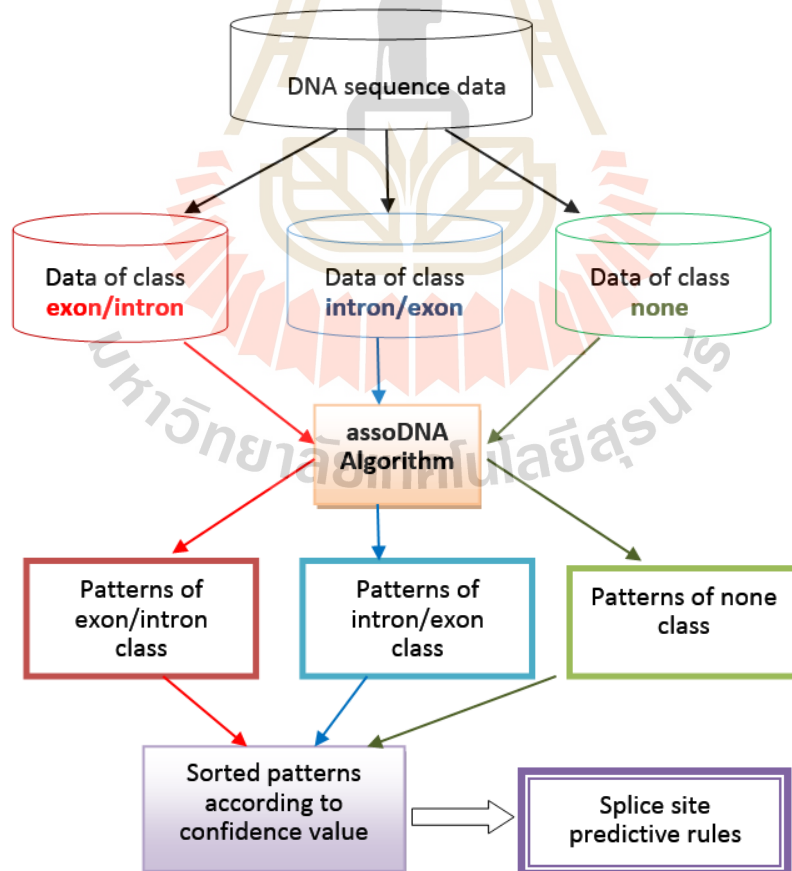


บทที่ 2

การออกแบบและพัฒนาโปรแกรม

2.1 กรอบแนวคิด

โครงการวิจัยนี้มีวัตถุประสงค์หลักเพื่อสร้างโมเดลสำหรับทำนาย splice sites หรือจุดเชื่อมต่อ exon-intron/intron-exon ในสายดีเอ็นเอ การสร้างโมเดลจะใช้วิธีการเรียนรู้รูปแบบจากข้อมูลฝึก เมื่อได้โมเดลเพื่อการทำนายตำแหน่งจุดเชื่อมต่อ หรือ splice sites ที่ผ่านการทดสอบความแม่นยำในการทำนายแล้ว โมเดลจะถูกแปลงเป็นข้อมูลในฐานความรู้เพื่อใช้ประโยชน์ในงานชีวสารสนเทศ เนื่องจากโมเดลทำนาย splice sites จะต้องเอื้อต่อการแปลงเป็นฐานความรู้ในขั้นตอนต่อไป งานวิจัยนี้จึงเลือกใช้เทคนิคการค้นพบความสัมพันธ์ (association mining) เป็นอัลกอริทึมพื้นฐานสำหรับการสร้างโมเดล เนื่องจากให้ผลลัพธ์เป็นความสัมพันธ์ที่สามารถแปลงเป็นข้อความในลักษณะกฎของฐานความรู้ได้ โครงสร้างของโปรแกรมการค้นพบรูปแบบที่ปรากฏอยู่ในข้อมูลดีเอ็นเอสำหรับทำนายจุดเชื่อมต่อในสายดีเอ็นเอ (เรียกชื่อย่อว่าโปรแกรม assoDNA) แสดงได้ดังรูปที่ 2.1



รูปที่ 2.1 แนวคิดของโปรแกรม assoDNA สำหรับทำนายจุดเชื่อมต่อในดีเอ็นเอ

ขั้นตอนการทำงานหลักของโปรแกรมการค้นหารูปแบบที่ปรากฏบ่อยสำหรับทำนายจุดเชื่อมต่อในสายดีเอ็นเอ (โปรแกรม assoDNA) จะเริ่มต้นด้วยการนำเข้าสู่ข้อมูลสายดีเอ็นเอเพื่อเป็นข้อมูลฝึกสำหรับให้โปรแกรมเรียนรู้ กระบวนการเรียนรู้มีวัตถุประสงค์เพื่อค้นหาแบบของ splice site ที่เป็นจุดเชื่อมต่อของ exon และ intron หรือจุดเชื่อมต่อระหว่าง intron และ exon เมื่อนำเข้าสู่ชุดข้อมูลฝึกแล้วจัดแบ่งข้อมูลเป็น ๓ กลุ่ม คือกลุ่มแรกเป็นสายดีเอ็นเอที่ปรากฏจุดเชื่อมต่อ exon-intron (เรียกว่า donor site) กลุ่มที่สองเป็นสายดีเอ็นเอที่ปรากฏจุดเชื่อมต่อ intron-exon (เรียกว่า acceptor site) และกลุ่มที่สามเป็นสายดีเอ็นเอที่ไม่ปรากฏทั้ง donor และ acceptor sites

จากนั้นข้อมูลในแต่ละกลุ่มจะถูกนำเข้าสู่โปรแกรม assoDNA เพื่อเรียนรู้รูปแบบที่ปรากฏบ่อย และแสดงผลลัพธ์เป็นรูปแบบที่ปรากฏบ่อยที่สุด (support) เรียงลดหลั่นตามลำดับค่าความถี่ที่ปรากฏบ่อยจากสูงไปต่ำ และเรียงลำดับตามค่าความเชื่อมั่น (confidence) ซึ่งมักจะใช้เป็นมาตรวัดแทนค่าความถูกต้องของรูปแบบที่ค้นพบ ในงานวิจัยนี้จะพิจารณาเลือกใช้รูปแบบที่ปรากฏบ่อยและความเชื่อมั่นสูงในช่วง ๓-๕ ลำดับแรกเพื่อแปลงเป็นข้อมูลเก็บไว้ในฐานความรู้ รูปแบบของ splice sites ที่บันทึกไว้ในฐานความรู้จะถูกนำไปใช้งานเพื่อการทำนายจุดเชื่อมต่อสายดีเอ็นเอ สำหรับกรณีที่ไม่ทราบแน่ชัดว่าสายดีเอ็นเอไหนปรากฏ donor site หรือ acceptor site หรือไม่

2.2 การออกแบบและพัฒนาโปรแกรม

การออกแบบโปรแกรม assoDNA ใช้แนวทางการค้นหารูปแบบที่ปรากฏบ่อยของอัลกอริทึม Apriori (Agrawal and Srikant, 1993) เป็นพื้นฐานในการพัฒนา ขั้นตอนวิธีของโปรแกรม assoDNA แสดงได้ในลักษณะของ pseudocode ดังรูปที่ 2.2 และแสดงในลักษณะของ flowchart ดังรูปที่ 2.3

ขั้นตอนแรกของโปรแกรม assoDNA จะเป็นการแบ่งข้อมูลออกเป็นสามกลุ่มย่อย ได้แก่ exon/intron, intron/exon, none เพื่อแทนกลุ่มของข้อมูลดีเอ็นเอที่ปรากฏจุดเชื่อมต่อระหว่าง exon-intron กลุ่มของข้อมูลดีเอ็นเอที่ปรากฏจุดเชื่อมต่อระหว่าง intron-exon และกลุ่มของข้อมูลดีเอ็นเอที่ไม่ปรากฏจุดเชื่อมต่อ ตามลำดับ

ขั้นตอนที่สอง เป็นการค้นหารูปแบบของนิวคลีโอไทด์ในแต่ละกลุ่มโดยเริ่มจากรูปแบบที่ประกอบด้วย 1 นิวคลีโอไทด์ และเพิ่มจำนวนนิวคลีโอไทด์ยาวขึ้นเป็น 2, 3, 4, 5, ..., 60 และนับค่าความถี่ของรูปแบบเหล่านั้น ค่าความถี่นี้เรียกว่าค่าสนับสนุน (support) รูปแบบของนิวคลีโอไทด์ที่ปรากฏบ่อยจะถูกคัดเลือกเพื่อส่งต่อไปสู่การทำงานในขั้นตอนต่อไป ในการใช้งานโปรแกรมค่าความถี่ขั้นต่ำหรือค่าสนับสนุนขั้นต่ำ (minimum support, minSup) จะระบุโดยผู้ใช้งานโปรแกรม

ขั้นตอนที่สามของโปรแกรม เป็นการคำนวณค่าความเชื่อมั่น (confidence) ของแต่ละรูปแบบนิวคลีโอไทด์ รูปแบบที่ผ่านเกณฑ์ความเชื่อมั่นขั้นต่ำ (minimum confidence, minConf) จะถูกคัดเลือกและส่งต่อไปยังขั้นตอนต่อไป ค่า minConf จะระบุโดยผู้ใช้งานโปรแกรม

ขั้นตอนที่สี่และห้า เป็นการนำผลลัพธ์รูปแบบนิพจน์ไอเทดที่ปรากฏบ่อยและมีค่าความเชื่อมั่นสูงแสดงเป็นกฎเพื่อบันทึกในฐานความรู้

Step 1: Initialization phase

Split the training dataset into three subsets according to the class value. Thus, we will get data of class exon/intron, data of class intron/exon, and data of class none.

Step 2: Generation of frequent patterns

Each data subset is processed through the following steps:

- 2.1 Set the given minimum support as minSup
- 2.2 Initialize R (a set of frequent patterns) to be empty, $R = \emptyset$
- 2.3 Build a candidate pattern P of length K

$$P = \bigwedge_K (L_i = B_j)$$
 where K starts from 1, $i \in \{-30, \dots, +30\}$, and $B_j \in \{A,C,T,G,D,N,S,R\}$
- 2.4 Select a pattern P with support \geq minSup to contain in a set S
- 2.5 Set $R = R \cup S$
- 2.6 If $S = \emptyset$, then continue to step 3
else increment K and go back to step 2.3

Step 3: Confidence computation

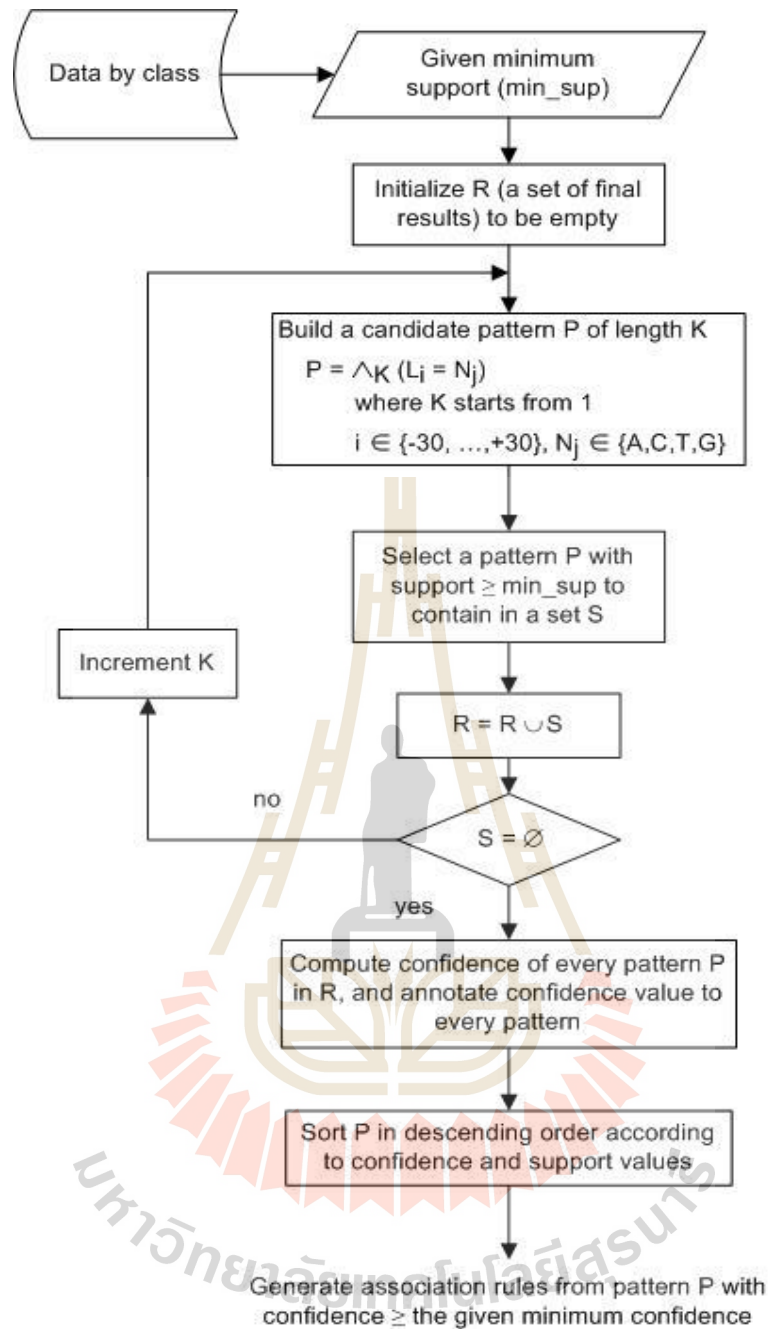
- 3.1 Compute confidence value of every pattern P in R, and annotate confidence value to every pattern
- 3.2 Sort P in descending order according to confidence value, for a tie then descending sort with respect to a support value

Step 4: Rule generation

- 4.1 Set the given minimum confidence as minConf
- 4.2 Generate association rules from every pattern P in R that has confidence \geq minConf

Step 5: Building predictor model

Combine rules from the process of every data subset and sort according their confidence and support values



รูปที่ 2.3 การทำงานของโปรแกรม assoDNA แสดงในลักษณะ flowchart

การพัฒนาขั้นตอนวิธีดังแสดงในรูปที่ 2.2 และ 2.3 เป็นโปรแกรมจะใช้ภาษาเออแลง (Erlang) ซึ่งเป็นภาษาเชิงฟังก์ชัน (ดาวน์โหลดได้ฟรีจาก <http://www.erlang.org>) เนื่องจากเป็นภาษาที่เหมาะสมสำหรับการพัฒนาต้นแบบอย่างรวดเร็ว (rapid prototyping) และสามารถพัฒนาเป็นการโปรแกรมแบบขนานได้ง่าย ตัวอย่างโปรแกรมบางส่วนแสดงได้ดังรูปที่ 2.4 (รหัสต้นฉบับทั้งหมดของโปรแกรม assoDNA สามารถดูได้ในภาคผนวก ข) และหน้าจอฟลลัพท์ของการประมวลผลโปรแกรมแสดงดังรูปที่ 2.5

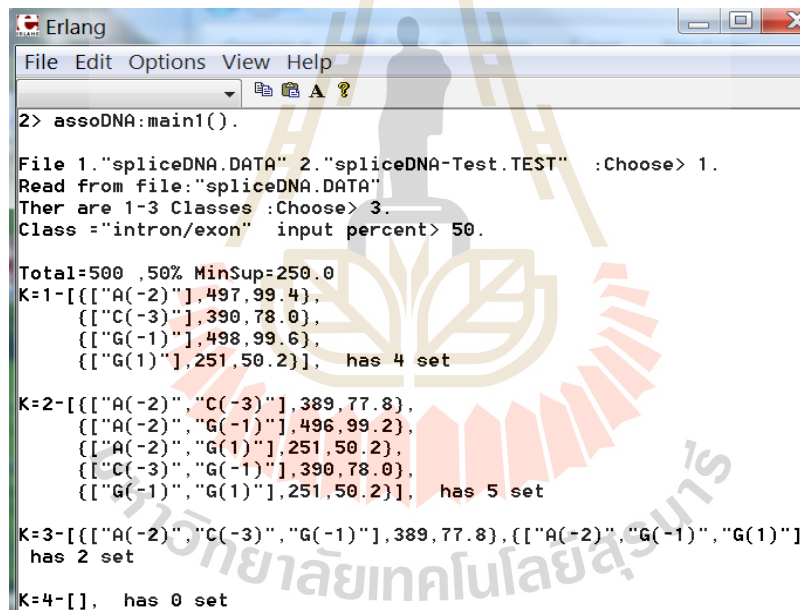
```

main1() ->
  {AllInput,FNo,ThisClass} = input(),
  DB = myToSet(AllInput),
  Total = length(AllInput),
  {_,Per} = io:read(" input percent> "),
  {FNo, ThisClass, DB, Per} .

apriori(DB, Items, Min) ->
  C1=[ {from_list([X]), findSup(from_list([X]), DB) } || X <- Items ],
  L1=[{FS,Sup} || {FS,Sup} <- C1,Sup>=Min] ,
  LkPrint=[ {to_list(FS), Sup,Sup/length(DB)*100} || {FS,Sup} <- L1] ,
  K = 2,
  LS = [FS || {FS,_} <- L1],
  aprioriLoopPar(L1, DB, LS, K, Min) .

```

รูปที่ 2.4 ฟังก์ชันหลักและฟังก์ชันค้นหาความสัมพันธ์ของโปรแกรม assoDNA



```

Erlang
File Edit Options View Help
2> assoDNA:main1().

File 1."spliceDNA.DATA" 2."spliceDNA-Test.TEST" :Choose> 1.
Read from file:"spliceDNA.DATA"
There are 1-3 Classes :Choose> 3.
Class ="intron/exon" input percent> 50.

Total=500 ,50% MinSup=250.0
K=1-[[["A(-2)"],497,99.4],
      [{"C(-3)"],390,78.0},
      [{"G(-1)"],498,99.6},
      [{"G(1)"],251,50.2}], has 4 set

K=2-[[["A(-2)","C(-3)"],389,77.8],
      [{"A(-2)","G(-1)"],496,99.2},
      [{"A(-2)","G(1)"],251,50.2},
      [{"C(-3)","G(-1)"],390,78.0},
      [{"G(-1)","G(1)"],251,50.2}], has 5 set

K=3-[[["A(-2)","C(-3)","G(-1)"],389,77.8], [{"A(-2)","G(-1)","G(1)"]
has 2 set

K=4-[], has 0 set

```

รูปที่ 2.5 ตัวอย่างจอภาพที่เป็นผลการทำงานของโปรแกรม assoDNA

ผลการทำงานของโปรแกรม assoDNA ดังรูปที่ 2.5 เป็นการค้นหารูปแบบของนิวคลีโอไทด์ที่ปรากฏบ่อยในกลุ่มของดีเอ็นเอที่มีจุดเชื่อมต่อ intron-exon โดยกำหนดค่าความถี่ของการปรากฏบ่อย หรือ minSup เป็น 50% รูปแบบที่ถึงเกณฑ์ขั้นต่ำนี้มีทั้งสิ้น 11 รูปแบบ ดังนี้

Intron-exon splice site patterns:

รูปแบบที่ 1 นิวคลีโอไทด์ A ปรากฏที่ตำแหน่ง -2 (confidence = 99.4%)

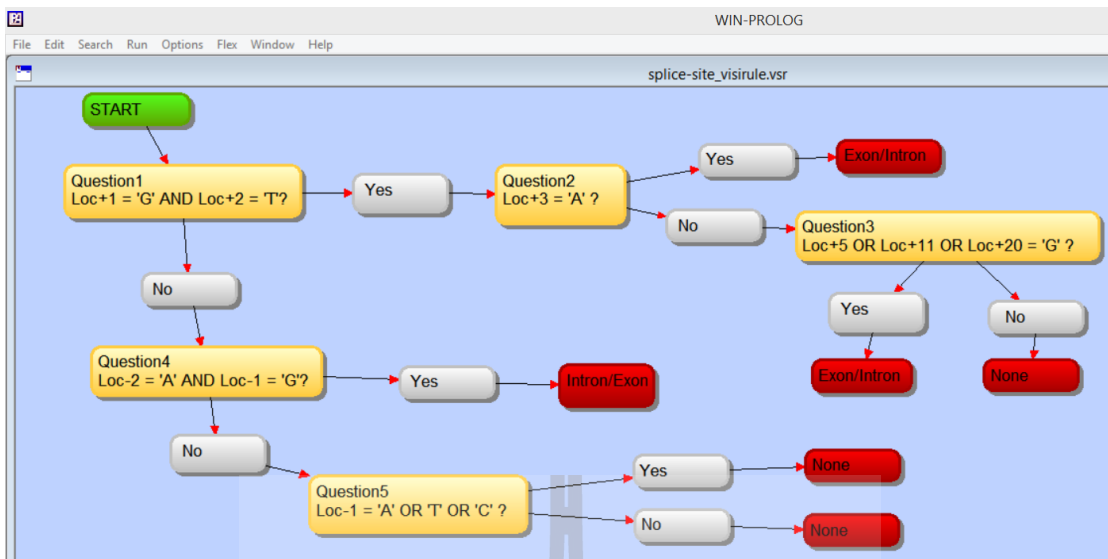
รูปแบบที่ 2 นิวคลีโอไทด์ C ปรากฏที่ตำแหน่ง -3 (confidence = 78.0%)

- รูปแบบที่ 3 นิวคลีโอไทด์ G ปรากฏที่ตำแหน่ง -1 (confidence = 99.6%)
- รูปแบบที่ 4 นิวคลีโอไทด์ G ปรากฏที่ตำแหน่ง +1 (confidence = 50.2%)
- รูปแบบที่ 5 นิวคลีโอไทด์ A ปรากฏที่ตำแหน่ง -2 และ
นิวคลีโอไทด์ C ปรากฏที่ตำแหน่ง -3 (confidence = 77.8%)
- รูปแบบที่ 6 นิวคลีโอไทด์ A ปรากฏที่ตำแหน่ง -2 และ
นิวคลีโอไทด์ G ปรากฏที่ตำแหน่ง -1 (confidence = 99.2%)
- รูปแบบที่ 7 นิวคลีโอไทด์ A ปรากฏที่ตำแหน่ง -2 และ
นิวคลีโอไทด์ G ปรากฏที่ตำแหน่ง +1 (confidence = 50.2%)
- รูปแบบที่ 8 นิวคลีโอไทด์ C ปรากฏที่ตำแหน่ง -2 และ
นิวคลีโอไทด์ G ปรากฏที่ตำแหน่ง -1 (confidence = 78.0%)
- รูปแบบที่ 9 นิวคลีโอไทด์ G ปรากฏที่ตำแหน่ง -1 และ
นิวคลีโอไทด์ G ปรากฏที่ตำแหน่ง +1 (confidence = 50.2%)
- รูปแบบที่ 10 นิวคลีโอไทด์ A ปรากฏที่ตำแหน่ง -2 และ
นิวคลีโอไทด์ C ปรากฏที่ตำแหน่ง -3 และ
นิวคลีโอไทด์ G ปรากฏที่ตำแหน่ง -1 (confidence = 77.8%)
- รูปแบบที่ 11 นิวคลีโอไทด์ A ปรากฏที่ตำแหน่ง -2 และ
นิวคลีโอไทด์ G ปรากฏที่ตำแหน่ง -1 และ
นิวคลีโอไทด์ G ปรากฏที่ตำแหน่ง +1 (confidence = 50.2%)

2.3 การแปลงโมเดลเป็นฐานความรู้

โมเดลสำหรับรูปแบบเชื่อมต่อ exon-intron รูปแบบเชื่อมต่อ intron-exon และรูปแบบที่ไม่มีจุดเชื่อมต่อ (none) ที่ได้จากโปรแกรม assoDNA จะถูกแปลงเป็นฐานความรู้โดยการใส่โปรแกรมเชิงตรรกะชื่อ Win-prolog (<http://www.lpa.co.uk/win.htm>) และโปรแกรมเชิงภาพชื่อ VisiRule (<http://www.lpa.co.uk/vsr.htm>) ช่วยในการสร้างฐานความรู้ แผนภาพของการสร้างฐานความรู้แสดงได้ดังรูปที่ 2.6 เมื่อประมวลผลแผนภาพจะได้ชุดคำสั่งในภาษาเชิงตรรกะดังรูปที่ 2.7

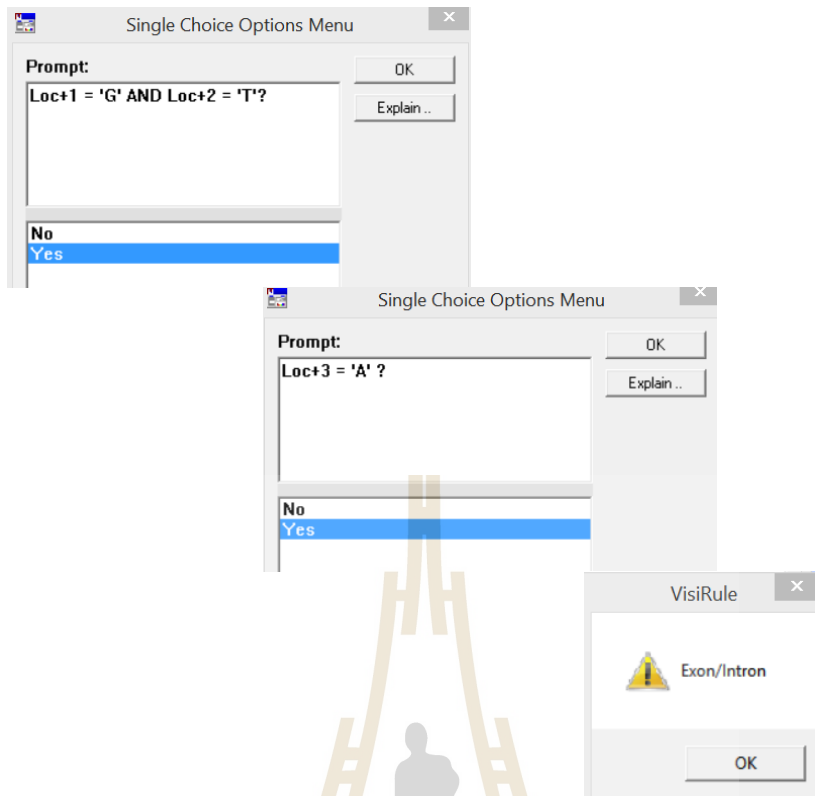
การใช้งานฐานความรู้ จะเป็นการประมวลผลโปรแกรมเชิงตรรกะในรูปที่ 2.7 การใช้งานโปรแกรมจะเป็นลักษณะโต้ตอบ โดยโปรแกรมจะตั้งคำถามให้ผู้ใช้ตอบ คำตอบจะเป็นสองทางเลือกคือ yes/no ตัวอย่างการถามตอบเกี่ยวกับตำแหน่งที่ปรากฏนิวคลีโอไทด์แบบต่าง ๆ ในสายดีเอ็นเอแล้วให้ผลลัพธ์เป็นรูปแบบเชื่อมต่อแบบ exon-intron แสดงดังรูปที่ 2.8 การโต้ตอบกับฐานความรู้แล้วให้ผลลัพธ์เป็นรูปแบบเชื่อมต่อ intron-exon แสดงดังรูปที่ 2.9 และการโต้ตอบที่ให้ผลลัพธ์เป็นไม่ปรากฏตำแหน่งเชื่อมต่อในสายดีเอ็นเอ แสดงดังรูปที่ 2.10



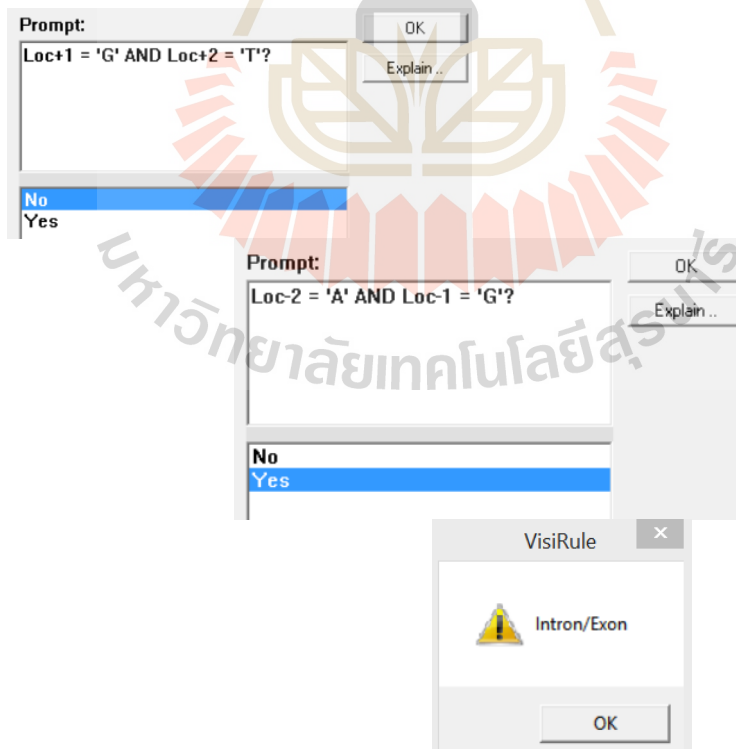
รูปที่ 2.6 การแปลงโมเดลเป็นฐานความรู้ด้วยโปรแกรมเชิงภาพ VisiRule

<pre> do ensure_loaded(system(vrllib)). relation 'START'(Conclusion) if q_Question1(Conclusion) . relation q_Question1(Conclusion) if the answer to 'Question1' is _ and check('Question1', =, 'No') and q_Question4(Conclusion) . relation q_Question1(Conclusion) if the answer to 'Question1' is _ and check('Question1', =, 'Yes') and q_Question2(Conclusion) . relation q_Question4(Conclusion) if the answer to 'Question4' is _ and check('Question4', =, 'No') and q_Question5(Conclusion) . relation q_Question4(Conclusion) if the answer to 'Question4' is _ and check('Question4', =, 'Yes') and Conclusion = 'Intron/Exon' . relation q_Question5(Conclusion) if the answer to 'Question5' is _ and check('Question5', =, 'Yes') and Conclusion = 'None' . relation q_Question5(Conclusion) if the answer to 'Question5' is _ and check('Question5', =, 'No') and Conclusion = 'None' . relation q_Question2(Conclusion) if the answer to 'Question2' is _ and check('Question2', =, 'No') and q_Question3(Conclusion) . relation q_Question2(Conclusion) if the answer to 'Question2' is _ and </pre>	<pre> check('Question2', =, 'Yes') and Conclusion = 'Exon/Intron' . relation q_Question3(Conclusion) if the answer to 'Question3' is _ and check('Question3', =, 'Yes') and Conclusion = 'Exon/Intron' . relation q_Question3(Conclusion) if the answer to 'Question3' is _ and check('Question3', =, 'No') and Conclusion = 'None' .group group1 'No', 'Yes' . question 'Question2' 'Loc+3 = "A" ?' ; choose one of group1 because " . group group2 'Yes', 'No' . question 'Question3' 'Loc+5 OR Loc+11 OR Loc+20 = "G" ?' ; choose one of group2 because " . question 'Question4' 'Loc-2 = "A" AND Loc-1 = "G"?~M~J' ; choose one of group1 because " . question 'Question5' 'Loc-1 = "A" OR "T" OR "C" ?' ; choose one of group2 because ' ' . question 'Question1' 'Loc+1 = "G" AND Loc+2 = "T"?' ; choose one of group1 because " . </pre>
--	---

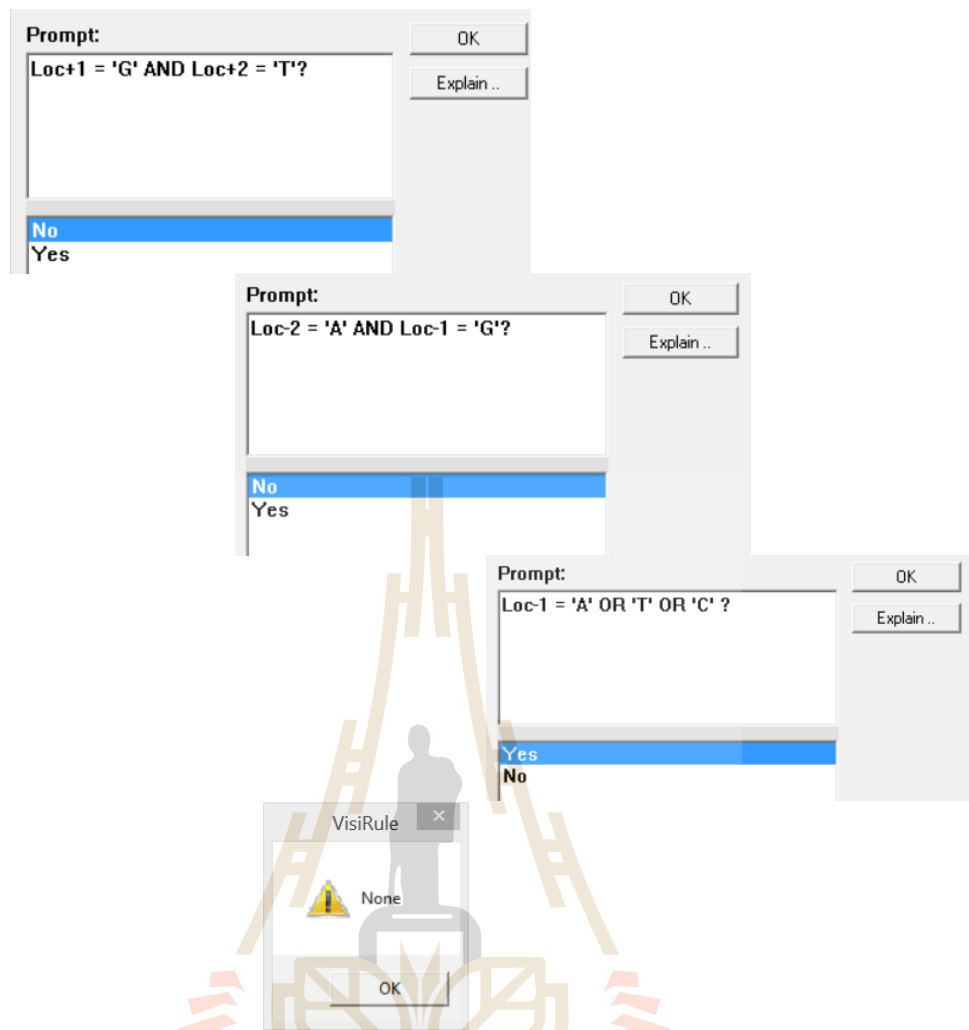
รูปที่ 2.7 ชุดคำสั่งภาษาเชิงตรรกะที่ได้จากการประมวลผลภาพในรูปที่ 2.6



รูปที่ 2.8 การโต้ตอบกับฐานความรู้แล้วให้ผลวินิจฉัยเป็นรูปแบบเชื่อมต่อ exon-intron



รูปที่ 2.9 การโต้ตอบกับฐานความรู้แล้วให้ผลวินิจฉัยเป็นรูปแบบเชื่อมต่อ intron-exon



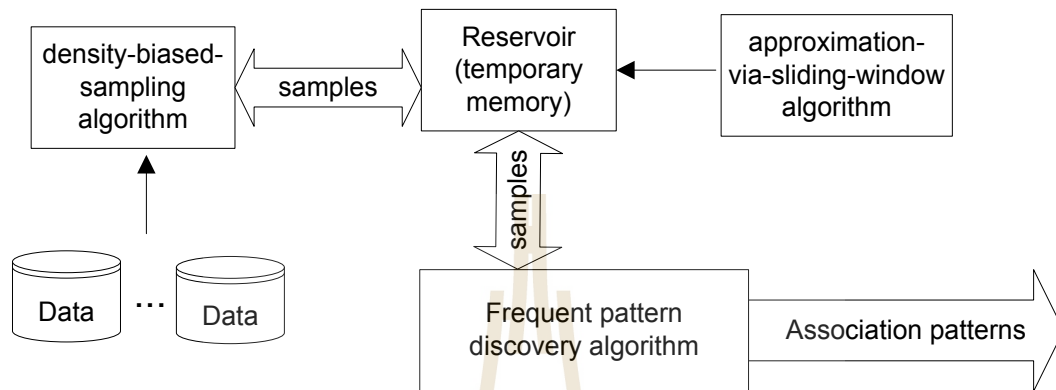
รูปที่ 2.10 การโต้ตอบกับฐานความรู้แล้วให้ผลวินิจฉัยว่าไม่ปรากฏรูปแบบเชื่อมต่อ

2.4 การจัดการความไม่แน่นอนด้วยวิธีการค้นพบโดยประมาณ

ในกรณีที่ข้อมูลนำเข้าไม่สมบูรณ์ โดยอาจมีบางตำแหน่งของสายดีเอ็นเอที่ไม่ทราบรหัสพันธุกรรมที่ถูกต้อง หรือในกรณีของการค้นหาจุดเชื่อมต่อในฐานข้อมูลดีเอ็นเอที่มีขนาดใหญ่มาก ขั้นตอนการค้นหาด้วยวิธีปกติจะใช้เวลาประมวลผลและทรัพยากรหน่วยความจำสูงมาก การใช้วิธีการค้นพบโดยประมาณจะช่วยแก้ปัญหาข้อมูลไม่สมบูรณ์และปัญหาข้อมูลมีขนาดใหญ่ได้ แนวคิดพื้นฐานของการค้นพบโดยประมาณคือใช้การสุ่มเลือกข้อมูลเพื่อนำมาประมวลผลด้วยโปรแกรม assoDNA

แนวทางการสุ่มที่นำเสนอในงานวิจัยนี้คือการสุ่มตามความหนาแน่นของข้อมูลในแต่ละคลาส ได้แก่ คลาส exon-intron คลาส intron-exon และคลาส none ในกรณีของข้อมูลขนาดใหญ่ การสุ่มจะใช้วิธีสุ่มจากกรอบข้อมูลขนาดเล็ก จากนั้นเลื่อนกรอบข้อมูลไปตามลำดับในลักษณะของวิธีการเลื่อนกรอบหน้าต่าง (sliding window) ข้อมูลภายในกรอบที่มีความหนาแน่นถึงเกณฑ์ที่กำหนด จะถูกสุ่มเลือกมาเก็บภายในพื้นที่หน่วยความจำชั่วคราวที่เรียกว่าแหล่งกักเก็บ (reservoir)

ข้อมูลในแหล่งกักเก็บนี้เท่านั้น ที่จะถูกนำไปค้นหารูปแบบด้วยโปรแกรม assoDNA แนวคิดของการสุ่มตามความหนาแน่นของข้อมูลเพื่อรวบรวมข้อมูลที่หนาแน่นถึงเกณฑ์ที่กำหนด บันทึกไว้ในแหล่งกักเก็บแสดงเป็นแผนภาพได้ดังรูปที่ 2.11 และขั้นตอนการทำงานในลักษณะของ pseudocode แสดงได้ดังรูปที่ 2.12



รูปที่ 2.11 กรอบแนวคิดของการสุ่มเลือกข้อมูลตามความหนาแน่นเพื่อบันทึกไว้ในแหล่งกักเก็บ

Input: a set of data points represented as vectors

Output: a new set of transformed data points annotated with density value

% Initialize windows

- (1) Interact with user to obtain dimension value
 - (2) Generate window grid of size W along dimension axes
- % Count density*
- (3) Sequential move on each window and count number of data points, N , in the window
 - (4) Record a list of window's central point and its N value in a file F
 - (5) Return F as a set of transformed data
-

รูปที่ 2.12 ขั้นตอนวิธีของการสุ่มเลือกข้อมูลตามความหนาแน่นด้วยการเลื่อนกรอบหน้าต่าง

2.5 การเพิ่มประสิทธิภาพโปรแกรมด้วยการทำงานแบบขนาน

ในกรณีที่ใช้งานโปรแกรม assoDNA ต้องการเพิ่มความเร็วในการประมวลผล สามารถเลือกใช้โปรแกรมที่มีการทำงานแบบขนาน ภาษาเอแอลงอำนวยความสะดวกเกี่ยวกับการประมวลผล โปรแกรมแบบขนานด้วยการใช้วิธีการสร้างหลายโพรเซส โดยโปรแกรมเมอร์จะเป็นผู้พิจารณาและกำหนดว่าโพรเซสสำหรับการประมวลผลในโปรแกรมมีฟังก์ชันใดที่สามารถทำงานพร้อมกันได้ ถ้าปรากฏฟังก์ชันที่มีลักษณะดังกล่าว สามารถใช้คำสั่ง spawn เพื่อสร้างโพรเซสใหม่ให้ทำงานคู่ขนานแบบพร้อมกันกับโพรเซสอื่น ชุดคำสั่งของโปรแกรม assoDNA ที่ถูกปรับปรุงให้ทำงานแบบขนานแสดงได้ดังรูปที่ 2.13 และตัวอย่างจอภาพของการประมวลผลแบบขนานแสดงได้ดังรูปที่ 2.14

```
-module(assoDNA_par).
concurrent(P1, P2, P3) ->
    spawn(assoDNA_par, run, [self(),P1]),
    spawn(assoDNA_par, run, [self(),P2]),
    spawn(assoDNA_par,run,[self(),P3]),
    receive
        my_end -> ok
    end.

run(MasterID, InputL) ->
    R = main2(any, 3, InputL),
    file:delete("out.txt"),
    AD = lists:last(R),
    [ADD_] = AD,
    Rules = lists:sublist(R, length(R)-1),
    PrintRules = map(fun( {D, S, Per, Class} ) ->
        {to_Col3(notLast(D)), S, Per, transformBack(Class) }
        end, Rules),
    ADP = lists:map(fun(Data) -> {Data, checkRules(Data, Rules) }
        end, AD),
    ADPprint = map(fun({Data, V}) -> Predict = transformBack(V),
        {Data, [last(Data), Predict,
            mark(last(Data), Predict) ] }
        end, ADP),
    Predict = map(fun( {F, S} ) -> {to_Col3(notLast(F)), S}
        end, ADPprint),
    writeToFile(Predict),
    [_,Stop_] = InputL,
    if Stop ==2 -> MasterID ! my_end ;
        true -> MasterID ! not_end
    end.
```

รูปที่ 2.13 ชุดคำสั่งภาษาเอแอลงสำหรับประมวลผลโปรแกรม assoDNA แบบขนาน

```

Erlang
File Edit Options View Help
3> f(), {T,_}=timer:tc(assoDNA_par,concurrent,[[1,1,80],[1,2,80],[1,3,80]]).

=====Read from file:"spliceDNA.DATA"=====
=====Read from file:"spliceDNA.DATA"=====
=====Read from file:"spliceDNA.DATA"=====
Ther are 1-3 ClassesClass ="exon/intron"
----START---Apriori(in class=2,Min Support80%=400.0)---
Ther are 1-3 Classes
Ther are 1-3 ClassesClass ="none" Class ="intron/exon"
----START---Apriori(in class=3,Min Support80%=400.0)---
----START---Apriori(in class=1,Min Support80%=800.0)---
K=2-[[["G(1)","G(5)"],427,85.39999999999999],
      [{"G(1)","T(2)"],494,98.8},
      [{"G(5)","T(2)"],424,84.8}], has 3 set

K=3-[[["G(1)","G(5)","T(2)"],424,84.8)], has 1 set

[["TQ"],494),
 [{"GP"},499),
 [{"GT"},427),
 [{"GP","GT"},427),
 [{"GP","TQ"},494),
 [{"GT","TQ"},424),
 [{"GP","GT","TQ"},424]]

K=2-[[["A(-2)","G(-1)"],496,99.2)], has 1 set

[["AM"],497),[["GN"],498),[["AM","GN"],496)]
{5304000,ok}

```

รูปที่ 2.14 จอภาพแสดงการประมวลผลโปรแกรม assoDNA แบบขนาน

การแสดงรูปแบบของจุดเชื่อมต่อของโปรแกรม assoDNA แบบขนาน (รูปที่ 2.14) แสดงตำแหน่งของนิวคลีโอไทด์ทั้งในแบบที่ใช้ตัวเลขและแบบที่ใช้รหัส ASCII แทนตัวเลข เช่นกรณีของการใช้ตัวเลข "A(-2)" หมายถึง นิวคลีโอไทด์ A (or adenine) ณ ตำแหน่ง -2 ในสายของดีเอ็นเอ และกรณีที่ใช้รหัส ASCII เช่น "AM" หมายถึง นิวคลีโอไทด์ A ณ ตำแหน่ง 29 (รหัส ASCII ของตัวอักษร M) ในสายของดีเอ็นเอ

บทที่ 3

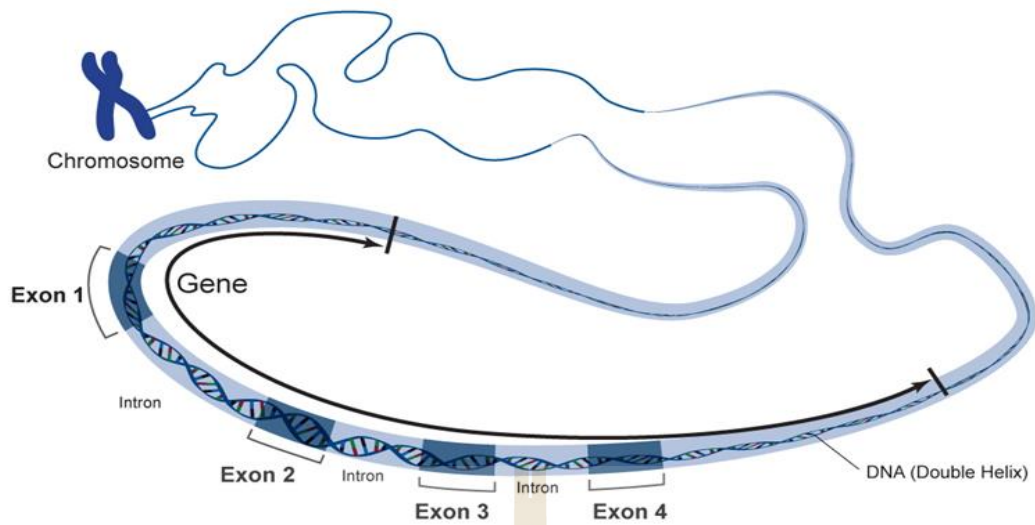
ผลการทดสอบโปรแกรม

โครงการวิจัยนี้มีวัตถุประสงค์หลักสองประการคือ (๑) ออกแบบวิธีการเพื่อให้ได้โมเดลหรือชุดของรูปแบบสำหรับตรวจจับ splice sites ในสายดีเอ็นเอให้ได้ผลของการตรวจจับอัตโนมัติที่มีค่า false positive ต่ำ และพัฒนาผลการออกแบบนั้นเป็นโปรแกรมการค้นพบรูปแบบที่ปรากฏบ่อยในข้อมูลดีเอ็นเอ หรือโปรแกรม assoDNA (๒) ใช้วิธีการทางวิศวกรรมความรู้แปลงโมเดลเพื่อการตรวจจับจุดเชื่อมต่อ หรือ splice sites เป็นโมเดลในรูปแบบที่สามารถใช้งานกับระบบฐานความรู้ได้ ขั้นตอนสำคัญของโครงการวิจัยนี้คือการออกแบบและพัฒนาโปรแกรม assoDNA ดังนั้นการทดสอบโปรแกรมจึงเป็นการทดสอบความถูกต้องของโมเดลในการทำนายจุดเชื่อมต่อในสายดีเอ็นเอ ด้วยชุดข้อมูลทดสอบ นอกจากนี้ยังมีการทดสอบประสิทธิภาพของโปรแกรมที่ค้นพบรูปแบบโดยวิธีประมาณและค้นพบรูปแบบด้วยเทคนิคการประมวลผลแบบขนาน ซึ่งเป็นเทคนิคเพิ่มเติมจากขั้นตอนหลัก

3.1 ข้อมูลที่ใช้ในการทดสอบ

ในการทดสอบความถูกต้องและประสิทธิภาพของโปรแกรม assoDNA ในการค้นหารูปแบบจุดเชื่อมต่อในสายดีเอ็นเอ งานวิจัยนี้ใช้ข้อมูลชื่อ splice junctions จากฐานข้อมูล UCI The Machine Learning Data Repository (2017) ของมหาวิทยาลัยแห่งแคลิฟอร์เนียเมืองเออไวน์ ข้อมูล splice junctions ชุดดั้งเดิมประกอบด้วย 3190 เรคคอร์ด แต่เนื่องจากมีข้อมูล 4 เรคคอร์ดที่รหัสพันธุกรรมในบางตำแหน่งกำกวม (มีความไม่แน่นอนว่าเป็นรหัส A หรือ G หรือ C หรือ T) ในงานวิจัยนี้จึงตัดข้อมูลทั้ง 4 เรคคอร์ดทิ้ง ทำให้คงเหลือข้อมูลที่ใช้ในการทดลองรวม 3186 เรคคอร์ด

ข้อมูลในแต่ละเรคคอร์ดประกอบด้วย 61 คอลัมน์ คอลัมน์ที่ 1-60 เป็นรหัสนิวคลีโอไทด์ ณ ตำแหน่งต่าง ๆ ของสายดีเอ็นเอ โดยสายดีเอ็นเอเริ่มต้นที่ตำแหน่ง -30 สิ้นสุดที่ตำแหน่ง +30 รหัสนิวคลีโอไทด์ในแต่ละตำแหน่งจะปรากฏเป็น A หรือ G หรือ T หรือ C คอลัมน์สุดท้ายของข้อมูลจะระบุชนิดของจุดเชื่อมต่อว่าเป็น exon/intron หรือ intron/exon หรือ none ซึ่งหมายถึงไม่ปรากฏจุดเชื่อมต่อทั้งสองแบบ รูปที่ 3.1 แสดงโครงสร้างของยีนที่ประกอบด้วย exon และ intron รูปที่ 3.2 แสดงตัวอย่างข้อมูลที่ใช้ในการทดสอบโปรแกรมจำนวน 5 เรคคอร์ด



รูปที่ 3.1 โครงสร้างของยีนที่ประกอบด้วยส่วน exon และ intron
(ที่มาจาก <http://genome.gov/Glossary/>)

```

% Data : A sequence of DNA starting at position -30 and ending at position +30.
% Problem statement :
%     Given a sequence of DNA, recognize the exon/intron (donor site)
%     and intron/exon (acceptor site) boundaries.
% Data examples :
T,T,C,T,A,T,G,A,G,A,A,A,C,G,T,G,G,C,A,T,T,G,T,G,C,G,C,A,A,G,G,T,G,G,G,C,C,C,G,C,G,G,G,A,
    C,G,G,G,G,C,A,G,C,T,C,C,G,G,G, exon/intron
C,T,C,C,C,A,C,C,A,C,C,T,G,T,C,C,A,C,C,G,C,C,G,C,A,G,A,T,C,G,C,T,T,C,C,T,G,G,A,G,C,
    C,A,G,G,C,A,A,G,A,A,C,T,C,C,A, intron/exon
C,T,G,A,C,T,A,A,G,C,C,G,C,C,C,T,G,T,C,C,C,T,T,C,T,C,A,G,A,T,T,A,T,G,T,T,T,G,A,G,A,C,C,
    T,T,C,A,A,C,A,C,C,C,G,G,C,C, intron/exon
G,A,G,G,A,G,C,T,A,G,A,C,A,A,G,T,A,C,T,G,G,T,C,T,C,A,G,C,A,G,G,T,G,C,G,T,G,A,G,G,G,G,A,G,
    G,G,G,A,T,G,G,C,T,G,C,C,A,A,G,G, exon/intron
A,A,G,G,C,T,C,A,G,G,A,G,G,A,G,G,A,G,A,T,C,A,A,C,A,T,C,A,A,C,C,T,G,C,C,C,G,C,C,C,C,C,T,
    C,C,C,C,A,G,C,C,T,G,A,T,A,A, none

```

รูปที่ 3.2 ตัวอย่างข้อมูลที่ใช้ในการทดสอบโปรแกรม

ข้อมูลจำนวน 3186 เรคคอร์ดนี้ จะถูกแบ่งเป็นสองส่วนที่มีจำนวน 61 คอลัมน์เหมือนกัน ข้อมูลส่วนแรกประกอบด้วย 2000 เรคคอร์ด ใช้สำหรับเป็นข้อมูลฝึก (training data) เพื่อให้โปรแกรม assoDNA ใช้ในการเรียนรู้เพื่อค้นหารูปแบบของจุดเชื่อมต่อ exon/intron และ intron/exon ข้อมูลส่วนที่สองประกอบด้วย 1186 เรคคอร์ด ใช้สำหรับทดสอบ (test data) ความแม่นยำของโมเดลตรวจจับจุดเชื่อมต่อในสายดีเอ็นเอ การกระจายของข้อมูลดีเอ็นเอทั้งสามกลุ่มย่อย (หรือเรียกว่า คลาส) ในชุดข้อมูลฝึกและชุดข้อมูลทดสอบ สรุปได้ดังตารางที่ 3.1

ตารางที่ 3.1 การกระจายของข้อมูลในแต่ละคลาสของชุดข้อมูลฝึกและชุดข้อมูลทดสอบ

Class	Splice site data	Training data	Test data
exon/intron	767 (24%)	464 (23.20%)	303 (25.55%)
intron/exon	765 (24%)	485 (24.25%)	280 (23.61%)
none	1,654 (52%)	1,051 (52.55%)	603 (50.84%)
Total	3,186	2,000	1,186

ชุดข้อมูลเริ่มต้นมีสัดส่วนของข้อมูลในคลาส exon/intron คิดเป็น 24% สัดส่วนของข้อมูลในคลาส intron/exon คิดเป็น 24% และสัดส่วนของข้อมูลในคลาส none คิดเป็น 52% เมื่อแยกข้อมูลเป็นสองชุดย่อยเพื่อทำหน้าที่เป็นข้อมูลฝึกและชุดข้อมูลทดสอบ การสุ่มเลือกข้อมูลจึงใช้วิธีสุ่มแบบแบ่งเป็นชั้น (stratified sampling) เพื่อให้สัดส่วนของข้อมูลในทั้งสามคลาสในชุดข้อมูลฝึกและชุดข้อมูลทดสอบใกล้เคียงกับข้อมูลดั้งเดิมให้มากที่สุด

3.2 มาตรฐานประสิทธิภาพโมเดลในการทำนายจุดเชื่อมต่อในยีน

การพัฒนาโมเดลหรือชุดของรูปแบบที่ใช้ทำนายจุดเชื่อมต่อในยีนหรือสายของดีเอ็นเอว่าเป็น exon/intron หรือ intron/exon หรือไม่ปรากฏจุดเชื่อมต่อทั้งสองแบบ (none) เป็นวิธีการสร้างโมเดลแบบอัตโนมัติ โดยให้โปรแกรมเรียนรู้รูปแบบจากชุดข้อมูลฝึกที่มีผลเฉลยอยู่แล้วว่าดีเอ็นเอสายใดปรากฏจุดเชื่อมต่อแบบ exon/intron ดีเอ็นเอสายใดปรากฏจุดเชื่อมต่อแบบ intron/exon และดีเอ็นเอสายใดไม่ปรากฏจุดเชื่อมต่อทั้งสองแบบ โมเดลที่ได้จากการเรียนรู้อัตโนมัติแบบนี้มักจะมี ความผิดพลาดแฝงอยู่เนื่องจากความไม่สมบูรณ์แบบของข้อมูลฝึก จึงต้องมีการประเมินความแม่นยำในการทำนายของโมเดลโดยใช้ชุดข้อมูลทดสอบ เครื่องมือที่นิยมใช้ในการบันทึกผลการประเมินของโมเดลคือเมตริกซ์ที่มีชื่อเรียกเฉพาะว่า เมตริกซ์สับสน หรือ คอนฟิวชันเมตริกซ์ (confusion matrix) โครงสร้างของคอนฟิวชันเมตริกซ์ในกรณีข้อมูลจุดเชื่อมต่อในดีเอ็นเอที่มีสามคลาส แสดงได้ดังตารางที่ 3.2

ตารางที่ 3.2 โครงสร้างคอนฟิวชันเมตริกซ์สำหรับข้อมูลการทำนายจุดเชื่อมต่อในสายดีเอ็นเอ

	Predicted exon/intron	Predicted intron/exon	Predicted none	
Actual exon/intron	<i>a</i>	<i>b</i>	<i>c</i>	<i>actual_EI</i>
Actual intron/exon	<i>d</i>	<i>e</i>	<i>f</i>	<i>actual_IE</i>
Actual none	<i>g</i>	<i>h</i>	<i>i</i>	<i>actual_None</i>
	<i>predict_EI</i>	<i>predict_IE</i>	<i>predict_None</i>	

มาตรวัดประสิทธิภาพของโมเดล จะคำนวณจากค่าที่ปรากฏในแต่ละช่องของคอนฟิวชันเมตริกซ์ตามที่ปรากฏในตารางที่ 3.2 โดยค่าของตัวแปรในแต่ละช่องมีความหมายดังนี้

- a* หมายถึง จำนวนข้อมูลทดสอบที่เป็นคลาส exon/intron และโมเดลทำนายได้ถูกต้องว่าเป็น exon/intron
- b* หมายถึง จำนวนข้อมูลทดสอบที่เป็นคลาส exon/intron แต่โมเดลทำนายผิดว่าเป็น intron/exon
- c* หมายถึง จำนวนข้อมูลทดสอบที่เป็นคลาส exon/intron แต่โมเดลทำนายผิดว่าเป็น none
- d* หมายถึง จำนวนข้อมูลทดสอบที่เป็นคลาส intron/exon แต่โมเดลทำนายผิดว่าเป็น exon/intron
- e* หมายถึง จำนวนข้อมูลทดสอบที่เป็นคลาส intron/exon และโมเดลทำนายได้ถูกต้องว่าเป็น intron/exon
- f* หมายถึง จำนวนข้อมูลทดสอบที่เป็นคลาส intron/exon แต่โมเดลทำนายผิดว่าเป็น none
- g* หมายถึง จำนวนข้อมูลทดสอบที่เป็นคลาส none แต่โมเดลทำนายผิดว่าเป็น exon/intron
- h* หมายถึง จำนวนข้อมูลทดสอบที่เป็นคลาส none แต่โมเดลทำนายผิดว่าเป็น intron/exon
- i* หมายถึง จำนวนข้อมูลทดสอบที่เป็นคลาส none และโมเดลทำนายได้ถูกต้องว่าเป็น none

ค่าที่ขอบของเมตริกซ์เป็นค่าผลรวมในแนวนอนและแนวตั้ง มีความหมายดังนี้

actual_EI หมายถึง จำนวนข้อมูลทดสอบทั้งหมดที่มีคลาสที่แท้จริงเป็น exon/intron
คำนวณได้จากผลรวมของค่าในแนวนอนได้แก่ ($a + b + c$)

actual_IE หมายถึง จำนวนข้อมูลทดสอบทั้งหมดที่มีคลาสที่แท้จริงเป็น intron/exon
คำนวณได้จากผลรวมของค่าในแนวนอนได้แก่ ($d + e + f$)

actual_None หมายถึง จำนวนข้อมูลทดสอบทั้งหมดที่มีคลาสที่แท้จริงเป็น none คำนวณได้จาก
ผลรวมของค่าในแนวนอนได้แก่ ($g + h + i$)

<i>predict_EI</i>	หมายถึง จำนวนข้อมูลทดสอบทั้งหมดที่โมเดลทำนายว่าเป็นคลาส exon/intron คำนวณได้จากผลรวมของค่าในแนวตั้งได้แก่ $(a + d + g)$
<i>predict_IE</i>	หมายถึง จำนวนข้อมูลทดสอบทั้งหมดที่โมเดลทำนายว่าเป็นคลาส intron/exon คำนวณได้จากผลรวมของค่าในแนวตั้งได้แก่ $(b + e + h)$
<i>predict_None</i>	หมายถึง จำนวนข้อมูลทดสอบทั้งหมดที่โมเดลทำนายว่าเป็นคลาส none คำนวณได้จากผลรวมของค่าในแนวตั้งได้แก่ $(c + f + i)$

การวัดความแม่นยำในการทำนายโดยรวมของโมเดล (overall accuracy) จะใช้ผลรวมของการทำนายถูกต้องในช่อง a, e, i หารด้วยจำนวนข้อมูลทดสอบทั้งหมด แต่ในกรณีที่ข้อมูลในแต่ละคลาสมีจำนวนไม่เท่ากันจะใช้การวิเคราะห์อย่างละเอียดเป็นรายคลาส มาตรการที่นิยมใช้ในการวัดประสิทธิภาพโมเดลอย่างละเอียดเป็นรายคลาส ประกอบด้วยมาตรวัด true positive rate มาตรการวัด false positive rate มาตรการวัด precision และมาตรการวัด F-measure การคำนวณในแต่ละมาตรวัดอธิบายได้ดังต่อไปนี้

มาตรวัดอัตราทำนายคลาสเป้าหมายถูกต้อง (true positive rate, or TP rate) หรือค่าระลึก (recall) หรือค่าความไว (sensitivity) โดยที่คลาสเป้าหมายของการวิเคราะห์นิยมเรียกว่า คลาสบวก (positive class) มาตรการนี้ใช้เพื่อวัดว่าข้อมูลที่เป็นคลาสบวก โมเดลสามารถทำนายได้ถูกต้องว่าเป็นคลาสบวก มีจำนวนมากหรือน้อยเพียงใด การคำนวณจะเป็นดังสมการที่ 3.1 - 3.3 และค่าอัตราทำนายคลาสเป้าหมายถูกต้องจะมีค่าอยู่ระหว่าง 0 ถึง 1 ค่าที่เข้าใกล้หนึ่งถือเป็นค่าที่ดี แสดงถึงความไวของโมเดลที่สามารถตรวจจับข้อมูลในคลาสเป้าหมายได้ครบถ้วน

$$\text{TP rate (for class exon/intron)} = \frac{a}{(\text{actual_EI})} \quad (3.1)$$

$$\text{TP rate (for class intron/exon)} = \frac{e}{(\text{actual_IE})} \quad (3.2)$$

$$\text{TP rate (for class none)} = \frac{i}{(\text{actual_None})} \quad (3.3)$$

มาตรวัดอัตราทำนายคลาสอื่นผิดว่าเป็นคลาสเป้าหมาย (false positive rate, or FP rate) มาตรการนี้ใช้วัดว่าข้อมูลที่มีใช่คลาสเป้าหมายแต่โมเดลทำนายผิดว่าเป็นคลาสเป้าหมาย มีจำนวนมากหรือน้อยเพียงใด การคำนวณจะเป็นดังสมการที่ 3.4 - 3.6 ค่าอัตราทำนายคลาสอื่นผิดว่าเป็นคลาสเป้าหมาย (บางครั้งเรียกว่าสัญญาณเตือนผิด หรือ false alarm) จะมีค่าอยู่ระหว่าง 0 ถึง 1 ค่าที่เข้าใกล้ศูนย์ถือเป็นค่าที่ดี ค่านี้แสดงถึงความสามารถของโมเดลที่ตรวจจับคลาสเป้าหมายไม่ไวมากเกินไปจนจุดพอดี จนกระทั่งทำนายคลาสอื่นว่าเป็นคลาสเป้าหมาย

$$\text{FP rate (for class exon/intron)} = \frac{(d + g)}{(\text{actual_IE} + \text{actual_None})} \quad (3.4)$$

$$\text{FP rate (for class intron/exon)} = \frac{(b + h)}{(\text{actual_EI} + \text{actual_None})} \quad (3.5)$$

$$\text{FP rate (for class none)} = \frac{(c + f)}{(\text{actual_IE} + \text{actual_EI})} \quad (3.6)$$

มาตรวัดความแม่นยำ (precision) ใช้วัดความสามารถของโมเดลว่าเมื่อโมเดลทำนายข้อมูลว่าเป็นคลาสเป้าหมาย การทำนายนั้นถูกต้องเพียงใด ค่าความแม่นยำอยู่ระหว่าง 0 ถึง 1 ค่าที่เข้าใกล้หนึ่งมากที่สุดถือว่าเป็นค่าที่ดี การคำนวณจะเป็นดังสมการที่ 3.7 – 3.9

$$\text{Precision (for class exon/intron)} = \frac{a}{(\text{predict_EI})} \quad (3.7)$$

$$\text{Precision (for class intron/exon)} = \frac{e}{(\text{predict_IE})} \quad (3.8)$$

$$\text{Precision (for class none)} = \frac{i}{(\text{predict_None})} \quad (3.9)$$

มาตรวัดเอฟ (F-measure) ใช้วัดความสามารถของโมเดลทั้งด้านค่าความไว (หรือค่าระลึก) และค่าความแม่นยำ ทั้งนี้เนื่องจากบางโมเดลอาจมีค่าความไวสูงแต่มีค่าความแม่นยำต่ำ เกิดจากการที่โมเดลทำนายข้อมูลทั้งหมดว่าเป็นคลาสเป้าหมาย จะทำให้ได้ค่าความไวที่สูงสุด (= 1) แต่ค่าความแม่นยำต่ำเนื่องจากข้อมูลในคลาสอื่นทั้งหมดจะถูกทำนายว่าเป็นคลาสเป้าหมาย มาตรวัดเอฟจึงถูกพัฒนาขึ้นเพื่อวัดทั้งค่าความไวและค่าความแม่นยำพร้อมกัน ค่าเอฟอยู่ระหว่าง 0 ถึง 1 ค่าที่เข้าใกล้หนึ่งมากที่สุดถือว่าเป็นค่าที่ดี แสดงถึงประสิทธิภาพของโมเดลว่ามีทั้งค่าความไว (หรือค่าระลึก) และค่าความแม่นยำที่ดี การคำนวณค่าเอฟจะเป็นดังสมการที่ 3.10

$$\text{F-measure (by class)} = \frac{(2 \times \text{Precision} \times \text{Recall})}{(\text{Precision} + \text{Recall})} \quad (3.10)$$

3.3 ผลการทดสอบโปรแกรมการค้นพบรูปแบบที่ปรากฏบ่อยในข้อมูลดีเอ็นเอ

การทดสอบประสิทธิภาพของโมเดลเพื่อการทำนายจุดเชื่อมต่อในสายดีเอ็นเอที่ใช้เทคนิคการค้นพบรูปแบบที่ปรากฏบ่อย โดยโปรแกรม assoDNA จะทดสอบความสามารถของโมเดลเปรียบเทียบกับโมเดลที่ได้จากเทคนิคอื่นอีก 4 เทคนิค ที่นิยมใช้ในงานการเรียนรู้ของเครื่อง ได้แก่

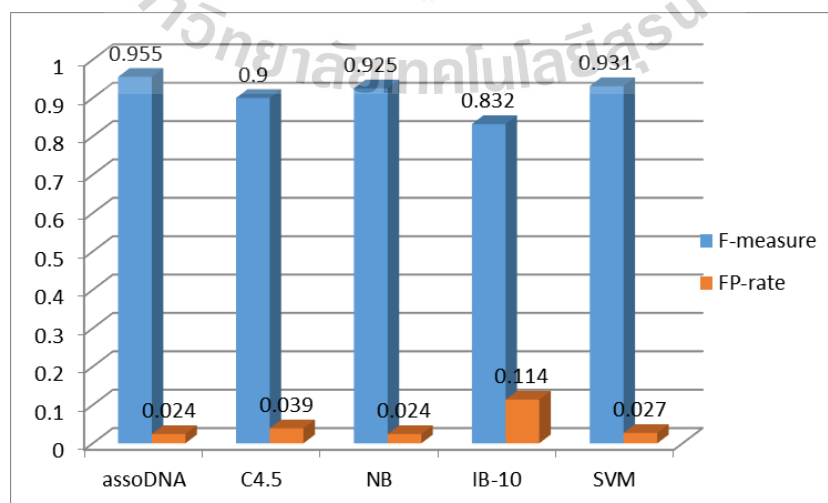
C4.5, Naïve Bayes, Instance based (using 10 nearest neighbors) และ Support vector machine (using polynomial kernel)

การเปรียบเทียบจะใช้มาตรวัด TP rate, FP rate, Precision และ F-measure ผลการเปรียบเทียบประสิทธิภาพการตรวจจับจุดเชื่อมต่อ exon/intron แสดงดังตารางที่ 3.3 และแสดงภาพกราฟ (รูปที่ 3.3) เปรียบเทียบเฉพาะมาตรวัด F-measure และ FP rate เนื่องจากเป็นวัตถุประสงค์หลักของงานวิจัยนี้ที่ต้องการโมเดลที่มีประสิทธิภาพการทำนายที่ดี ซึ่งมีทั้งความไวและความแม่นยำในการทำนาย และขณะเดียวกันจะต้องมีค่า false positive ที่ต่ำ

ผลการเปรียบเทียบประสิทธิภาพการตรวจจับจุดเชื่อมต่อ intron/exon แสดงดังตารางที่ 3.4 และเปรียบเทียบ F-measure และ FP rate ด้วยกราฟดังรูปที่ 3.4 ผลการเปรียบเทียบประสิทธิภาพการตรวจจับสายดีเอ็นเอกรณีที่ไม่ปรากฏจุดเชื่อมต่อ แสดงดังตารางที่ 3.5 และเปรียบเทียบ F-measure และ FP rate ด้วยกราฟดังรูปที่ 3.5

ตารางที่ 3.3 การเปรียบเทียบเทคนิคการทำนายจุดเชื่อมต่อ exon/intron

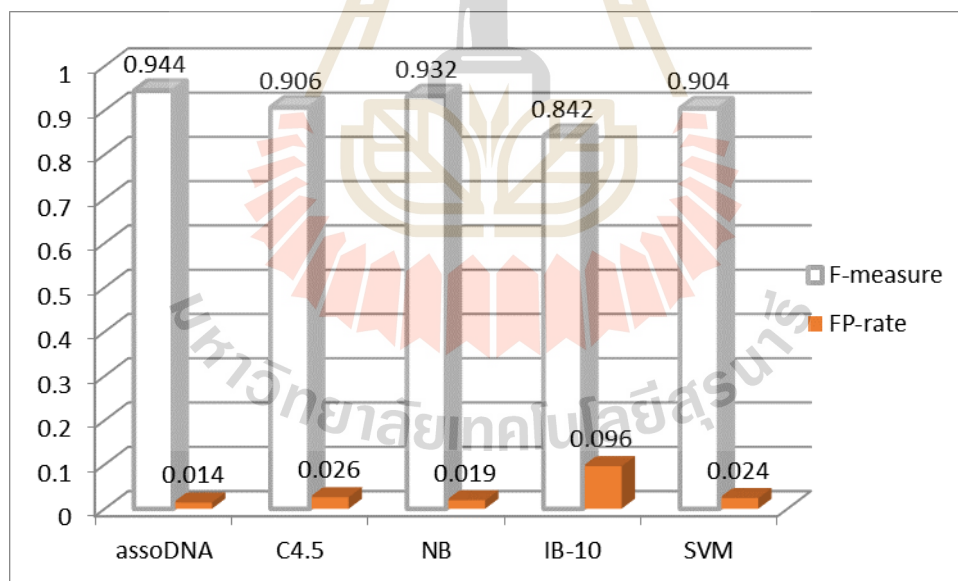
Method	Model performance metrics			
	TP rate (recall)	FP rate	Precision	F-measure
assoDNA	0.977	0.024	0.934	0.955
C4.5	0.911	0.039	0.890	0.900
Naïve Bayes	0.921	0.024	0.930	0.925
Instance based (10 neighbors)	0.950	0.114	0.740	0.832
Support vector machine (polynomial kernel)	0.941	0.027	0.922	0.931



รูปที่ 3.3 การเปรียบเทียบค่า F-measure และ FP rate ของโมเดลตรวจจับ exon/intron

ตารางที่ 3.4 การเปรียบเทียบเทคนิคการทำนายจุดเชื่อมต่อก่อน intron/exon

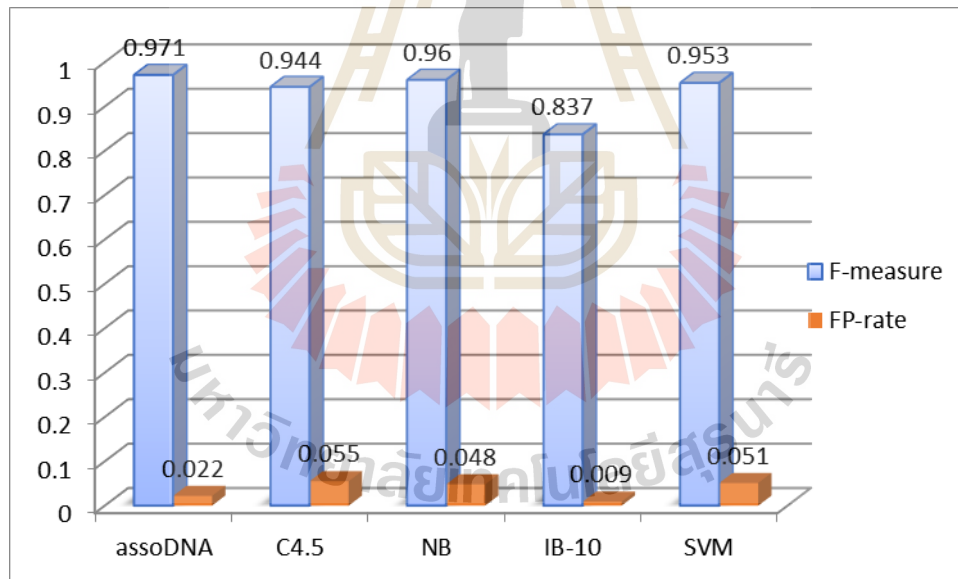
Method	Model performance metrics			
	TP rate (recall)	FP rate	Precision	F-measure
assoDNA	0.936	0.014	0.953	0.944
C4.5	0.900	0.026	0.913	0.906
Naïve Bayes	0.925	0.019	0.938	0.932
Instance based (10 neighbors)	0.954	0.096	0.754	0.842
Support vector machine (polynomial kernel)	0.889	0.024	0.919	0.904



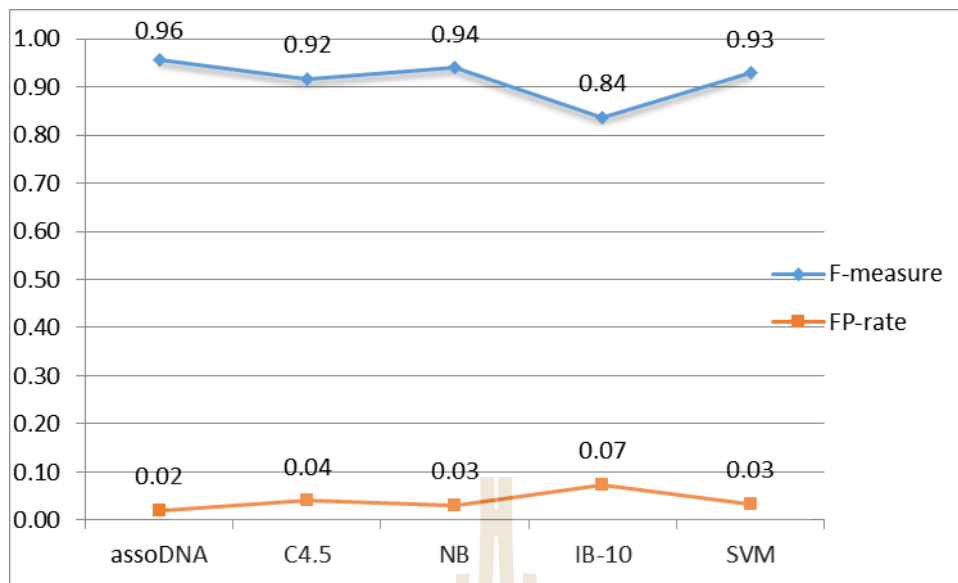
รูปที่ 3.4 การเปรียบเทียบค่า F-measure และ FP rate ของโมเดลตรวจจับ intron/exon

ตารางที่ 3.5 การเปรียบเทียบเทคนิคการทำนายดีเอ็นเอกรณีไม่ปรากฏจุดเชื่อมต่อ

Method	Model performance metrics			
	TP rate (recall)	FP rate	Precision	F-measure
assoDNA	0.964	0.022	0.978	0.971
C4.5	0.942	0.055	0.947	0.944
Naïve Bayes	0.965	0.048	0.954	0.960
Instance based (10 neighbors)	0.726	0.009	0.989	0.837
Support vector machine (polynomial kernel)	0.955	0.051	0.950	0.953



รูปที่ 3.5 การเปรียบเทียบค่า F-measure และ FP rate ของโมเดลกรณีไม่ปรากฏจุดเชื่อมต่อ



รูปที่ 3.6 การเปรียบเทียบค่า F-measure และ FP rate ของโมเดลโดยเฉลี่ยของทุกคลาส

ผลการทดสอบประสิทธิภาพของโมเดลที่ได้จากโปรแกรม assoDNA สำหรับทำนายจุดเชื่อมต่อ exon/intron จุดเชื่อมต่อ intron/exon และกรณีที่ไม่ปรากฏจุดเชื่อมต่อในสายดีเอ็นเอ ปรากฏว่าให้ค่าความไว (TP rate) ค่าความแม่นยำ (Precision) และค่าเอฟ (F-measure) ที่สูงกว่าโมเดลที่ได้จากเทคนิคอื่น ในขณะที่ค่าความผิดพลาดประเภท false positive rate มีค่าที่ต่ำกว่าเทคนิคอื่น ๆ

เมื่อเปรียบเทียบค่า F-measure และค่า FP rate โดยเฉลี่ยจากข้อมูลทั้งสามคลาส (รูปที่ 3.6) ผลปรากฏว่าเทคนิคของโปรแกรม assoDNA ให้โมเดลที่มีประสิทธิภาพดีกว่าโมเดลอื่น โมเดลที่ตีรองลงมาคือ Naïve Bayes และ Support vector machine

3.4 ผลการทดสอบวิธีการค้นพบโดยประมาณ

ในกรณีที่ข้อมูลดีเอ็นเอมีขนาดใหญ่ โครงการวิจัยนี้นำเสนอเทคนิคเพิ่มเติมคือใช้การสุ่มตามความหนาแน่นของข้อมูล เพื่อลดขนาดของข้อมูลให้สามารถประมวลผลได้ในเครื่องคอมพิวเตอร์ที่มีหน่วยความจำจำกัด การทดสอบโมเดลที่ได้จากวิธีการสุ่มซึ่งเป็นการค้นพบโดยประมาณนี้ ใช้วิธีการทดสอบเปรียบเทียบกับโมเดลที่ได้จากการค้นพบรูปแบบจากข้อมูลทั้งหมดโดยไม่มี การสุ่มเลือกเกณฑ์ในการวัดผลใช้การนับจำนวนรูปแบบที่ค้นพบจากวิธี assoDNA ปกติ เปรียบเทียบกับรูปแบบที่ได้จากการประมวลผลข้อมูลที่ผ่านมาขั้นตอนการสุ่ม ผลการทดลองแสดงดังตารางที่ 3.6 และผลการเปรียบเทียบรูปแบบที่ตรงกันของวิธีปกติและวิธีโดยประมาณสรุปได้ดังตารางที่ 3.7 จากการวิเคราะห์ผลการเปรียบเทียบรูปแบบ สรุปได้ว่าวิธีปกติและวิธีโดยประมาณให้ผลลัพธ์ที่ใกล้เคียงกัน

ตารางที่ 3.6 การเปรียบเทียบจำนวนรูปแบบที่ค้นพบโดยวิธีปกติและโดยวิธีประมาณ

Minimum support	Traditional pattern discovery method				Approximate method				#Matched patterns
	# 1-item	# 2-item	# 3-item	# 4-item	# 1-item	# 2-item	# 3-item	# 4-item	
Class = "none"									
50%	0	0	0	0	0	0	0	0	0
45%	0	0	0	0	0	0	0	0	0
40%	0	0	0	0	0	0	0	0	0
35%	0	0	0	0	0	0	0	0	0
30%	1	0	0	0	1	0	0	0	1
25%	117	0	0	0	111	0	0	0	111
Class = "exon/intron"									
85%	3	2	0	0	3	2	0	0	5
80%	4	5	2	0	4	5	2	0	11
75%	4	5	2	0	4	5	2	0	11
70%	4	6	3	0	4	6	3	0	13
65%	5	8	5	1	5	8	5	1	19
60%	5	9	7	2	5	8	5	1	19
Class = "intron/exon"									
85%	2	1	0	0	2	1	0	0	3
80%	3	3	1	0	3	3	1	0	7
75%	3	3	1	0	3	3	1	0	7
70%	3	3	1	0	3	3	1	0	7
65%	3	3	1	0	3	3	1	0	7
60%	3	3	1	0	3	3	1	0	7

ตารางที่ 3.7 ผลการวิเคราะห์จำนวนรูปแบบที่ตรงกันระหว่างการค้นพบแบบปกติและแบบประมาณ

Class	Matched patterns (traditional method)		Matched patterns (approximate method)		Difference (traditional vs approximate)	
	2-item patterns	3-item patterns	2-item patterns	3-item patterns	2-item patterns	3-item patterns
none	--	--	--	--	--	--
exon/intron	91.24%	84.35%	90.98%	83.27%	0.26	1.08
intron/exon	90.11%	87.62%	90.09%	86.89%	0.02	0.73

3.5 ผลการทดสอบวิธีการค้นพบรูปแบบที่ปรากฏบ่อยแบบขนาน

การใช้วิธีการโปรแกรมแบบขนาน (ด้วยโปรแกรม Parallel-assoDNA) เมื่อเปรียบเทียบกับวิธีการโปรแกรมแบบลำดับ (ด้วยโปรแกรม assoDNA) พบว่าการโปรแกรมแบบขนานให้โมเดลผลลัพธ์ตรงกันกับวิธีการโปรแกรมแบบลำดับ แต่ใช้เวลาในการประมวลผลต่ำกว่าประมาณ 46.29% แสดงผลการทดสอบได้ดังรูปที่ 3.6 บรรทัดสุดท้ายของจอภาพด้านบนแสดงเวลาในการประมวลผลแบบลำดับ (หน่วยของเวลาเป็นไมโครวินาที) และจอภาพด้านล่างแสดงเวลาในการประมวลผลแบบขนาน

```

Erlang
File Edit Options View Help
2> timer:tc(assoDNA_par,run,[self(),[1,1,80,1,2,80,1,3,80]]).
=====Read from file:"spliceDNA.DATA"=====
There are 1-3 ClassesClass ="none"
-----START---Apriori(in class=1,Min Support80%=800.0)---
[]
=====Read from file:"spliceDNA.DATA"=====
There are 1-3 ClassesClass ="exon/intron"
-----START---Apriori(in class=2,Min Support80%=400.0)---
K=2-[[{"G(1)","G(5)",427.85.39999999999999},
      [{"G(1)","T(2)",494.98.8},
       [{"G(5)","T(2)",424.84.8}], has 3 set
K=3-[[{"G(1)","G(5)","T(2)",424.84.8}], has 1 set
[ [{"TQ",494},
  [{"GP",499},
  [{"GT",427},
  [{"GP","GT",427},
  [{"GP","TQ",494},
  [{"GT","TQ",424},
  [{"GP","GT","TQ",424}]]]
=====Read from file:"spliceDNA.DATA"=====
There are 1-3 ClassesClass ="intron/exon"
-----START---Apriori(in class=3,Min Support80%=400.0)---
K=2-[[{"A(-2)","G(-1)",496.99.2}], has 1 set
[ [{"AM",497}, {"GN",498}, [{"AM","GN",496}]]
[9875000,not_end]

Erlang
File Edit Options View Help
3> f(), (T,_)=timer:tc(assoDNA_par,concurrent,[1,1,80],[1,2,80],[1,3,80]).
=====Read from file:"spliceDNA.DATA"=====
=====Read from file:"spliceDNA.DATA"=====
There are 1-3 ClassesClass ="exon/intron"
-----START---Apriori(in class=2,Min Support80%=400.0)---
There are 1-3 Classes
There are 1-3 ClassesClass ="none" Class ="intron/exon"
-----START---Apriori(in class=3,Min Support80%=400.0)---
-----START---Apriori(in class=1,Min Support80%=800.0)---
K=2-[[{"G(1)","G(5)",427.85.39999999999999},
      [{"G(1)","T(2)",494.98.8},
       [{"G(5)","T(2)",424.84.8}], has 3 set
K=3-[[{"G(1)","G(5)","T(2)",424.84.8}], has 1 set
[ [{"TQ",494},
  [{"GP",499},
  [{"GT",427},
  [{"GP","GT",427},
  [{"GP","TQ",494},
  [{"GT","TQ",424},
  [{"GP","GT","TQ",424}]]]
K=2-[[{"A(-2)","G(-1)",496.99.2}], has 1 set
[ [{"AM",497}, {"GN",498}, [{"AM","GN",496}]]
[5304000,ok]

```

การโปรแกรมแบบลำดับ (9,875,000 μ sec.)

การโปรแกรมแบบขนาน (5,304,000 μ sec.)

รูปที่ 3.7 เปรียบเทียบเวลาในการประมวลผลโปรแกรมแบบลำดับและแบบขนาน

บทที่ 4

บทสรุป

4.1 สรุปผลการวิจัย

วิธีการทางวิศวกรรมความรู้เพื่อการรู้จำบริเวณกำหนดรหัสทางพันธุกรรม ที่นำเสนอในโครงการวิจัยนี้เป็นการประยุกต์ใช้เทคนิคเหมืองข้อมูลประเภทการค้นพบความสัมพันธ์ (association mining) ช่วยในการค้นหารูปแบบที่มักจะปรากฏในสายดีเอ็นเอด้วยวิธีการเรียนรู้อัตโนมัติจากชุดข้อมูลฝึก รูปแบบในสายดีเอ็นเอของโครงการวิจัยนี้เน้นที่รูปแบบของจุดเชื่อมต่อเอ็กซอน/อินทรอนและอินทรอน/เอ็กซอน รูปแบบทั้งสองประเภทนี้รวมเรียกว่า รูปแบบจุดเชื่อมต่อ (splice-site patterns) ซึ่งการปรากฏจุดเชื่อมต่อจะเป็นเครื่องหมายบ่งชี้ว่าดีเอ็นเอบริเวณที่กำลังศึกษาเป็นส่วนทำหน้าที่ถ่ายถอดรหัสคำสั่งเพื่อควบคุมการสร้างโปรตีน เนื่องจากในขั้นตอนของการสังเคราะห์โปรตีนในเซลล์ ส่วนย่อยภายในสายดีเอ็นเอที่เรียกว่าอินทรอนจะถูกตัดทิ้งและส่วนที่เหลือคือเอ็กซอนจะถูกดึงมาเชื่อมต่อกัน นิวคลีโอไทด์ทั้งหมดในสายเอ็กซอนนี้จึงทำหน้าที่เป็นส่วนบรรจุรหัสคำสั่งสำหรับควบคุมการสร้างกรดอะมิโนแต่ละชนิด สายของกรดอะมิโนจะถูกพัฒนาโครงสร้างต่อไปด้วยพันธะเพปไทด์เกิดเป็นโมเลกุลของโปรตีนขนาดยาว ชนิดของโปรตีนที่สร้างจะแตกต่างกันไปตามรหัสคำสั่งที่บรรจุไว้ในสายของดีเอ็นเอ การที่สามารถระบุได้ว่าส่วนของดีเอ็นเอที่กำลังศึกษาอยู่นั้นมีรหัสคำสั่งสำหรับควบคุมการสร้างโปรตีนปรากฏอยู่หรือไม่ เป็นจุดเริ่มต้นของการวิเคราะห์ที่ซับซ้อนขึ้นในลำดับต่อไป เช่น โปรตีนที่จะถูกสร้างขึ้นจะเป็นโปรตีนชนิดใดและมีความบกพร่องในกระบวนการสร้างซ่อนเร้นอยู่ในรหัสคำสั่งหรือไม่

การพยายามรู้จำรูปแบบของจุดเชื่อมต่อเอ็กซอน/อินทรอนและอินทรอน/เอ็กซอนบนสายดีเอ็นเอด้วยวิธีการอัตโนมัติของการทำเหมืองข้อมูล จึงเป็นการศึกษาเกี่ยวกับรหัสทางพันธุกรรมโดยใช้คอมพิวเตอร์ช่วยในกระบวนการวิเคราะห์ โดยทั่วไปวิธีการทำเหมืองข้อมูลที่น่ามาใช้มักจะเป็นการเรียนรู้เพื่อระบุประเภทหรือจำแนกคลาสของข้อมูล (classification) จัดเป็นการเรียนรู้แบบมีผู้สอน (supervised learning) โดยผู้สอนจะหมายถึงชุดข้อมูลฝึกที่ข้อมูลแต่ละรายการมีคลาสของข้อมูลกำกับไว้ คลาสของข้อมูลที่กำกับไว้เป็นรายการคอร์ดนี้ทำหน้าที่ฝึกการเรียนรู้ของเครื่องคอมพิวเตอร์ให้สามารถสังเคราะห์รูปแบบที่เป็นลักษณะเด่นประจำแต่ละคลาสข้อมูล

ในโครงการวิจัยนี้ใช้เทคนิคที่แตกต่างจากงานวิจัยอื่น ๆ คือใช้วิธีการค้นพบความสัมพันธ์ซึ่งโดยปกติจัดเป็นการเรียนรู้แบบไม่มีผู้สอน (unsupervised learning) เนื่องจากชุดข้อมูลที่ใช้เป็นข้อมูลฝึกไม่จำเป็นต้องมีการระบุคลาส เพราะการเรียนรู้รูปแบบไม่ได้มีวัตถุประสงค์กำหนดรูปแบบที่เป็นลักษณะเฉพาะของคลาส แต่เป็นการเรียนรู้รูปแบบใด ๆ ที่ปรากฏขึ้นร่วมกันในข้อมูลฝึกชุดนั้น โดยข้อมูลที่ต่างคลาสนั้นอาจแสดงรูปแบบที่เหมือนกันได้ รูปแบบที่แสดงออกเหมือนกันจะถูกนำมา

วิเคราะห์ร่วมกัน และรูปแบบที่ปรากฏร่วมกันบ่อยที่สุดจะถูกคัดเลือกไว้เป็นผลลัพธ์สุดท้ายของกระบวนการค้นพบความสัมพันธ์ ซึ่งหมายถึงความสัมพันธ์ที่ปรากฏบ่อยในชุดข้อมูลฝึก

ในงานวิจัยนี้เลือกใช้วิธีการทำเหมืองข้อมูลประเภทการค้นพบความสัมพันธ์ แต่ปรับปรุงขั้นตอนก่อนหน้าการค้นหารูปแบบที่ปรากฏบ่อยด้วยการจัดข้อมูลเป็นกลุ่มย่อยตามคลาสที่สนใจ ซึ่งในงานวิจัยนี้ประกอบด้วย ๓ คลาสคือ คลาสของสายดีเอ็นเอที่มีจุดเชื่อมต่อเอ็กซอน/อินทรอน คลาสของสายดีเอ็นเอที่มีจุดเชื่อมต่ออินทรอน/เอ็กซอน และคลาสของสายดีเอ็นเอที่ไม่ปรากฏจุดเชื่อมต่อทั้งสองแบบ การค้นพบความสัมพันธ์กระทำกับข้อมูลในกลุ่มย่อยแต่ละกลุ่ม การเรียนรู้แบบนี้จึงจัดเป็นแบบกึ่งสอน (semi-supervised learning) เนื่องจากมีการชี้แนะทางอ้อมด้วยการแยกข้อมูลที่เป็นคลาสเดียวกันให้อยู่ในชุดข้อมูลฝึกชุดเดียวกัน จากนั้นจึงค้นหาลักษณะที่ปรากฏร่วมกันบ่อย ๆ ในชุดข้อมูลนั้น ลักษณะการเรียนรู้แบบนี้ต่างจากแบบการเรียนรู้แบบมีผู้สอนประเภทการจำแนกคลาสข้อมูล ตรงที่ข้อมูลแต่ละคลาสถูกแยกออกเป็นแต่ละกลุ่มย่อยและการเรียนรู้รูปแบบที่ปรากฏบ่อยกระทำในกลุ่มย่อย การเรียนรู้รูปแบบในแต่ละกลุ่มย่อยเป็นไปโดยอิสระทำให้ลักษณะเด่นของกลุ่มไม่ขึ้นกับข้อมูลในกลุ่มอื่น ขนาดของกลุ่มข้อมูลที่ไม่เท่ากันจึงไม่มีผลกระทบต่อกระบวนการเรียนรู้

ลักษณะเด่นของกระบวนการเรียนรู้แบบกึ่งสอนด้วยอัลกอริทึมการค้นพบความสัมพันธ์ จึงเป็นการค้นหารูปแบบที่แต่ละคลาสข้อมูลสามารถแสดงรูปแบบเด่นออกมาได้แม้จะมีข้อมูลในคลาสน้อยกว่าข้อมูลในคลาสนอื่น ๆ ผลการค้นพบรูปแบบจุดเชื่อมต่อในสายดีเอ็นเอได้ยืนยันลักษณะเด่นนี้ โดยชุดข้อมูลที่ใช้ในการทดลองมีข้อมูลในคลาสที่ไม่ปรากฏจุดเชื่อมต่อ มากเป็นสองเท่าของข้อมูลในคลาสเอ็กซอน/อินทรอน และมากเป็นสองเท่าของข้อมูลในคลาสอินทรอน/เอ็กซอน กรณีข้อมูลไม่สมดุลเช่นนี้รูปแบบเพื่อการจำแนกคลาสโดยทั่วไป จะเอนเอียงไปทางทำนายข้อมูลในคลาสส่วนใหญ่ได้ถูกต้องมากกว่าทำนายข้อมูลที่เป็นคลาสส่วนน้อย แต่การทำนายด้วยวิธีการเรียนรู้แบบกึ่งสอนด้วยอัลกอริทึมการค้นพบความสัมพันธ์ที่นำเสนอในโครงการวิจัยนี้ ให้ผลการทำนายที่มีความแม่นยำสูงทั้งในกรณีคลาสส่วนใหญ่และคลาสส่วนน้อย ดังแสดงด้วยผลการทดลองในบทที่ 3

นอกจากนี้ด้วยลักษณะการเรียนรู้แบบกึ่งสอนด้วยอัลกอริทึมการค้นพบความสัมพันธ์ ที่นำเสนอในงานวิจัยนี้มีการแยกข้อมูลเป็นกลุ่มย่อยตามคลาสของข้อมูล ทำให้เอื้อต่อการปรับปรุงกระบวนการให้สามารถประมวลผลแบบขนานได้ ผลการทดลองเพื่อเปรียบเทียบความเร็วและความถูกต้องในการประมวลผลของโปรแกรม assoDNA และโปรแกรม Parallel-assoDNA ยืนยันว่าการเรียนรู้แบบกึ่งสอนด้วยอัลกอริทึมการค้นพบความสัมพันธ์ที่นำเสนอในงานวิจัยนี้ สามารถพัฒนาประสิทธิภาพให้สูงขึ้นด้วยวิธีการประมวลผลแบบขนานได้ โดยที่ยังให้ผลการค้นพบรูปแบบที่ถูกต้องตรงกันกับวิธีการประมวลผลแบบลำดับ

ในกรณีของการประมวลผลกับข้อมูลดีเอ็นเอขนาดใหญ่ หรือมีข้อมูลที่เกิดขึ้นต่อเนื่องในลักษณะสตรีม วิธีการแบบประมาณที่ใช้แนวทางการสุ่มข้อมูลตามความหนาแน่นเพื่อนำมาประมวลผล

ด้วยโปรแกรม assoDNA ให้ผลลัพธ์ที่มีความถูกต้องใกล้เคียงกับวิธีการประมวลผลกับชุดข้อมูลทั้งหมด ดังแสดงในผลการทดลองของบทที่ 3

4.2 ข้อจำกัดของโปรแกรมและข้อเสนอแนะ

โปรแกรมเพื่อการรู้จำบริเวณกำหนดรหัสทางพันธุกรรม หรือ โปรแกรม assoDNA ยังมีข้อจำกัดตรงที่ขั้นตอนย่อยบางส่วนไม่ใช่ขั้นตอนอัตโนมัติทั้งหมด เช่น ขั้นตอนการแยกข้อมูลเป็นกลุ่มย่อยตามคลาสของข้อมูล ขั้นตอนการสุ่มตามความหนาแน่นเมื่อต้องการเรียนรู้รูปแบบแบบประมาณ ในอนาคตขั้นตอนเหล่านี้สามารถพัฒนาให้เป็นกระบวนการอัตโนมัติทั้งหมดได้

แนวทางการพัฒนาในอนาคตที่สำคัญอีกแนวทางหนึ่งคือ การปรับปรุงวิธีการประมวลผลของโปรแกรม assoDNA ให้สามารถรองรับข้อมูลขนาดใหญ่มากในลักษณะของ Big data ได้ด้วยวิธีการแบ่งข้อมูลเป็นส่วนย่อยและประมวลผลข้อมูลส่วนย่อยเหล่านั้นเพื่อเรียนรู้ลักษณะเด่นของกลุ่มข้อมูลในแบบเพิ่มพูน (incremental learning)



บรรณานุกรม

- R. Agrawal, T. Imielinski, and A. Swami (1993), Mining association rules between set of items in large databases, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 207-216.
- M. Alexandersson, S. Cawley, and L. Pachter (2003), SLAM: Cross-species gene finding and alignment with a generalized pair hidden Markov model, *Genome Research*, Vol.13, pp.496-502.
- M. Aniba, O. Poch, A. Marchler-Bauer, and J. Thompson (2010), AlexSys: A knowledge-based expert system for multiple sequence alignment construction and analysis, *Nucleic Acids Research*, June, pp.1-12.
- V. Bafna and D. Huson (2000), The conserved exon method for gene finding, *Proc. 8th Int. Conf. on Intelligent Systems for Molecular Biology*, pp.3-12.
- P. Baldi and S. Brunak (2001), *Bioinformatics: The Machine Learning Approach*, second edition, The MIT Press, Cambridge, Massachusetts.
- S. Bandyopadhyay, U. Maulik, and D. Roy (2008), Gene identification: Classical and computational intelligent approaches, *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, Vol.38, No.1, pp.55-68.
- S. Batzoglou, L. Pachter, J. Mesirov, B. Berger, and E. Lander (2000), Human and mouse gene structure: Comparative analysis and application to exon prediction, *Genome Research*, Vol.10, pp.950-958.
- A. Bernel, K. Crammer, A. Hatzigeorgiou, and F. Pereira (2007), Global discriminative learning for higher-accuracy computational gene prediction, *PLoS Computational Biology*, Vol.3, Issue 3, pp.488-497.
- C. Burge and S. Karlin (1997), Prediction of complete gene structures in human genomic DNA, *Journal of Molecular Biology*, Vol.268, pp.78-94.
- D. Carter and R. Durbin (2006), Vertebrate gene finding from multiple-species alignments using a two-level strategy, *Genome Biology*, Vol.7, Supplement 1, S6.
- J. Cohen (2004), Bioinformatics: An introduction for computer scientists, *ACM Computing Surveys*, Vol.36, No.2, pp.122-158.
- F. Crick (1970), Central dogma of molecular biology, *Nature*, Vol. 227, pp. 561-563.

- J. Do and D. Choi (2006), Computational approach to gene prediction, *The Journal of Microbiology*, Vol.44, No.2, pp.137-144.
- R. Dogan, L. Getoor, and W. Wilbur (2007), Characterizing RNA secondary-structure features and their effects on splice-site prediction, *Proc. 7th IEEE Int. Conf. on Data Mining – Workshop*, pp-89-94.
- J. Fickett (1982), Recognition of protein-coding regions in DNA sequences, *Nucleic Acids Research*, Vol.10, pp.5303-5318.
- P. Flicek (2007), Gene prediction: Compare and CONTRAST, *Genome Biology*, Vol.8, Issue 12, Article 233.
- P. Gouret, V. Vitiello, N. Balandrau, A. Gilles, P. Pontarotti, and E. Danchin (2005), FIGENIX: Intelligent automation of genomic annotation: Expertise integration in a new software platform, *BMC Bioinformatics*, Vol.6, Article 198.
- M. Gribskov, J. Devereux, and R. Burges (1984), The codon preference plot: Graphic analysis of protein coding sequences and prediction of gene expression, *Nucleic Acids Research*, Vol.12, pp.539-549.
- S. Gross and M. Brent (2005), Using multiple alignments to improve gene prediction, *Proc. 9th Annual Int. Conf. on Research in Computational Molecular Biology*, pp.374-388.
- S. Gross, C. Do, M. Sirota, and S. Batzoglou (2007), CONTRAST: A discriminative, phylogeny-free approach to multiple informant de novo gene prediction, *Genome Biology*, Vol.8, Issue 12, Article R269.
- I. Korf, P. Flicek, D. Duan, and M. Brent (2001), Integrating genomic homology into gene structure prediction, *Bioinformatics*, Vol.17, Supplement I, pp. SI40-SI49.
- The Machine Learning Database Repository (2017), <http://mlearn.ics.uci.edu/databases/molecular-biology/splice-junction-gene-sequences>.
- W. Majoros, M. Pertea, and S. Salzberg (2004), TigrScan and Glimmer HMM: Two open source ab initio eukaryotic genefinders, *Bioinformatics*, Vol.20, pp.2878-2879.
- S. Marashi, C. Eslahchi, H. Pezeshk, and M. Sadeghi (2006), Impact of RNA structure on the prediction of donor and acceptor splice sites, *BMC Bioinformatics*, Vol.7, Article 297.

- C. Mathe, M. Sagot, T. Schiex, and P. Rouze (2002), Current methods of gene prediction, their strengths and weaknesses, *Nucleic Acids Research*, Vol.3, No.19, pp.4103-4117.
- C. Ouzounis and A. Valencia (2003), Early bioinformatics: The birth of a discipline – a personal view, *Bioinformatics Journal*, Vol.19, No.17, pp.2176-2190.
- G. Parra, P. Agarwal, J. Abril, T. Wiehe, J. Fickett, and R. Guigo (2003), Comparative gene prediction in human and mouse, *Genome Research*, Vol.13, pp.108-117.
- J. Pedersen and J. Hein (2003), Gene finding with a hidden Markov model of genome structure and evolution, *Bioinformatics*, Vol. 19, pp.219-227.
- V. Rascol, A. Levasseur, O. Chabrol, S. Grusea, P. Gouret, E. Danchin, and P. Pontarotti (2009), CASSIOPE: An expert system for conserved regions searches, *BMC Bioinformatics*, Vol.10, Article 284.
- S. Rogic (2006), The role of pre-mRNA secondary structure in gene splicing in *Saccharomyces cerevisiae*, *PhD Dissertation*, University of British Columbia.
- A. Siepel and D. Haussler (2004), Computational identification of evolutionary conserved exons, *Proc. 8th Annual Int. Conf. on Research in Computational Molecular Biology*, pp.177-186.
- M. Sparks and V. Brendel (2005), Incorporation of splice site probability models for non-canonical introns improves gene structure prediction in plants, *Bioinformatics*, Vol.21, Supplement 3, pp. iii20-iii30.
- R. Staden (1984), Measurements of the effect that coding for a protein has on DNA sequence and their use for finding genes, *Nucleic Acids Research*, Vol.12, pp.551-567.
- M. Stanke and S. Waack (2003), Gene prediction with a hidden Markov model and a new intron submodel, *Bioinformatics*, Vol.19, Supplement 2, pp. ii215-ii228.
- L. Stein (2004), Human genome: End of the beginning, *Nature*, Vol.431, pp.915-916.
- P. Wodehouse (2006), Bioinformatics and pattern recognition come together, *Journal of Pattern Recognition Research*, Vol.1, pp.37-41.

ภาคผนวก



ภาคผนวก ก

ผลผลิตของงานวิจัย: บทความวิจัยตีพิมพ์ในวารสารวิชาการและ Proceedings

1. N. Kerdprasop, K. Kerdprasop (2015). Constraint-based system for genomic analysis. *International Journal of Information and Education Technology*, vol.5, no.2, February, pp.119-123. (indexing: INSPEC, ISSN: 2010-3689)
2. N. Kerdprasop, K. Kerdprasop (2014). Visual knowledge mining and utilization in the inductive expert system. *International Journal of Computers*, vol.8, pp.157-165. (ISSN 1998-4308)
3. N. Kerdprasop, K. Kerdprasop (2014). The discovery of top-k DNA frequent patterns with approximate method. *Malaysian Journal of Computing*, vol.2, no.2, November/December, Article No.3. 12pp. (ISSN 2231-7473)
4. K. Kerdprasop, N. Kerdprasop (2013). Concurrent data mining and genetic computing implemented with Erlang language. *International Journal of Software Engineering and Its Applications*, vol.7, no.3, May, pp.63-75. (indexing: Scopus, ISSN: 1738-9984)
5. N. Kerdprasop, F. Koongaew, K. Kerdprasop (2013). Building and querying a decision tree model with constraint logic programming. *International Journal of Software Engineering and Its Applications*, vol.7, no.5, September, pp.269-282. (indexing: Scopus, ISSN: 1738-9984)
6. N. Kerdprasop, K. Kerdprasop (2013). Knowledge engineering process for a rapid prototyping of inductive expert system. *International Journal of Computer Science Issues (IJCSI)*, vol.10, issue 2, no.3, March, pp.408-414.
7. N. Kerdprasop, K. Kerdprasop (2014). Visual data mining and the creation of inductive knowledge base. *Proceedings of the 5th International Conference on Circuits, Systems, Control, Signals (CSCS'14)*, Salerno, Italy, 3-5 June, pp.181-186.
8. N. Kerdprasop, K. Kerdprasop (2013). Knowledge mining and its application to support computational health informatics. *Proceedings of the International Conference on Education, Language, Society, Science and Engineering in ASEAN and its Neighbors*, Kunming, People's Republic of China, 23-28 February, pp.274-280.

Constraint-Based System for Genomic Analysis

Nittaya Kerdprasop and Kittisak Kerdprasop

Abstract—Recent advent of the new high-throughput biological technologies has brought more challenges to the computer science community in terms of the amount and variety of biological data awaiting for analysis. Computationally intensive techniques such as pattern recognition and machine learning algorithms have been applied to extract knowledge from several biological domains ranging from genomics, proteomics to system biology and evolution process. Learning techniques applied to the computational biology are mostly in the category of classification. Therefore, the sequence analysis problem has to be formulated as classification task, which is quite difficult due to the unobvious one-to-one mapping of the problem. In this paper, we propose a different setting of sequence analysis formulation based on the nucleotide patterns using a constraint logic programming paradigm, in which the sequence alignment can be performed through pattern matching techniques. With available knowledge from the field of pattern mining, we can apply the well-established techniques within the new framework of constraint programming. However, to make the system efficiently work, we need a new set of constraint solver algorithms specifically designed for the sequence analysis problem. The design and implementation of such algorithms are thus the main focus of our research project. We propose in this paper the design of a constraint-based system for genomic sequence analysis including the algorithm for the constraint solver, a major part of the proposed system.

Index Terms—Genomic sequence analysis, constraint-based system, constraint solver algorithm, constraint programming.

I. INTRODUCTION

Living organisms contain multiples cells to perform different functions. There are two basic types of cells: prokaryote cells (found in bacteria) and eukaryote cells (appeared in plants and animals). Contained within the cell membrane are several organelles and thousands of different types of molecules, the important one is DNA (deoxyribonucleic acid) that carries the entire genetic inheritance, or genes, of the cell. DNA is a long polymer molecule that contains sugar, phosphate group, and a mixture of four different nucleotide bases: adenine (A), cytosine (C), guanine (G), and thymine (T).

In 1953, Watson and Crick [1] discovered the DNA double helix structure in a complementary base pairing that A-T and G-C units always occur together, they are thus referred to as base pairs. In 1957, Crick [2], [3] described the flow of

genetic information in biological systems (Fig. 1) that firstly DNA is copied to more DNA in the replication process, then DNA is transcribed into mRNA (or messenger-RNA) in a transcription process and finally mRNA is translated (by ribosome) into protein in a translation process.

This overall process of biological protein synthesis is known as gene expression. Understanding the process of gene expression in different types of cells and under different conditions is one of the fundamental research aspects of genomics, which is all the studies related to genes.

In prokaryotes, genetic information is encoded continuously on a DNA strand. But in eukaryotes, regions that code for protein (called exons) are interrupted by the non-coding regions (called introns). During the transcription of most eukaryotic genes, the primary RNA transcript (or pre-mRNA) needs additional modification step called splicing to remove introns and join exons together to make one long continuous mRNA strand (Fig. 2). The ribosome is an organelle that translates code on mRNA to different kinds of amino acids.

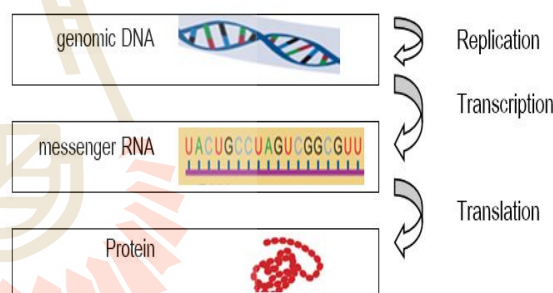


Fig. 1. Flow of genetic information in biological systems.

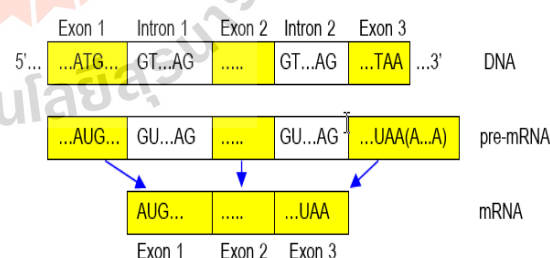


Fig. 2. Removal of introns and joining of exons in the splicing process during the DNA transcription.

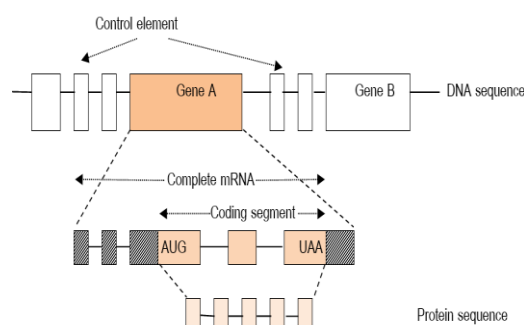


Fig. 3. Structure of gene and genome.

Manuscript received November 28, 2013; revised March 14, 2014. This work was supported in part by the National Research Council of Thailand and Suranaree University of Technology through the funding of Data Engineering Research Unit.

The authors are with the School of Computer Engineering. They are also with Data Engineering Research Unit, Suranaree University of Technology, 111 University Avenue, Muang District, Nakhon Ratchasima 30000, Thailand (e-mail: nittaya@sut.ac.th, kittisakThailand@gmail.com).

In multi-cellular organisms, the DNA in each and every cell is identical. Different cells can perform different functions because different portions of the DNA molecule are active in different cells at different times. The entire DNA sequence contained in a cell, including the genes (stretches of DNA that code for a protein) and the control elements, is referred to as genome. Sketch of genome structure is presented in Fig. 3.

Genome is not only important to the life cycle of the cell, but also represents a blueprint for the life of the organism. Large genome sequencing projects have been set up by many governments and commercial organizations. The development of automated sequencing technologies such as shotgun sequencing technique allows scientists to decode genomes of many organisms at a significantly increasing rate. After a genome is reconstructed from the pieces of sequencing data, the next and most important step is to understand the content of the genome. That is, to identify the gene location and then annotate the function of each gene.

Since the announcement of the draft version of the human genomic sequence in 2001 [4] and the completion of several genome projects during the past decade, enormous amount of raw biological sequence data has been stored and awaited for interpretation. Dealing with large volume of data, efficient computational methods and intelligent techniques are a real need.

Sequence comparison is a fundamental operation in the field of computational molecular biology to detect similarity between biological sequences such as proteins and DNA sequences. The optimal match in a comparison between two sequences can be achieved through the dynamic programming technique. But it is very slow when it has to compare a sequence against many others, or compare among a group of related sequences in large database. To solve the computational time problem, approximate techniques tend to be the method of choice.

Many search tools employ a sophisticated statistical-based technique such as hidden Markov model (HMM). An HMM is a stochastic model that characterizes a coding sequence by computing probability of appearance of a nucleotide base (A, C, G, or T) based on the k previous nucleotides in the sequence. Computer scientists from the machine learning field prefer the neural network and support vector machines approaches rather than the statistical method of HMM. However, to characterize reliable coding sequences, these approaches require a large number of training sequences. The significant impact of such requirement is a very large search space. We, therefore, propose to tackle the sequence analysis problem through the constraint-based setting in which the search space could be reduced prior to the search for solution. Our prototype implementation is based on the constraint logic programming paradigm.

II. LITERATURE REVIEW

Genomic sequences are just raw biological data. To understand biological process inherent in the genomic sequences, computational and statistical techniques such as pattern recognition and machine learning algorithms are essential tools for the analysis of such large amount of

genetic sequences. The analysis task over DNA, RNA and protein sequences includes several subtasks of

- searching for patterns within a sequence,
- searching for similarities between two sequences including sequence alignment for a pairs of sequences,
- searching for similarities among many sequences and performing multiple sequence alignment,
- constructing phylogenetic trees based on sequences,
- predicting and analyzing the secondary structures based on the sequences, and
- predicting and analyzing tertiary structure and folding patterns for protein and RNA sequences.

In this research we concentrate on the first three subtasks of analyses. That is, the problem of finding and parsing eukaryotic protein-coding genes.

Gene finding is generally the detection of sequence elements such as exons, introns, and the intergenic regions that separate genes. Once genes are found, their internal exon-intron structure can be predicted so that the encoded protein may be deduced. The gene finding problem in eukaryotes is difficult because the genes comprise less than 30% of the genome and once a gene is found, the locations of introns within the gene must be precisely determined in order to accurately deduce the protein product of the gene.

In the past, genes were identified with experimental validation on living cells and organisms. It is the most reliable method, but costly and labor intensive. At present, most biologists rely on the computational methods to automatically analyzed the uncharacterized genomic sequences. Some of the frequently applied computational techniques include dynamic programming, linear discriminant analysis, linguistic methods, hidden Markov model, and machine learning techniques such as neural network, decision-tree induction, support vector machines. Several successful gene-finding programs are based on the hidden Markov model algorithms.

A Markov model is a model of discrete stochastic process that evolves through the states from the set $S = \{s_1, s_2, \dots, s_n\}$. The main assumption is that the probability of appearance of any future state depends only on the k preceding states, for some constant k . Given a learning set of sequences, a Markov model can be built by computing the probability that a certain nucleotide x_i appears after a sequence s_i , for example,

$$\begin{aligned} < P(x_i = A \mid s_i = \text{TTGGA}), \\ & P(x_i = C \mid s_i = \text{TTGGA}), \\ & P(x_i = G \mid s_i = \text{TTGGA}), \\ & P(x_i = T \mid s_i = \text{TTGGA}) > \end{aligned}$$

is a computing scheme of the Markov model of degree 5 (that is, $k = 5$). To model the codon usage that appears as a triplet of nucleotide bases, the degree k of the Markov model is normally set to 2, 5, 8, and so on.

The oldest gene-finding method based on Markov model is GeneMark [5]. From the success of GeneMark as an accurate tool to recognize and annotate genes in genome projects, a family of GeneMark programs have been developed including GeneMark.hmm [6], GeneMarkS [7], GeneMarkE [8], GenScan [9], EuGene [10], and GeneTack [11]. The

GeneMark family detects genes by identifying open reading frames (the regions between start and stop codons) using precomputed species-specific gene models as training data to determine parameters of the protein-coding and non-coding regions.

The major limitation of GeneMark family is the fixed-order Markov model such that models of higher order require exponentially more training data, which are usually not available for new sequences. The Glimmer gene-finding program [12] introduces a generalized hidden Markov model with variable order called the interpolated Markov model. Other gene-finding programs that based on the concept of interpolated Markov model and generalized Markov model include FGENESH [13], HMMGene [14], and AUGUSTUS [15].

Machine learning and data mining methods have been successfully applied to various kinds of prediction problems such as exon prediction [16], start codon prediction [17], and splice site prediction [18], [19]. More than 90% of nucleotides can be correctly identified as either coding, or non-coding. But the exact boundaries of the exons and their assemblies into complete coding regions are much more difficult to predict correctly using the classification-oriented formulation. DNA sequences are rather parsing-oriented in their nature.

We thus propose a novel setting of constraint logic programming to formulate a computational method toward the problem of gene searching and recognition in DNA sequences. This new scheme of DNA sequence analysis has just recently gained interest with some preliminary work appeared in the literature [20]-[22].

III. METHODOLOGY

A. Constraint Programming for Computational Genome Analysis

Constraint programming is a programming paradigm normally applied to solve combinatorial search problems such as flight scheduling, crew rostering, logistic planning, and many more of this kind. The main steps of constraint programming are:

- 1) Users specify a problem by defining the variables together with their associated domains and constraints on these variables,
- 2) The search procedure and constraint solver find solutions, which are values assigned to the specified variables such that all constraints are satisfied.

It is obvious from the program structure that constraint programming has been designed to solve constraint satisfaction problems that have been extensively studied in artificial intelligence. The efficiency of constraint programs is basically due to the constraint propagator feature in a constraint solver. The function of constraint propagator is to reduce the domains of variables by inferring from the existing constraints and then to prevent the search procedure from visiting parts of the search tree that do not contain any solution.

A constraint propagator takes as input a domain D from which a variable can be assigned its value, and a set of

constraints C . The output of the propagator is a reduced domain D' . For instance, given that X, Y, Z are variables, the domains

$$\begin{aligned} D(X) &= \{a, c, d\}, \\ D(Y) &= \{a, b, c, d\}, \\ D(Z) &= \{c\}, \end{aligned}$$

and a set of constraints

$$C = \{ X=Y \wedge Y \neq Z \},$$

the output of the constraint propagator are

$$\begin{aligned} D'(X) &= D'(Y) = \{a, d\}, \text{ and} \\ D'(Z) &= \{c\}. \end{aligned}$$

The repeated application of propagator can lead to increasingly stronger (that is, smaller) domains. The propagator continues until it reaches a fixed point in which the domains cannot be further reduced. At this stage, the search procedure (either global or heuristic-based) can efficiently start assigning possible value to each variable. A toy example of map coloring in Fig. 4 illustrates the constrain-and-search strategy of constraint logic programming (CLP), as opposed to the generate-and-test of logic programming (LP) scheme.

```

%% CLP style: Constrain-and-search
:- lib(fd).

map_color_CLP([A,B,C,D]) :-
    % declare variables and domains
    [A,B,C,D]:: red,green,blue,yellow],
    % constrain
    alldifferent([A,B,C,D]),
    % then search
    labeling([A,B,C,D]).

%% LP style: Generate-and-test
color(red).    color(green).
color(blue).   color(yellow).

map_color_LP([A,B,C,D]) :-
    % generate solution
    color(A), color(B),
    color(C), color(D),
    % then test for constraints
    A \= B,    A \= C,    A \= D,
    B \= C,    B \= D,    C \= D.
    
```

Fig. 4. Constraint logic programming versus logic programming schemes.

At present, there are several constraint systems that provide functions to specify (or model) the problems and maintain the constraint consistency efficiently. They are called constraint programming systems if they are based on procedural languages. The systems are classified as constraint logic programming systems if they are based on logic programming languages. The focus of this research is the development of constraint solvers (that is, the integration of constraint propagators and search procedures) for a specific application of genomic analysis using the constraint logic programming paradigm. The main benefits of such scheme are two folds:

- 1) the declarative style allows users to specify a problem itself, instead of specifying how to solve the problem, and

2) a high level of knowledge representation facilitates genomic pattern specification and the inclusion of new knowledge which is highly dynamic in the active area of genomics and computational microbiology.

Most constraint logic programming systems provide a large set of predefined constraints such as `alldifferent` and powerful search commands such as `labeling` to solve the combinatorial problems. The predefined constraints and exhaustive depth-first search procedure aim at solving a general class of constraint satisfaction problems. We argue that for a specific problem of genomic sequence analysis, a new set of built-in constraints that propagate with the already known biological relations together with alternative approximate search methods can more or less benefit the sequence analysis tool.

B. A Framework of Constraint-Based System

The main purpose of our research is the design and implementation of a computational system for genomic sequence detection and recognition of its structure and function using the declarative paradigm of constraint logic programming. The advantage of such paradigm is its powerful features to handle patterns within the DNA and RNA sequences. In addition, the heuristic search such as branch and bound, simulated annealing can be applied to speed up the computation time. The design is sketched as shown in Fig. 5.

A constraint-based approach to DNA sequence analysis starts from the modeling of sequence search and gene finding problems as constraint satisfaction problems, and also constraint optimization problems if preferences are to be numerically measured or when several solutions are generated. We name this step as “Query reformulator.”

The constraints involved in the sequence analysis problems could be symbolic and numeric constraints over finite domains. The constraints formulated at this step can be either local or global constraints. Local constraints are restrictions over local patterns, whereas the global constraints address restrictions over the whole set of solutions.

patterns under constraints can be checked independently of the other patterns holding in the data. The reduced domain of data values is then passed over to the global constraint propagator. The product of this step is the domain store to collect variable domains that their sizes have been reduced by the constraint propagators. The search procedure designed for gene-finding task can now start its process and report solutions to the user. The skeleton of constraint solver is shown in Fig. 6.

1. Develop the problem’s model, in terms of variables and constraints, as a constraint satisfaction problem (or constraint optimization problem if some costs, distance measures, or other measurable metrics are specified)
2. Initialize for all variable-value pairs as modeled in step 1
3. Repeat until a termination condition is reached (this can be a maximum number of iteration, acceptable score range, or other conditions)
 - 3.1 Call the local constraint propagator to constrain the domain space of each variable and return a possibly smaller set of domains
 - 3.2 Call the global constraint propagator to further constrain the domain space
 - 3.3 Perform a search procedure (including a heuristic-based method)
 - 3.3.1 Select a variable
 - 3.3.2 Select a value from the domain
 - 3.3.3 Instantiate the variable
 - 3.3.4 If the instantiation fails (because constraint is violated), then backtrack to the step 3.3.2 and select another value
 - 3.3.5 If the sets of variables and their values are not empty yet, then repeat the steps 3.3.1-3.3.4
4. Return the solution (which is a set of variables and their values as modeled in step 1) if it exists

Fig. 6. A constraint solver algorithm for the genomic sequence analysis system.

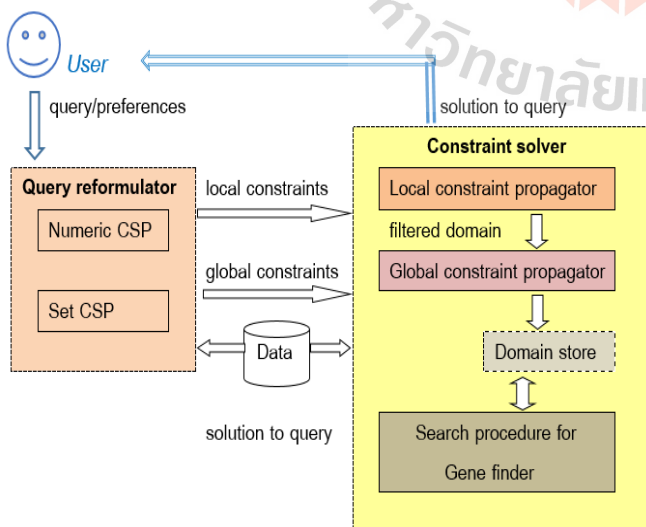


Fig. 5. Schematic overview of a constraint-based system for genomic sequence analysis.

In a subsequent phase of constraint solving, local constraints are handled prior to the global ones because local

IV. CONCLUSION

In the past, genes were identified with experimental validation on living cells and organisms. It is the most reliable method, but costly and labor intensive. At present, most biologists rely on the computational methods to automatically analyze the uncharacterized genomic sequences. Gene finders are programs that analyze and predict the exon-intron structures of genes using the sequences of one or more genomes as their only input. Many algorithms implement statistical and intelligent methods to represent sequence patterns and output a model of the gene structure. Some of the frequently applied techniques include dynamic programming, linear discriminant analysis, linguistic methods, hidden Markov model, and various machine learning techniques such as neural network, decision-tree induction, and support vector machines.

However, the insufficiency of known genes causes trouble to many algorithms to produce accurate prediction model. Some gene finders find most of the genes, but have a significant number of false positives. We thus propose a novel setting of constraint logic programming to formulate a

computational method toward the problem of gene searching and recognition in DNA sequences. This new scheme of DNA sequence analysis has just recently gained interest with some preliminary work appeared in the literature.

We concentrate our research study on the early biological process of gene detection and prediction because the understanding of gene structure and its function is important to the subsequent knowledge of protein analysis. Most of previous constraint-based work has based their constraint implementation on the constraint handling rules. The proposed techniques of our constraint solvers are mostly constraint propagation and search procedures embedded in the libraries of a finite and symbolic domains of the logic-based constraint system. Upon completion of this research project, we therefore expect to achieve some advancement to not only the computational gene-finding research area, but also to the constraint solving field.

ACKNOWLEDGMENT

This work has been supported by grants from the National Research Council of Thailand (NRCT) and the authors are supported by research funding from Suranaree University of Technology.

REFERENCES

[1] J. Watson and F. Crick, "A structure of deoxyribose nucleic acid," *Nature*, 1953, pp.171, 737.

[2] F. Crick, "Nucleic acids," *Scientific American*, vol. 197, 1957, pp. 188-200.

[3] F. Crick, "Central dogma of molecular biology," *Nature*, vol. 227, 1970, pp. 561-563.

[4] International Human Genome Sequencing Consortium, "Initial sequencing and analysis of the human genome," *Nature*, vol. 409, 2001, pp. 860-921.

[5] M. Borodovsky and J. McIninch, "GeneMark: parallel gene recognition for both DNA strands," *Computers & Chemistry*, vol. 17, no. 2, 1993, pp. 123-133.

[6] M. Borodovsky, A. Lomsadze, N. Ivanov, and R. Mills, "Eukaryotic gene prediction using GeneMark.hmm," *Current Protocols in Bioinformatics*, 2003, pp. 4.6.1-4.6.12.

[7] J. Besemer, A. Lomsadze, and M. Borodovsky, "GeneMarkS: a self-training method for prediction of gene starts in microbial genomes, implications for finding sequence motifs in regulatory regions," *Nucleic Acids Research*, vol. 29, no. 12, 2001, pp. 2607-2618.

[8] J. Besemer and M. Borodovsky, "GeneMark: web software for gene finding in prokaryotes, eukaryotes and viruses," *Nucleic Acids Research*, vol. 33, 2005, pp. W451-W454.

[9] C. Burge and S. Karlin, "Prediction of complete gene structures in human genomic DNA," *Journal of Molecular Biology*, vol. 268, 1997, pp. 78-94.

[10] T. Schiex, A. Moisan, and P. Rouze, "EuGene: an eukaryotic gene finder that combines several sources," in *Proc. JOBIM*, 2001, pp. 111-125.

[11] I. Antonov and M. Borodovsky, "GeneTack: frameshift identification in protein-coding sequences by the Viterbi algorithm," *Journal of*

Bioinformatics and Computational Biology, vol. 8, no. 3, 2010, pp. 535-551.

[12] A. Delcher, K. Bratke, E. Powers, and S. Salzberg, "Identifying bacterial genes and endosymbiont DNA with Glimmer," *Bioinformatics*, vol. 23, 2007, pp. 673-679.

[13] A. Rust, E. Mongin, and E. Birney, "Genome annotation techniques: new approaches and challenges," *Drug Discovery Today*, vol. 7, no. 11, 2002, pp. S70-S76.

[14] A. Krogh, "Two methods for improving performance of an HMM and their application for gene finding," in *Proc. Int. Conf. Intelligent Systems for Molecular Biology*, vol. 5, 1997, pp. 179-186.

[15] M. Stanke, O. Keller, I. Gunduz, A. Hayes, S. Waack, and B. Morgenstern, "AUGUSTUS: ab initio prediction of alternative transcripts," *Nucleic Acids Research*, vol. 34, 2006, pp. W435-W439.

[16] T. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," *Advances in Neural Information Processing Systems*, vol. 11, 1999, pp. 487-493.

[17] A. Zien, G. Ratsch, S. Mika, B. Scholkopf, T. Lengauer, and K. Muller, "Engineering support vector machine kernels that recognize translation initiation sites," *Bioinformatics*, vol. 16, 2000, pp. 799-807.

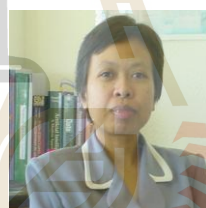
[18] N. Kerdprasop and K. Kerdprasop, "Recognizing DNA splice sites with the frequent pattern mining technique," *International Journal of Mathematical Models and Methods in Applied Sciences*, vol. 5, no. 1, 2011, pp. 87-94.

[19] Y. Sun, X. Fan, and Y. Li, "Identifying splicing sites in eukaryotic RNA: support vector machine approach," *Computers in Biology and Medicine*, vol. 33, no. 1, 2003, pp. 17-29.

[20] H. Christiansen, "Logic-statistic modeling and analysis of biological sequence data: a research agenda," in *Proc. Int. Workshop Abduction and Induction in Artificial Intelligence*, 2007, pp. 42-49.

[21] C. Rigotti, I. Mitasiunaite, J. Besson, L. Meyniel, J. Boulicaut, and O. Gandrillon, "Using a solver over the string pattern domain to analyze gene promoter sequences," in S. Dzeroski (ed.), *Inductive Databases and Constraint-Based Data Mining*, Springer-Verlag, 2010, pp. 407-423.

[22] R. Yap, "Parametric sequence alignment with constraints," *Constraints*, vol. 6, 2001, pp. 157-172.



Nittaya Kerdprasop is an associate professor at the School of Computer Engineering, Suranaree University of Technology, Thailand. She received her bachelor degree in radiation techniques from Mahidol University, Thailand, in 1985, master degree in computer science from the Prince of Songkla University, Thailand, in 1991 and doctoral degree in computer science from Nova Southeastern University, U.S.A, in 1999. Her research of interest includes knowledge discovery in databases, artificial intelligence, logic programming, and intelligent databases.



Kittisak Kerdprasop is an associate professor and chair of the School of Computer Engineering, Suranaree University of Technology, Thailand. He received his bachelor degree in mathematics from Srinakarinwirot University, Thailand, in 1986, master degree in computer science from the Prince of Songkla University, Thailand, in 1991 and doctoral degree in computer science from Nova Southeastern University, USA, in 1999. His current research includes data mining, artificial intelligence, functional and logic programming languages, computational statistics.

Visual Knowledge Mining and Utilization in the Inductive Expert System

Nittaya Kerdprasop and Kittisak Kerdprasop

Abstract—Advances in computer graphics and the human-machine visual systems have made visualization become an important tool in current data exploration and analysis tasks. Visual data mining is the combination of visualization and data mining algorithms in such a way that users can explore their data and extract the models in an interactive way. Existing visual data mining tools allow users to interactively control the three main steps of data mining: input data, explore data distribution, and extract patterns or models from data. In this paper, we propose a framework to extend these visually controlled steps to the level of model deployment. We demonstrate in this paper that both model induction and model deployment can be done through the visual method using the KNIME and Win-Prolog tools for knowledge acquisition and knowledge deployment, respectively. Model deployment presented in this paper is the utilization of induced data model as an inductive knowledge source for the inductive expert system, which is the next generation of knowledge base systems that integrate automatic learning ability in their knowledge acquisition part.

Keywords— Visual data mining, Inductive learning, Inductive expert system, Visual logic program.

I. INTRODUCTION

VISUAL data mining is an automatic and intelligent data analysis technique that utilizes visualization as a means to communicate between user and the computer to explore data and to extract hidden patterns from stored databases [33], [18]. The main benefit of visualization is that it allows easy understanding for novice users and it is also natural to human perception [15], [28], [36]. Recent trend in intelligent manufacturing and other engineering fields [8], [13], [17], [32] is to apply data mining techniques to automatically identify patterns and causal relationships that are too obscure and unobvious to be detected by human's eyes.

Applying data mining technique to high dimensional and

large amount data is however not a straightforward task because the induced patterns are normally low accurate if the input data are not well prepared or not in an appropriate form. Numerous available learning algorithms and many data preparation techniques supported by most data mining systems are also a hindrance to users who are unfamiliar with the knowledge discovery process.

We thus illustrate in this paper a natural way to do data mining through visualization. We also propose a semi-automatic technique to transfer the data mining output to be a knowledge base content in the inductive expert system.

The main characteristic of current expert systems is the separation of a knowledge base that may be changed from one application to another from the inference engine that still remains the same across applications. The delay in the development of many expert systems is due to the difficulty in acquiring and eliciting knowledge from the human domain experts [11], [20].

The concept of inductive expert system is thus been devised to overcome such bottleneck by incorporating automatic knowledge acquisition module in the system. According to this new concept, knowledge can now be induced or learned in an automatic way from archived databases that are normally available in most organizations. In this paper, we propose an architecture of the inductive expert system that includes the knowledge mining engine part to automatically forming expert rules from the stored data. The learned knowledge can be visually transformed to be the knowledge base and automatically encoded as inference rules of the inductive expert system. We show in this paper that the processes of knowledge mining and knowledge acquisition in the inductive expert system can be done through the support of available visual tools, namely the KNIME system [7] and the Win-Prolog [23].

II. PRELIMINARIES AND RELATED WORK

A. The Visual Data Mining Tools

Data mining is the search for useful patterns that normally are difficult to be recognized by human's eyes due to the large amount of data items stored in the databases. Most algorithms used in the data mining software construct a mathematical model from the data instances for the purpose of describing common patterns or predicting some unknown attribute values

Manuscript received May 20, 2014; Revised version received July 4, 2014. This work was supported by grants from the National Research Council of Thailand (NRCT) and Suranaree University of Technology through the funding of Data Engineering Research Unit.

N. Kerdprasop is an associate professor and the director of Data Engineering Research Unit, School of Computer Engineering, Suranaree University of Technology, 111 University Avenue, Muang District, Nakhon Ratchasima 30000, Thailand (phone: +66-44-224-432; fax: +66-44-224-602; e-mail: nittaya@sut.ac.th).

K. Kerdprasop is with the School of Computer Engineering and Data Engineering Research Unit, Suranaree University of Technology, Nakhon Ratchasima, Thailand (e-mail: KittisakThailand@gmail.com).

in the future cases. Model construction is a core step of data mining process. To train the learning algorithm to construct an accurate predictive model needs some parameter tuning, which is absolutely not an easy task for casual users. Therefore, modern data mining software provides some graphical interfaces to help users in controlling the process [4], [10], [21].

Visualization can be used in several stages of data mining: data exploration, analysis, and knowledge representation. The existing visual exploring and analytic tools [2], [3], [30] are fast and effective enough for the interactive induction of hidden knowledge. In this work, we adopt the KNIME visual data mining system [7] for knowledge mining.

B. Expert System and Data Mining

Since the release of DENDRAL in the 1960s from the Stanford Heuristic Programming Project [22] as the first practical knowledge-driven program, expert systems have enormously proliferated and been applied to all areas of computer-based problem solving [34]. The inventors of DENDRAL system have introduced the novel and important concept of knowledge base separation in that the content of knowledge could be added and refined independently from the program module. This module is called the inference engine responsible for interpreting and using the knowledge. The loosely coupling of a knowledge base and an inference engine is an influential concept to all successor rule-based expert systems such as MYCIN [35], INTERNIST-1 [26], and many others [14], [16].

Since the 1980s expert systems, also called knowledge-based systems, have shifted from the medical and scientific application domains to various areas. In manufacturing, mechanical analysis, and other engineering applications, rule-based expert systems are commonly applied to solve optimization problems, diagnose equipment failures, plan manufacturing scheduling, and other stages of the manufacturing process [6].

The increasing popularity of rule-based expert systems is due to the simplicity of the *if-then* rules that are easy to comprehend by humans. Many expert system tools such as Clips and Jess are available as a rule engine to facilitate rule generation for a knowledge base. These tools help facilitating the part of knowledge representation, but knowledge acquisition and elicitation are still the labor-intensive tasks for most knowledge engineers.

Modern expert system development process has thus moved toward the automating methodology by applying intelligent knowledge extraction techniques [12], [29]. Such intelligent techniques can be acquired through the machine learning and data mining technologies. There have been increasing numbers of research work attempting to apply learning techniques to automatically extract and elicit knowledge [1], [19], [27], [37]. These attempts have pushed the current expert system technology to the next generation of an inductive expert

system in the sense that besides the knowledge base and the inference engine, the system now includes the learning component.

The research work presented in this paper takes the same direction as most researchers in an attempt to automate knowledge extraction and elicitation with machine learning and data mining techniques. Our work, however, is different from others in that not only proposing an architecture of the learnable inductive expert system, but we also cover the knowledge mining from existing databases, knowledge transfer as a set of rules to be stored in the knowledge base, and knowledge reasoning through a logic-based inference engine. The process of knowledge mining and knowledge utilization have been demonstrated through the adoption of existing visual tools.

III. BRIDGING MINING MODEL WITH THE KNOWLEDGE ACQUISITION OF INDUCTIVE EXPERT SYSTEM

Knowledge mining [24], [25] is the discovery of hidden knowledge stored possibly in various forms and places in large data repositories. The whole process of knowledge mining works around data, meta-data, and previously discovered patterns. It can be conceptually shown as in Figure 1.

The initial step of knowledge mining focuses on setting the mining goal which can be achieved through understanding the task objectives and organization requirements. Problem defining is important because it will guide activities in subsequent steps to collect only relevant data, to do mining with appropriate algorithm, and to keep only pertinent and actionable knowledge.

The second step covers all activities necessary for preparing high quality data suitable for mining algorithm. This data preparation step includes collecting data from multiple sources, transforming the data format, selecting data representatives with minimum but sufficient attributes. Data preparation is typically time consuming and likely to be performed iteratively. Meta-data and background knowledge are kinds of supportive information that can be applied in this step.

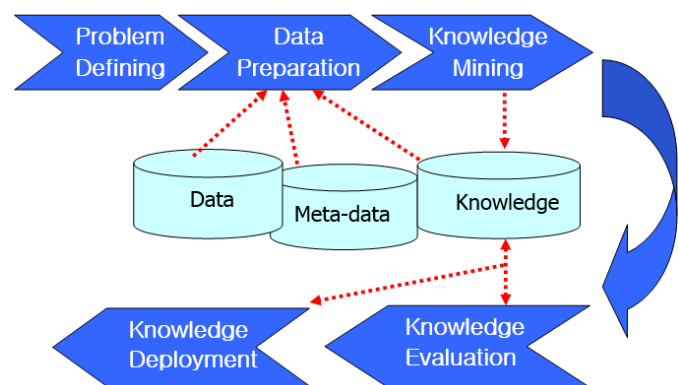


Fig. 1 The knowledge mining process

The third step is knowledge mining which is the search and extraction of interesting patterns (local generalized structures) or models (global generalized structures) from data. Such patterns and models are called knowledge. This step is the backbone of the knowledge mining process.

The fourth step is for evaluating accuracy, significance, and interestingness of the discovered knowledge based on some threshold values. The accurate, significant, and interesting knowledge is finally fed to the deployment step to be actionable information for the organization, or it can even be put back into the repositories to be background knowledge for other knowledge mining tasks.

We design (in Figure 2) an architecture of the inductive expert system to include the knowledge engine facility. Knowledge induction phase is the back-end of the system responsible for acquiring and discovering new and useful knowledge. Usefulness is to be validated at the final step by human experts. Discovered knowledge is stored in the knowledge base to be applied to solve new cases in knowledge inferring phase which is the front-end of the proposed system.

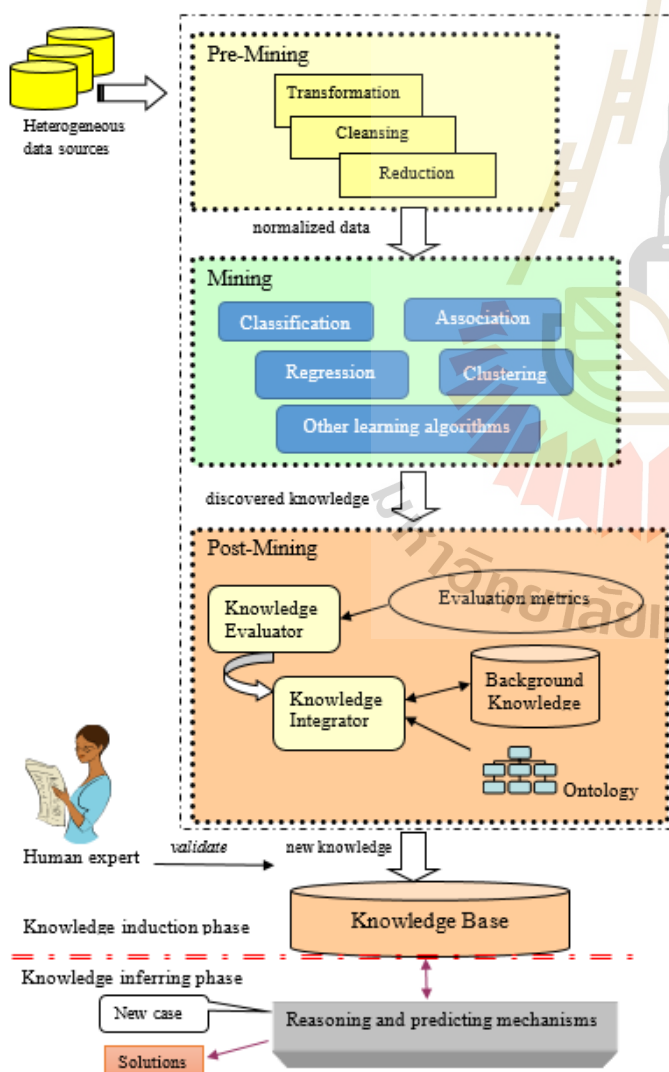


Fig. 2 Architecture of an inductive expert system

The knowledge induction phase is comprised of three main modules: pre-mining, mining, post-mining. The term *mining* means automatic learning of patterns or models from specific data. *Pattern* is an expression describing a subset of the data. For example, $f(x) = 3x^2 + 3$ is a pattern induced from a given 2-dimensional dataset $\{(0,3), (1,6), (2,15), (3,30), (4,51)\}$, whereas the term *model* refers to a function representing the source that generates the data. For this example, the model is $f(x) = ax^2 + b$. In his paper we refer to both patterns and models as new knowledge discovered from data sources.

The pre-mining module performs data preparation tasks such as

- locate and access relevant data sets,
- transform the data format,
- clean the data if there exists noise and missing values,
- reduce the data to a reasonable and sufficient size with only relevant attributes.

The mining module performs mining tasks including classification, regression, clustering, association, and other learning task. The post-mining module is composed of two main components: knowledge evaluator and knowledge integrator. These components perform major functionalities aiming at a feasible deployment of the discovered knowledge.

Knowledge evaluator involves evaluation, based on corresponding measurement metrics, of the mining results. Knowledge integrator examines the induced patterns to remove redundant knowledge. Ontology has also been applied at this step to provide essential semantics regarding the domain problems.

IV. DEMONSTRATION AND RESULTS

A. Data Set

The purpose of this experimentation is to illustrate the proposed automatic knowledge base creation method with real data. We use a car evaluation data set [9], [38] obtain from the UCI Machine Learning Repository [5]. In this data set, each car is to be evaluated its acceptability level as either very good (vgood), good, acceptable (acc), or unacceptable (unacc).

The car acceptability has been evaluated from the six attributes: the buying price (buying), price of maintenance (maint), number of doors (doors), capacity in terms of persons to carry (persons), the size of luggage boot (lug_boot), and the estimated safety of the car (safety). This data set has 1728 data instances. Examples of data instances are shown in Table I.

This data set has been used in this paper as a training set for constructing a conceptual model of car acceptability decision based on the price and other technical characteristics. Class distribution of each acceptability levels is as follows: unacc = 70.02%, acc = 22.22%, good = 3.99%, and v-good = 3.76%.

Table 1. Some instances of a car evaluation data set

buying	maint	doors	persons	lug_ boot	safety	class
vhigh	vhigh	2	2	small	low	unacc
high	high	4	more	small	low	unacc
vhigh	med	2	4	big	high	acc
high	high	4	4	big	med	acc
med	low	4	more	small	high	good
med	low	4	4	big	high	vgood

B. Visual Data Mining and Its Result

The car evaluation data set has been mined with the visual data mining tool named KNIME [7]. The visual data mining process is illustrated in Figure 3, and the mining result as decision tree is shown in Figure 4. The learning algorithm used in this demonstration is the decision tree induction algorithm [31] because of its efficiency. Moreover, the structure of the induced tree is appropriate for generating reasoning and explanation part in the expert system shell.

The steps graphically shown in Figure 3 are the process to generate a decision tree model. The first step is to read the input data; this can be done through the icon 'File Reader'. We then partition the input data into the training set and the test set (through the 'Partitioning' icon). The model induction part is accomplished through the use of 'Decision Tree Learner' icon. The output of this process is the learned knowledge to be used by the expert system shell. The 'Decision Tree Predictor' and 'Scorer' icons are used only for evaluating the accuracy of the tree model. Evaluation result of the model accuracy is shown in Figure 5.

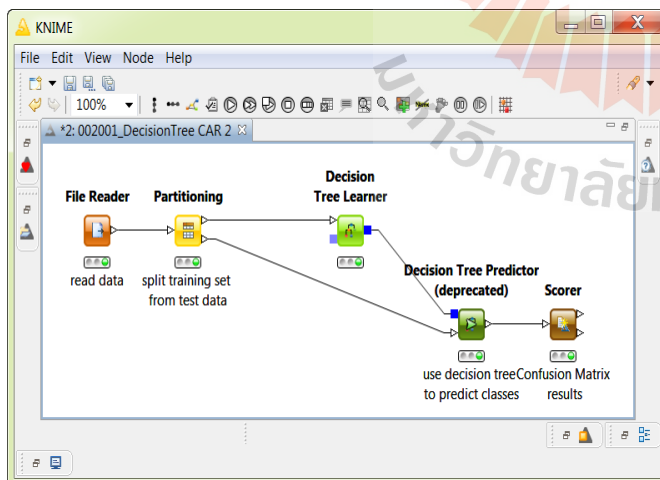


Fig. 3 Data mining process through the connection of visual icons in KNIME system

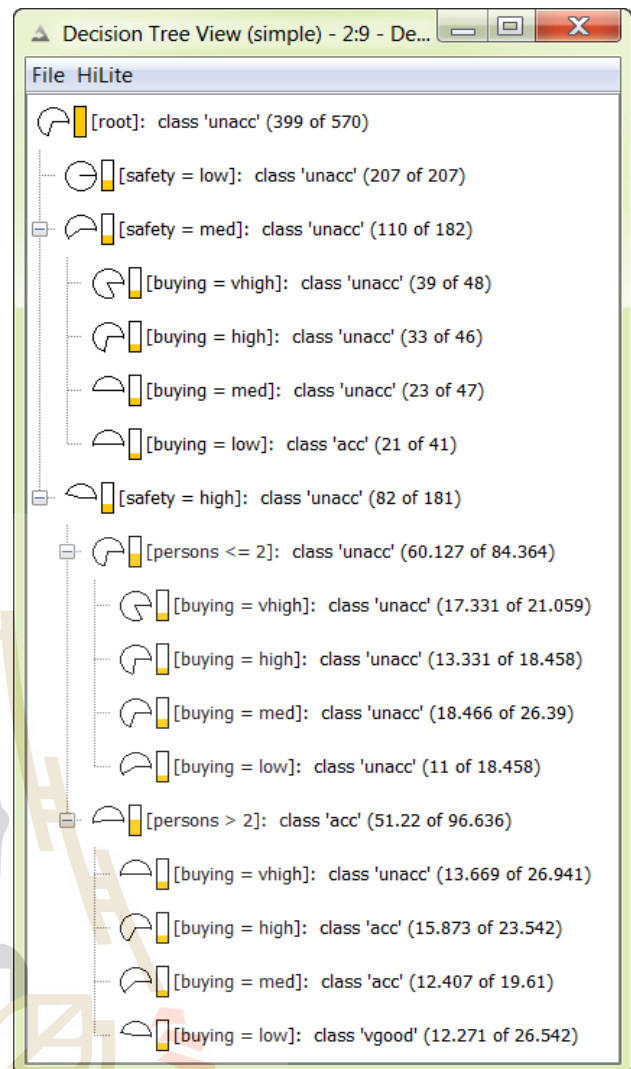


Fig. 4 A decision tree model to classify acceptability of a car as unacc/acc/vgood

C. Visual Knowledge Acquisition and Consultation

The induced knowledge as a decision tree is subsequently to be transformed into a format of decision rules by the Win-Prolog visual tool [23]. The rule generation can be done automatically. These rules are to be used by the inference engine for giving recommendation to users. These rules are also capable of giving explanation when requested by the users.

The knowledge acquisition starts when the decision tree has been built and output by the KNIME tool. The output from KNIME (Figure 4) has to be manually transformed into a format understandable by the Win-Prolog. The transformation is however easy via the support of visualization tools. The transformed rules are graphically shown in Figure 6. The meaning of the two formats is the same; only graphical representation is slightly different.

Accuracy statistics - 2:5 - Scorer(Confusion Matrix)

Row ID	TrueP...	FalseP...	TrueN...	False...	D Recall	D Precisi...	D Sensit...	D Specifity	D F-mea...	D Accur...	D Cohen...
unacc	757	197	150	54	0.933	0.794	0.933	0.432	0.858	?	?
acc	85	89	812	172	0.331	0.489	0.331	0.901	0.394	?	?
vgood	10	20	1094	34	0.227	0.333	0.227	0.982	0.27	?	?
good	0	0	1112	46	0	?	0	1	?	?	?
Overall	?	?	?	?	?	?	?	?	?	0.736	0.32

Fig. 5 Accuracy evaluation results of the induced decision tree model

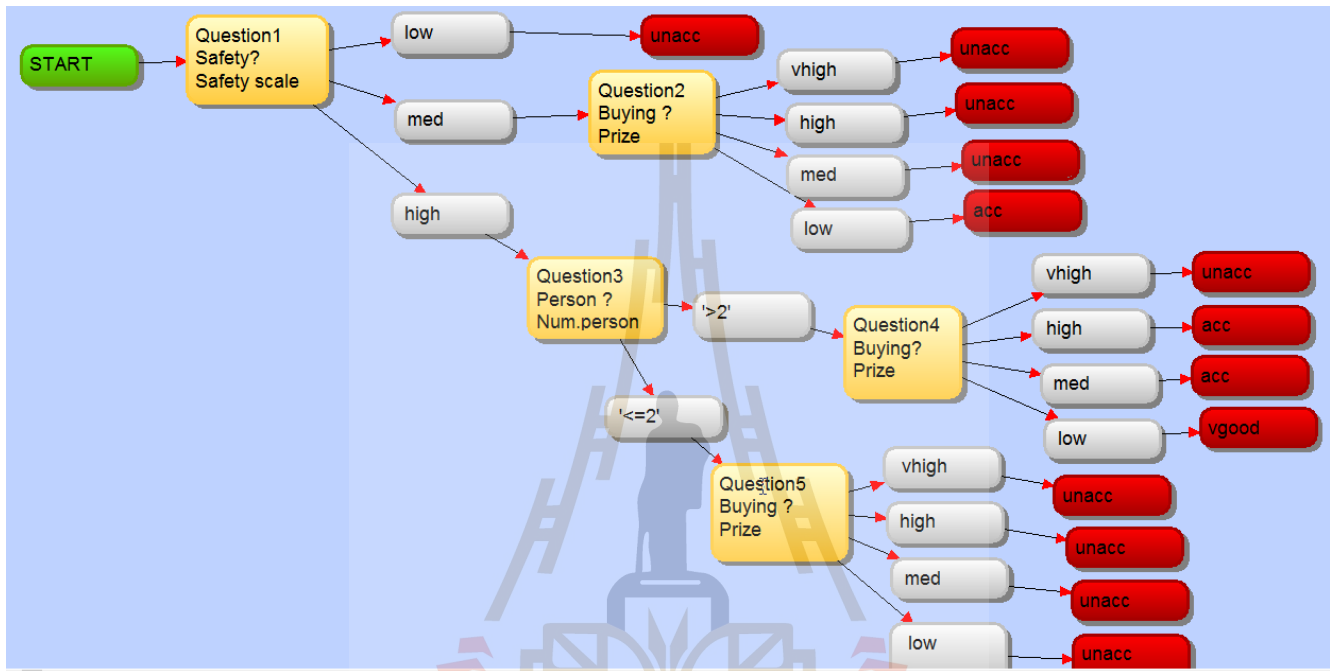


Fig. 6 A decision tree model in a Win-Prolog format

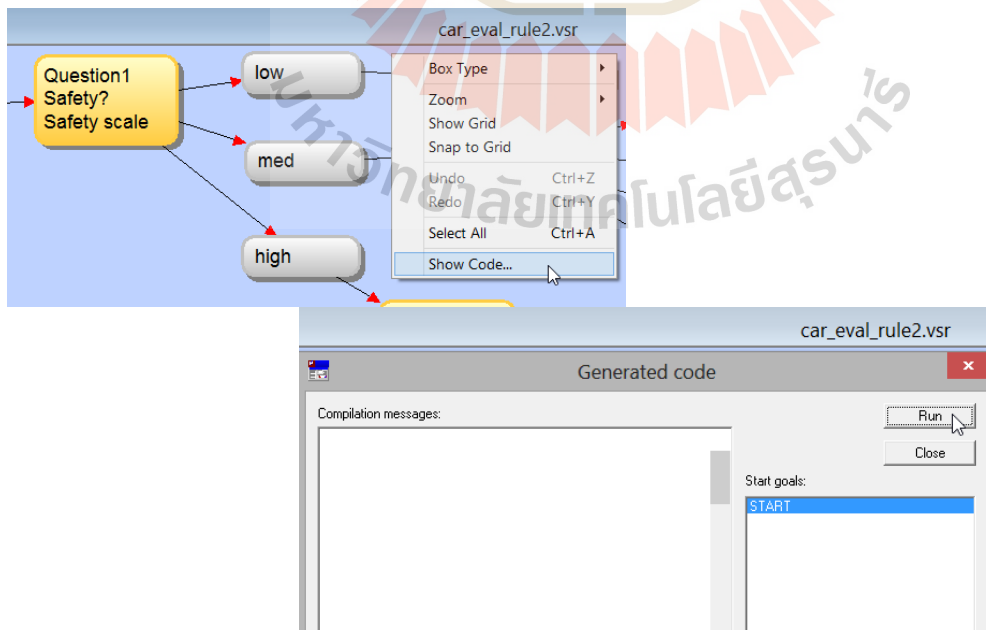


Fig. 7 Steps to generate Prolog code from a given decision tree model

To generate expert system shell for a specific decision tree model, we run the Win-Prolog and use the command:

```
load_files(system(visirule)).
```

Then, click 'File' to open the saved 'VisiRule Files'. In this demonstration, our model (Figure 6) has been saved in a file named 'car_eval_rule2.vsr'. When the model appears, user can right-click on the screen to choose 'Show Code...' and click 'Run' button on the 'Generated code' window. These steps are graphically shown in Figure 7.

The Prolog code automatically generated from the visual form of a decision tree model is as follows:

```
do ensure_loaded( system(vrllib) ) .

relation 'START'( Conclusion ) if
  q_Question1( Conclusion ) .

relation q_Question1( Conclusion ) if
  the answer to 'Question1' is _ and
  check( 'Question1', =, high ) and
  q_Question3( Conclusion ) .

relation q_Question1( Conclusion ) if
  the answer to 'Question1' is _ and
  check( 'Question1', =, low ) and
  Conclusion = unacc .

relation q_Question1( Conclusion ) if
  the answer to 'Question1' is _ and
  check( 'Question1', =, med ) and
  q_Question2( Conclusion ) .

relation q_Question3( Conclusion ) if
  the answer to 'Question3' is _ and
  check( 'Question3', =, '<=2' ) and
  q_Question5( Conclusion ) .

relation q_Question3( Conclusion ) if
  the answer to 'Question3' is _ and
  check( 'Question3', =, '>2' ) and
  q_Question4( Conclusion ) .

relation q_Question5( Conclusion ) if
  the answer to 'Question5' is _ and
  check( 'Question5', =, vhigh ) and
  Conclusion = unacc .

relation q_Question5( Conclusion ) if
  the answer to 'Question5' is _ and
  check( 'Question5', =, high ) and
  Conclusion = unacc .

relation q_Question5( Conclusion ) if
  the answer to 'Question5' is _ and
  check( 'Question5', =, low ) and
  Conclusion = unacc .

relation q_Question5( Conclusion ) if
  the answer to 'Question5' is _ and
  check( 'Question5', =, med ) and
  Conclusion = unacc .

relation q_Question4( Conclusion ) if
  the answer to 'Question4' is _ and
  check( 'Question4', =, high ) and
  Conclusion = acc .

relation q_Question4( Conclusion ) if
  the answer to 'Question4' is _ and
  check( 'Question4', =, vhigh ) and
  Conclusion = unacc .

relation q_Question4( Conclusion ) if
  the answer to 'Question4' is _ and
  check( 'Question4', =, med ) and
  Conclusion = acc .

relation q_Question4( Conclusion ) if
  the answer to 'Question4' is _ and
  check( 'Question4', =, low ) and
  Conclusion = vgood .

relation q_Question2( Conclusion ) if
  the answer to 'Question2' is _ and
  check( 'Question2', =, vhigh ) and
  Conclusion = unacc .

relation q_Question2( Conclusion ) if
  the answer to 'Question2' is _ and
  check( 'Question2', =, high ) and
  Conclusion = unacc .

relation q_Question2( Conclusion ) if
  the answer to 'Question2' is _ and
  check( 'Question2', =, med ) and
  Conclusion = unacc .

relation q_Question2( Conclusion ) if
  the answer to 'Question2' is _ and
  check( 'Question2', =, low ) and
  Conclusion = acc .

group group1
  low, high, med .
question 'Question1'
  'Safety?' ;
choose one of group1
  because 'Safety scale' .

group group2
  vhigh, high, med, low .
question 'Question4'
  'Buying?' ;
choose one of group2
  because 'Prize' .

group group3
  '<=2', '>2' .
question 'Question3'
  'Person ?' ;
choose one of group3
  because 'Num.person' .

group group4
  vhigh, high, low, med .
question 'Question5'
  'Buying ?' ;
choose one of group4
  because 'Prize' .

question 'Question2'
  'Buying ?' ;
```

```
choose one of group2
because 'Prize' .
```

The generated Prolog program acts as an inference engine of the expert system. We test the recommendation given by the system for two cases: unacceptable car and a very good car. The unacceptable case (Figure 8) has been recommended after asking for only one question. That is, for the low safety car, it is unacceptable.

For the second case, we provide the system the following information:

Safety = high
Number of persons that a car can carry > 2
Buying price = low

The recommendation given by the system is that the acceptability level of this car is very good (shown in Figure 9). As the inference engine is encoded in Prolog language that has the inherent ability of backtracking, the system can also search for other solutions if they exist (shown in Figure 10).

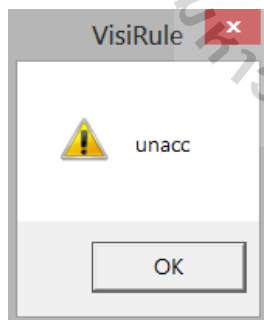
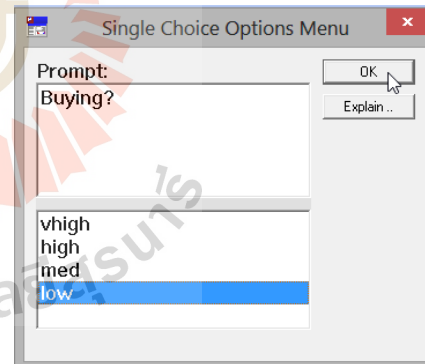
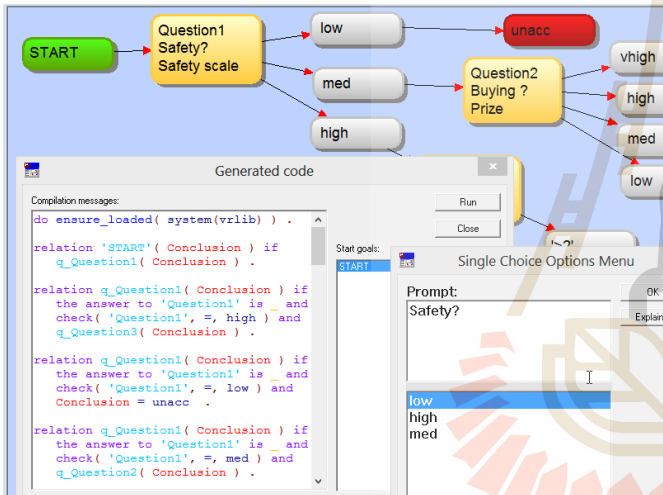
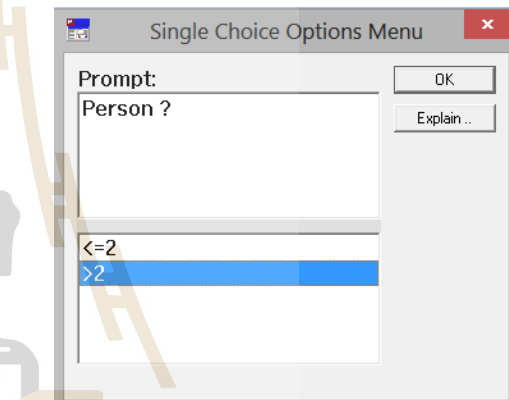
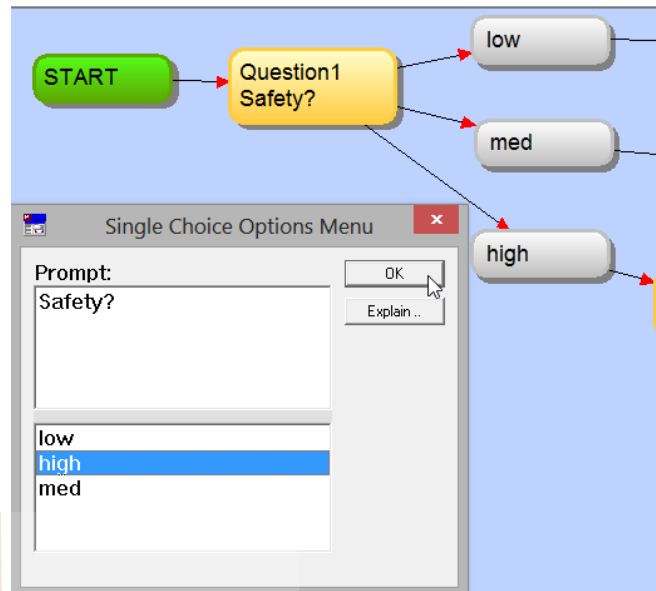


Fig. 8 The case of unacceptable car decision

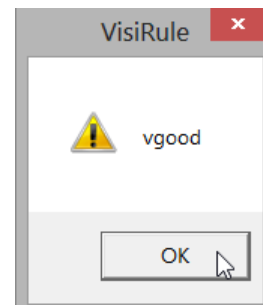


Fig. 9 The case of a very good car decision

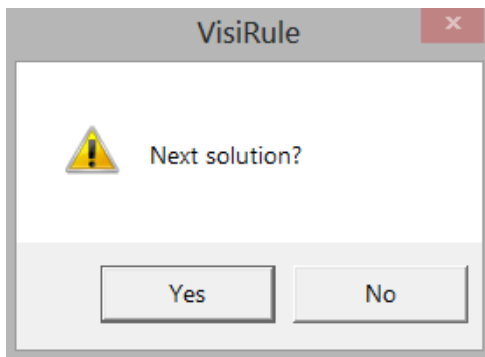


Fig. 10 Screenshot asking user for alternative solutions

From the experiments, we can observe that the generated expert system shell contains both the rule based knowledge and the inference ability. These rules are specific to the knowledge model, which is a decision tree in this demonstration. If the knowledge model has been changed, the generated rules would be changed accordingly.

V. CONCLUSION

Artificial intelligence, specifically expert systems, has played an important role in solving complex engineering, manufacturing, medicine, and many other problems for more than four decades. Knowledge base and inference procedures have been employed to solve the problems that require significant human expertise and domain-specific knowledge. The required knowledge has to be elicited by knowledge engineers. It is a labor-intensive task, and thus a bottle neck in building intelligent systems.

We propose to apply data mining technique as a major step in a knowledge engine component of the inductive expert system to assist the knowledge elicitation task. The proposed technique is a novel method for automating knowledge acquisition that help supporting manufacturing and other intelligent systems. We demonstrate knowledge mining through the visual tool called KNIME, which has many visualization features to support users who are not an expert in data mining.

Knowledge as a learned tree structure is then transformed by another visual tool called Win-Prolog to generate a Prolog program as a rule set that can be integrated into the knowledge base. The demonstration given in this paper has proved applicability and appropriateness for inferring and reasoning from the knowledge base that can be automatically induced from the stored database.

REFERENCES

- [1] F. Alonso, L. Martinez, A. Perez, and J.P. Valente, "Cooperation between expert knowledge and data mining discovered knowledge: Lesson learned," *Expert Systems with Applications*, vol.39, 2012, pp.7524-7535.
- [2] A. Ammoura, O. Zaiane, and Y. Ji, "Immersed visual data mining: walking the walk," in *Proceedings of the 18th British National Conference on Databases*, Chilton, U.K., 2001, pp. 202-218.
- [3] R. Anderson, *Visual Data Mining: The VisMiner Approach*. Wiley, 2012.
- [4] M. Ankerst, C. Elsen, M. Ester, and H. Kriegel, "Visual classification: an interactive approach to decision tree construction," in *Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining*, 1999, pp. 392-396.
- [5] K. Bache, and M. Lichman, *UCI Machine Learning Repository*, <http://archive.ics.uci.edu/ml>, University of California, Irvine, 2013.
- [6] A. Badiru, *Expert Systems: Applications in Engineering and Manufacturing*, Prentice-Hall, 1992.
- [7] M. Berthold, N. Cebron, F. Dill, T. Gabriel, T. Kotter, T. Meini, P. Ohl, C. Sieb, K. Thiel and B. Wiswedel, "KNIME: The Konstanz Information Miner," in *Proceedings of the 31st Annual Conference of the Gesellschaft für Klassifikation e.V.*, Albert-Ludwigs-Universität Freiburg, 2007, pp.319-326.
- [8] F. Bezerra, and J. Wainer, "Algorithms for anomaly detection of traces in logs of process aware information systems," *Information Systems*, vol. 38, 2013, pp. 33-44.
- [9] M. Bohanec, and V. Rajkovic, "Knowledge acquisition and explanation for multi-attribute decision making," in *Proceedings of the 8th International Workshop on Expert Systems and Their Applications*, France, 1988, pp. 59-78.
- [10] M. de Oliveira, and H. Levkowitz, "From visual data exploration to visual data mining: a survey," *IEEE Transactions on Visualization and Computer Graphics*, 9, 3, 2003, pp.378-394.
- [11] E.A. Feigenbaum, "Expert systems: principles and practice," in *The Encyclopedia of Computer Science and Engineering*, Stanford: Knowledge Systems Laboratory, 1992.
- [12] E. Flior, T. Anaya, C. Moody, M. Beheshti, H. Jianchao, and K. Kowalski, "A knowledge-based system implementation of intrusion detection rules," in *Proceedings of the 7th International Conference on Information Technology: New Generations*, 2010, pp. 738-742.
- [13] J. Fodor, R. Klempous and C. Suárez Araujo: *Recent Advances in Intelligent Engineering Systems*, Springer, 2012.
- [14] J.C. Giarratans and G.D. Riley, *Expert systems: Principles and Programming*, fourth edition, Canada: Thomason Learning, 2005.
- [15] Y. Gingchi, S. Chao, and H. Yang, "Identifying the technology trend of visual language researches," *WSEAS Transactions on Computers*, vol. 9, issue 11, 2010, pp. 1369-1379.
- [16] R. Girratano, *Expert Systems, Principles and Programming*, PWS, 1998.
- [17] L.-C. Huang, S.-S. Tseng, and Y.-S. Chu, "Building a wafer fab lot scheduling knowledge-based system," *WSEAS Transactions on Information Science and Applications*, vol. 3, no. 10, 2006, pp. 1994-2001.
- [18] D. Keim, "Information visualization and visual data mining," *IEEE Transactions on Visualization and Computer Graphics*, 8, 1, 2002, pp.1-8.
- [19] C. Leon, F. Biscarri, I. Monedero, J.I. Guerrero, J. Biscarri, and R. Millan, "Integrated expert system applied to the analysis of non-technical losses in power utilities," *Expert Systems with Applications*, vol. 38, 2011, pp. 10274-10285.

- [20] S.-H. Liao, "Expert system methodologies and applications – a decade review from 1995 to 2004," *Expert Systems with Applications*, vol. 28, 2005, pp. 93-103.
- [21] J. Lin, E. Keogh, S. Lonardi, J. Lankford, and D. Nystrom, "VizTree: a tool for visually mining and monitoring massive time series databases," in *Proceedings of the 30th VLDB Conference*, Toronto, Canada, 2004, pp. 1269-1272.
- [22] R. K. Lindsay, B. G. Buchanan, E. A. Feigenbaum, and J. Lederberg, "DENDRAL: A case study of the first expert system for scientific hypothesis formation," *Artificial Intelligence*, vol.61. no.2, 1993, pp.209-261.
- [23] Logic Programming Associates, Win-Prolog, <http://www.lpa.co.uk>
- [24] P. Markellou, M. Rigou, and S. Sirmakessis, "Knowledge mining: A quantitative synthesis of research results and findings," in S. Sirmakessis (Ed.), *Knowledge Mining*. The Netherlands: Springer-Verlag, 2005.
- [25] R. Michalski, "Knowledge mining: A proposed new direction," *Invited Talk at the Sanken Symposium on Data Mining and Semantic Web*, Osaka University, Japan, 10-11 March 2003. Retrieved from <http://www.mli.gmu.edu/papers/2003-2004/03-5.pdf>
- [26] R.A. Miller, H.E. Pople, and J.D. Myers, "INTERNIST-1, an experimental computer-based diagnostic consultant for general internal medicine," *New England Journal of Medicine*, vol. 307, no. 8, 1982, pp. 468-476.
- [27] A. H. Mohammad and N. A. M. Al Saiyd, "A framework for expert knowledge acquisition," *International Journal of Computer Science and Network Security*, vol.10, no.11, 2010, pp.145-151.
- [28] K. Navarro, E. Lawrence, and D. Martin, "Visual categorization of brain computer interface technologies," *WSEAS Transactions on Information Science and Applications*, vol. 2, issue 8, 2005, pp. 1184-1188.
- [29] R. A. Perez, L. O. Hall, S. Romaniuk, and J. T. Lilkendey, "Inductive learning for expert systems in manufacturing," in *Proceedings of the 25th Hawaii International Conference on System Sciences*, 1992, pp.14-25.
- [30] K. Puolamaki, P. Papapetrou, and J. Lijffijt, "Visually controllable data mining methods," in *Proceedings of the IEEE International Conference on Data Mining Workshops*, Sydney, Australia, 2010, pp. 409-417.
- [31] J. Quinlan, "Induction of decision trees," *Machine Learning*, 1, 1, 1986, pp.81-106.
- [32] N. Ruschin-Rimini, O. Maimon and R. Romano, "Visual analysis of quality-related manufacturing data using fractal geometry," *Journal of Intelligent manufacturing*, 23, 3, 2012, pp.481-495.
- [33] S. Simoff, M. Bohlen, and A. Mazeika (Eds.), *Visual Data Mining: Theory, Techniques and Tools for Visual Analytics*, Springer, 2008.
- [34] W.B. Schwartz, "Medicine and the computer: The promise and problems of changes," *New England Journal of Medicine*, vol. 283, 1970, pp. 1257-1264.
- [35] E.H. Shortliffe, *Computer-Based Medical Consultations: MYCIN*, Elsevier, 1976.
- [36] A. Tsolakidis, C. Sgouropoulou, E. Papageorgiou, O. Terraz, and G. Miaoulis, "Using visual representation for decision support in institutional research evaluation," in D. Plemenos, and G. Miaoulis (Eds.), *Intelligent Computer Graphics 2012*, Springer, 2012, pp. 41-57.
- [37] T. Witkowski, P. Antezak and A. Antezak, "Machine leaning-based classification in manufacturing system," in *Proceedings of the 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, Czech Republic, 2011, pp.580-585.
- [38] B. Zupan, M. Bohanec, I. Bratko, and J. Demsar, "Machine learning by function decomposition," in *Proceedings of the 14th International Conference on Machine Learning*, Nashville, U.S.A., 1997, pp. 421-429.

Nittaya Kerdprasop is an associate professor and the director of Data Engineering Research Unit, School of Computer Engineering, Suranaree University of Technology, Thailand. She received her B.S. in radiation techniques from Mahidol University, Thailand, in 1985, M.S. in computer science from the Prince of Songkla University, Thailand, in 1991 and Ph.D. in computer science from Nova Southeastern University, U.S.A., in 1999. She is a member of IAENG, ACM, and IEEE Computer Society. Her research of interest includes Knowledge Discovery in Databases, Data Mining, Artificial Intelligence, Logic and Constraint Programming, Deductive and Active Databases.

Kittisak Kerdprasop is an associate professor at the School of Computer Engineering and one of the principal researchers of Data Engineering Research Unit, Suranaree University of Technology, Thailand. He received his bachelor degree in Mathematics from Srinakarinwirot University, Thailand, in 1986, master degree in computer science from the Prince of Songkla University, Thailand, in 1991 and doctoral degree in computer science from Nova Southeastern University, USA, in 1999. His current research includes Data mining, Machine Learning, Artificial Intelligence, Logic and Functional Programming, Probabilistic Databases and Knowledge Bases.

THE DISCOVERY OF TOP-K DNA FREQUENT PATTERNS WITH APPROXIMATE METHOD

Nittaya Kerdprasop and Kittisak Kerdprasop

*Data Engineering and Knowledge Engineering Research Units,
School of Computer Engineering, Suranaree University of Technology,
Nakhon Ratchasima, Thailand
nittaya@sut.ac.th, kittisakThailand@gmail.com*

Abstract

Top-k frequent pattern discovery is indeed an association analysis concerning automatic extraction of the k most correlated and interesting patterns from large databases. Current studies in association mining concentrate on how to effectively find all objects that are frequently co-occurring. Given a set of objects with m features, there are almost 2^m frequent patterns to consider. For DNA data that are normally very high in dimensionality, frequent pattern discovery from genetic data is obviously a computationally expensive problem. We therefore devise an approximate approach to tackle this problem. We propose an approximate method based on the window sliding concept to estimate data density and obtain data characteristics from a small set of samples. Then we draw a set of representatives with reservoir sampling technique. These representatives are subsequently used in the main process of frequent pattern mining. Our designed algorithm had been implemented with the Erlang language, which is the functional programming paradigm with inherent support for pattern matching. The experimental results confirm the efficiency and reliability of our approximate method.

Keywords: *Top-k frequent patterns, Approximate method, DNA patterns, Window sliding, Reservoir sampling, Erlang language*

1. Introduction

Frequent pattern discovery is an essential operation for association analysis, which is the discovery process concerning an automatic extraction of interesting patterns and correlations from a large database. These patterns can reveal implicit relationships among set of objects (or items) that lead to the generation of association rules in a form of “if *antecedents* then *consequences*.” These rules have the potential use in medical diagnosis, customer behavioural forecast, financial decision support, and many other applications. The process of finding all frequent itemsets in a database is computationally expensive because it involves the search for all item combinations. For a data set with high dimensionality such as the genetic data, finding only top-k frequent itemsets is more practical than searching for all itemsets that meet the minimum support threshold. Top-k frequent pattern discovery (Han *et al.*, 2002) limits the search space to the k most frequently occurred patterns across the database.

In this paper, we study the top-k frequent pattern discovery in the data streaming scenario. The discovery of frequent patterns from a stream is considered a hard problem because of a continuously generated nature of stream that does not allow a revisit over passing data element. Moreover, the discovery process has been required to be fast to produce immediate results. From these requirements, we thus devise an approximate approach to solve the problem of top-k pattern discovery over continuous stream using the DNA data as an illustration. Our approximate algorithm is intended to be applied to process a stream prior to the pattern discovery process. The organization of this paper is as follows. After the literature review regarding association analysis and frequent pattern mining in section 2, we present our method in section 3. The experimental results are demonstrated in section 4. We conclude our paper in section 5 with the discussion of future research direction.

2. Literature Review

Since the introduction of the AIS (Agrawal-Imielinski-Swami) algorithm (Agrawal and Srikant, 1994b) by the three members of IBM Almaden Research Center in 1993 (Agrawal *et al.*, 1993), the concept of association rule mining from transactional databases has received much interest from many data mining researchers. A year later, Rakesh Agrawal and Ramakrishnan Srikant (1994a; 1994b) improved the algorithm by reducing its search space with apriori property of the search through a frequent itemset lattice. This new algorithm has been named Apriori. The advent of Apriori algorithm is a major milestone of advancement in association analysis.

Apriori algorithm has been widely used as a basis for subsequent improvement proposed by a number of research teams. Park *et al.* (1995a) proposed to use hashing technique for the improvement of frequent itemset search. Han and Fu (1995) introduced the idea of discovering multiple levels of association rules. For a very large transactional database, Savasere *et al.* (1995) suggested to split the database and then search for associative relationships in a reduced data set. Toivonen (1996) tackled the large database problem with a sampling idea to search for interesting association from data representatives. Cheung *et al.* (1996a) considered an incremental approach for gradually learning of association among itemsets. Parallel computation is another mainstream of research to speed up association rule mining (Park *et al.*, 1995b; Agrawal and Shafer, 1996; Zaki *et al.*, 1997).

For a non-Apriori based association mining algorithm, the FP-growth algorithm that uses a tree structure to store frequent itemsets is an efficient method for extracting frequent patterns. The algorithm had been proposed by Han *et al.* (2000) and gained popularity since then (Agrawal *et al.*, 2001; Pei *et al.*, 2001; Liu *et al.*, 2002; Grahne and Zhu, 2003).

In the emerging era of cloud technology, distributed computation of frequent patterns can be effectively accomplished. The research along this line has started since the last two decades (Cheung *et al.*, 1996b) and it is still an active research area (Coenen and Leng, 2006; Tseng *et al.*, 2010; Zhu *et al.*, 2011; Lin *et al.*, 2013; Cuzzocrea *et al.*, 2014; Elayyadi *et al.*, 2014).

With the advanced mobile devices, data collection and broadcasting occur at a very high speed. The frequent pattern discovery algorithms have to deal with the new kind of data, i.e., streaming data. A data stream is a sequence of digitally encoded data that are continuously transmitted from distributed sources (Guha *et al.*, 2001; Babcock *et al.*, 2002; Gaber *et al.*, 2005; Jiang and Gruenwald, 2006). Kargupta *et al.* (2004) developed the VEDAS system to monitor vehicles at real time. Cai *et al.* (2004) designed the MAIDS system to mine incidents from data streams. Halatchev and Gruenwald (2005) proposed an estimation technique to guess missing values in sensor data streams. Finding frequent itemsets over data stream is a research problem studied by several researchers (Chang and Lee, 2004; Charikar *et al.*, 2004; Chi *et al.*, 2004; Gaber *et al.*, 2004; Ghoting and Parthasarathy, 2004; Li *et al.*, 2004; Teng *et al.*, 2004; Yu *et al.*, 2004; Lin *et al.*, 2005; Mao *et al.*, 2005).

The work presented in this paper is also along the line of distributed data stream processing to find the top-k patterns from DNA data. To estimate the frequency of top-k patterns, we adapted the Monte Carlo approximate method (Kerdprasop *et al.*, 2006). The details of our design will be discussed in the next section.

3. Approximate Method for Top-k Pattern Discovery

A framework of our approximate top-k frequent pattern discovery is presented in figure 1. Contribution of our work is the design and implementation of the approximation-via-sliding-window (figure 2) and density-biased-sampling (figure 3) algorithms, whereas the frequent pattern discovery is Apriori-based algorithm (Agrawal and Srikant, 1994). Our sampling technique is based on the reservoir concept (Vitter, 1985; Kerdprasop *et al.*, 2005), but data representatives will be drawn only from the dense area. Thresholds for minimum density and area size can be adjusted by user.

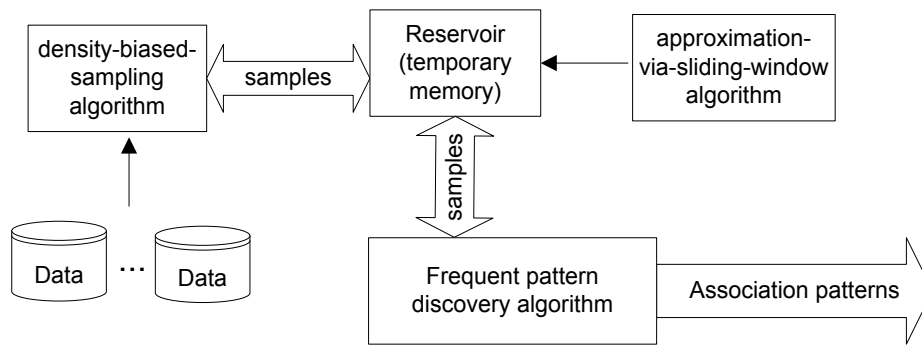


Figure 1. A framework of approximate method for top-k pattern discovery

Input:	a set of data points represented as vectors
Output:	a new set of transformed data points annotated with density value

% Initialize windows

- (1) Interact with user to obtain dimension value
- (2) Generate window grid of size W along dimension axes

% Count density

- (3) Sequential move on each window and count number of data points, N , in the window
- (4) Record a list of window's central point and its N value in a file F
- (5) Return F as a set of transformed data

Figure 2. Pseudocode of the approximation-via-sliding-window algorithm

Input:	a set of high density data from the approximation-via-sliding-window algorithm
Output:	a new set of data samples

- (1) Extract data from a condense form and obtain a desired sampling choice from user
- (2) If choice = 'Density-biased Reservoir+Hashing', then
 - (3) Interactive with user to obtain reservoir size
 - (4) Hash each data point to store in a reservoir R
 - (5) If collision occurs, then stored data item is replaced by a new one
 - (6) Repeat steps 4-5 until there is no more data point, and return R as output
- (7) If choice = 'Density-biased Reservoir+Simple Random Sampling', then
 - (8) Interact with user to obtain the bin size
 - (9) Randomly select data point to store in a reservoir R //sampling without replacement
 - (10) Repeat step 9 until R is full, and return R as an output
- (11) If choice = 'Density-biased Reservoir+Rejection Sampling', then
 - (12) Interact with user to obtain the bin size and interval I , $I \in [0.0..0.5]$
 - (13) Randomly select data point D // sampling without replacement
 - (14) Generate a uniform random number U from the range $[0.0 .. 1.0]$
 - (15) If U is within the range $[0.5-I .. 0.5+I]$, then store D in R
 - (16) Otherwise, reject and discard D
 - (17) Repeat steps 13-16 until R is full, and return R as an output

Figure 3. Pseudocode of the density-biased-sampling algorithm

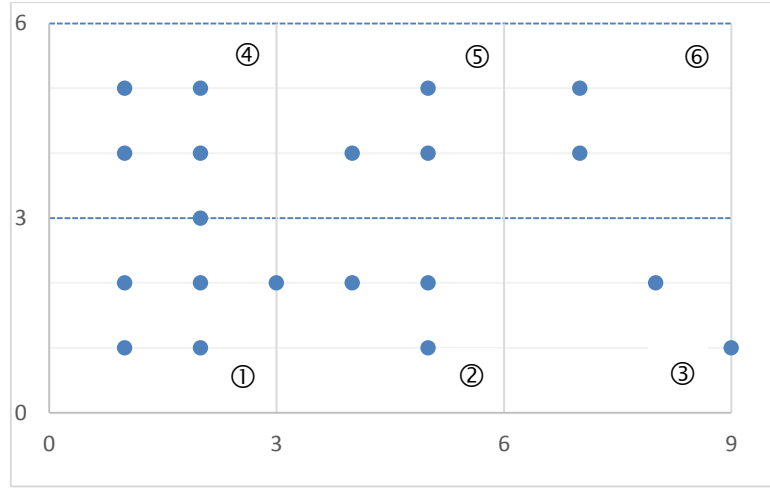


Figure 4. Twenty data points distributed within six windows of size 3×3

Our density-biased sampling technique (an algorithm in figure 2) has been designed to handle streaming in which input data are continuously processed by the system. To analyse each and every data item is almost impossible. We thus instead consider frequent patterns from the representatives. The intuitive idea of selecting representative data with the approximation-via-sliding-window algorithm can be demonstrated through a simple situation of processing a two-dimensional data set containing 20 data points, which are shown in figure 4. For the purpose of concise demonstration, we assume that the data points in this example limit themselves within the scale 9×6 along the horizontal and vertical axes, respectively.

The first step of a stream data density estimation is to decide the size of small grids, which we call windows in our algorithm. Suppose we choose the size 3×3 . The boundaries of each window can be listed with intervals in the $\langle x,y \rangle$ coordinates as follows (note that the interval such as $[0,3)$ represents the values ranging from zero up to 3, but does not include 3) :

	Range along $\langle x,y \rangle$ axes		Range along $\langle x,y \rangle$ axes
window ①:	$\langle [0,3), [0,3) \rangle$	window ④:	$\langle [0,3), [3,6) \rangle$
window ②:	$\langle [3,6), [0,3) \rangle$	window ⑤:	$\langle [3,6), [3,6) \rangle$
window ③:	$\langle [6,9), [0,3) \rangle$	window ⑥:	$\langle [6,9), [3,6) \rangle$

Data points in each window will be counted and condensed to the representation format that consumes less memory. The condensed form is per window, instead of per data point. In this condensed form, we store the central location of a window together with the number of data points existing in that window. For instance, all five data points in window ④ will be packed and stored as $\{ \langle 1.5, 4.5 \rangle, 5 \}$, where $\langle 1.5, 4.5 \rangle$ is the central point of this window. All 20 data points will be transformed as shown in Figure 5. These transformed data points that meet the minimum density requirement are the output of the approximation-via-sliding-window algorithm, and also the input for the density-biased sampling algorithm.

Raw data		Transformed data		Output
$\langle 1,1 \rangle$		$\{ \langle 1.5, 1.5 \rangle, 4 \}$		
$\langle 1,1 \rangle$		$\{ \langle 4.5, 1.5 \rangle, 4 \}$		$\{ \langle 1.5, 1.5 \rangle, 4 \}$
$\langle 1,4 \rangle$		$\{ \langle 7.5, 1.5 \rangle, 2 \}$		$\{ \langle 4.5, 1.5 \rangle, 4 \}$
$\langle 1,5 \rangle$		$\{ \langle 1.5, 4.5 \rangle, 5 \}$		$\{ \langle 1.5, 4.5 \rangle, 5 \}$
$\langle 2,1 \rangle$		$\{ \langle 4.5, 4.5 \rangle, 3 \}$		
$\langle 2,2 \rangle$		$\{ \langle 7.5, 4.5 \rangle, 2 \}$		
$\langle 2,3 \rangle$				

$\xrightarrow[\text{window size}=3 \times 3]{}$
 $\xrightarrow[\text{density threshold}=4]{}$

Figure 5. The transformation from raw data to the {central-point, density} format and the final output of approximation-via-sliding-window algorithm

Condensed data	Data representatives
{ <1.5,1.5>, 4 }	<1.5,1.5>, <1.5,1.5>, <1.5,1.5>, <1.5,1.5>
{ <4.5,1.5>, 4 }	<4.5,1.5>, <4.5,1.5>, <4.5,1.5>, <4.5,1.5>
{ <1.5,4.5>, 5 }	<1.5,4.5>, <1.5,4.5>, <1.5,4.5>, <1.5,4.5>, <1.5,4.5>

Figure 6. Data representatives that are generated back from the condensed format

The first step of the density-biased-sampling algorithm is the extraction of data points that are stored in the condensed form. After the extraction process, we obtain the representative data points as illustrated in figure 6. In the sampling step, user can choose different schemes of sample draw and temporary memory maintenance as follows:

- Density-biased reservoir + Hashing
- Density-biased reservoir + Simple random sampling
- Density-biased reservoir + Rejection sampling

A set of samples drawn from streaming data is then forwarded to the Apriori-based frequent pattern discovery algorithm (Agrawal and Srikant, 1994).

4. Experimental Results

A. DNA Data Set

The proposed approximate method has been applied to find top-k frequent patterns from the DNA data set (available at <http://archive.ics.uci.edu/ml/datasets/>). This data set contains 3,186 instances. We split the data into two parts: the first 2,000 instances to be used as a training data and the rest 1,186 instances are for testing correctness of the discovered patterns. Each data instance is a sequence of 60 genetic codes (A=adenine, T=thymine, C=cytosine, G=guanine) obtained from different location of a gene. Some data samples are displayed in figure 7.

These genetic codes can be categorized as either exon/intron, intron/exon, or none. The exon/intron is the border region of genetic codes that links the exon part to the intron part. The intron/exon can be interpreted in the same manner, but vice versa. Exon is the part containing genetic codes that control the protein synthesis. Intron is the intervening area between exons and it will later be discarded before the synthesis of proteins. The none category is the genetic string that does not bear genetic codes for protein synthesis. The structure of exon and intron in a gene is schematically shown in figure 8.

T,T,C,T,A,T,G,A,G,A,A,A,C,G,T,G,G,C,A,T,T,G,T,G,C,G,C,A,A,G,G,T,G,G,G,C,C,C, C,G,C,G,G,G,A,C,G,G,G,G,C,A,G,C,T,C,C,G,G,G,exon/intron
C,T,C,C,C,C,A,C,C,C,A,C,T,G,T,C,C,A,C,C,C,G,C,C,C,G,C,A,G,A,T,C,G,C,T,T,C,C, T,G,G,A,G,C,C,A,G,G,C,A,A,G,A,A,C,T,C,C,A,intron/exon
C,T,G,A,C,T,A,A,G,C,C,G,C,C,C,T,T,G,T,C,C,C,T,T,C,T,C,A,G,A,T,T,A,T,G,T,T,T, G,A,G,A,C,C,T,T,C,A,A,C,A,C,C,C,G,G,C,C,intron/exon
G,A,G,G,A,G,C,T,A,G,A,C,A,A,G,T,A,C,T,G,G,T,C,T,C,A,G,C,A,G,G,T,G,C,G,T,G,A, G,G,G,G,A,G,G,G,A,T,G,G,C,T,G,C,C,A,A,G,G,exon/intron
A,A,G,G,C,T,C,A,G,G,A,G,G,A,G,G,A,G,A,T,C,A,A,C,A,T,C,A,A,C,C,T,G,C,C,C,C,G, C,C,C,C,C,T,C,C,C,C,A,G,C,C,T,G,A,T,A,A,A,none

Figure 7. Some DNA data instances

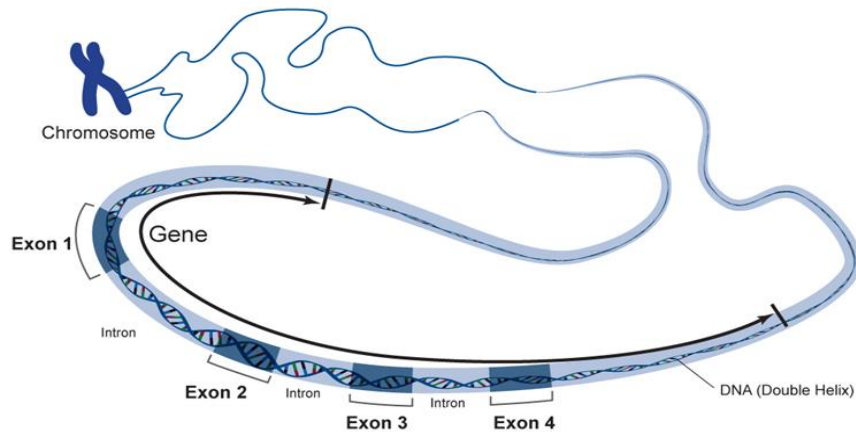


Figure 8. Structure of a gene with exon and intron parts (<http://genome.gov/Glossary/>)

B. Testing Scheme

We test the performance of our approximate method by simulating the DNA data set as a data stream, then feeding a stream to the density approximation and sampling algorithms. Data representatives are stored in a temporary memory area, called a reservoir. The representatives are finally processed by the frequent pattern discovery algorithm to find the top-k patterns. Completeness of the approximately discovered patterns is justified by the comparison against the frequent patterns that are discovered without the application of approximate method.

C. Program Running Results

We implemented our approximate frequent pattern discovery method with the Erlang programming language. The running result of the main function is shown in figure 9. Our approximate frequent pattern discovery program finds the frequent patterns of a specific class. In figure 9, we show the frequent patterns of a class intron/exon with the minimum support = 80%. At this level of support value, there are 3 frequent patterns of length 1 ($k=1$, or 1-item sets), 3 frequent patterns of length 2 ($k=2$, or 2-item sets), and 1 frequent pattern of length 3 ($k=3$, or 3-item set). These seven patterns (shown inside the red square in figure 9) can be interpreted as follows:

["AM"] means occurrence of the adenine base (A) at location 29 (ASCII code of M) in a DNA string

["CL"] means occurrence of the cytosine base (C) at location 28 (ASCII code of L) in a DNA string

["GN"] means occurrence of the guanine base (G) at location 30 (ASCII code of N) in a DNA string

["AM", "CL"] means co-occurrence of the adenine base at location 29 and cytosine base at location 28 in a DNA string

["AM", "GN"] means co-occurrence of the adenine base at location 29 and guanine base at location 30 in a DNA string

["CL", "GN"] means co-occurrence of the cytosine base at location 28 and guanine base at location 30 in a DNA string

["AM", "CL", "GN"] means co-occurrence of the adenine base at location 29, cytosine base at location 28, and guanine base at location 30 in a DNA string

```

Erlang
File Edit Options View Help
Eshell U5.7.5 (abort with ^G)
1> c(assoDNA,[export_all]).
{ok,assoDNA}
2> assoDNA:main2().

-----START-----
File 1."DNA-nominal.data" 2."DNA-nominal.test" :Choose> 1.
Read from file:"DNA-nominal.data"
Ther are 1-3 Classes :Choose> 3.
Class ="intron/exon" input percent> 80.

Total=485 ,80% MinSup=388.0
K=1-[[["AM"],484,99.79381443298969],
      [{"CL"},392,80.82474226804123],
      [{"GN"},483,99.58762886597938]], has 3 set
K=2-[[["AM","CL"],392,80.82474226804123],
      [{"AM","GN"},483,99.58762886597938],
      [{"CL","GN"},392,80.82474226804123]], has 3 set
K=3-[[["AM","CL","GN"],392,80.82474226804123]], has 1 set

AllRule=[[{"[65,77]"},{"[67,76]"},0.8099173553719008],{"[67,76]"},{"[65,77]"},1.0],{"[65,77]"},
{"[71,78]"},0.9979338842975206],{"[71,78]"},{"[65,77]"},1.0],{"[67,76]"},{"[71,78]"},1.0],
{"[71,78]"},{"[67,76]"},0.8115942028985508],{"[65,77]"},{"[67,76]"},{"[71,78]"},0.80991735537
19008],{"[65,77]"},{"[67,76]"},{"[71,78]"},1.0],{"[67,76]"},{"[71,78]"},{"[65,77]"},1.0],{"[67,76]"},
{"[71,78]"},{"[65,77]"},1.0],{"[71,78]"},{"[65,77]"},{"[67,76]"},0.8115942028985508],{"[71,78]"},
{"[65,77]"},{"[67,76]"},0.8115942028985508}],
There are 12 rules
-----ok
3>

```

Figure 9. Running result of intron/exon frequent patterns with at least 80% of occurrence frequency (that is, minimum support = 80%)

```

7> assoDNA:findSupOf(["AM","GN"]).
"DNA-nominal.data","DNA-nominal.test" Start NewJob FileName> "DNA-nominal.test".

Class:"none" has 41 = 6.799336650082918 percentOf 603
Class:"exon/intron" has 149 = 49.17491749174918 percentOf 303
Class:"intron/exon" has 278 = 99.28571428571429 percentOf 280endOfPrint
8>

```

Figure 10. The result of comparing the pattern ["AM","GN"] against the test data

Correctness of the discovered frequent patterns can be confirmed through the use of “findSupOf” function to predict the probable area of a gene in the test data set. Figure 10 shows the confirmation of the pattern [“AM”,“GN”], which is one of the discovered frequent patterns of a class intron/exon, through the search and comparison of this pattern against the whole test set. We found that this pattern matched 41 sub-patterns in the class none, 149 sub-patterns in the class exon/intron, and 278 sub-patterns in the class intron/exon. Based on the majority matching, we thus conclude that the discovered pattern [“AM”,“GN”] correctly represents the top frequent patterns of the class intron/exon.

For completeness confirmation, we compared the patterns discovered from our approximate method with those obtained from the traditional method that does not apply the density approximation and sampling technique. With varied percentages of minimum support value, our approximate method can discover patterns very close to the traditional method. The results are summarized in table 1.

Table 1. Comparative results of number of patterns discovered from our approximate method with those discovered from traditional method.

Minimum support	Traditional pattern discovery method				Approximate method				#Matched patterns
	# 1-item	# 2-item	# 3-item	# 4-item	# 1-item	# 2-item	# 3-item	# 4-item	
Class = "none"									
50%	0	0	0	0	0	0	0	0	0
45%	0	0	0	0	0	0	0	0	0
40%	0	0	0	0	0	0	0	0	0
35%	0	0	0	0	0	0	0	0	0
30%	1	0	0	0	1	0	0	0	1
25%	117	0	0	0	111	0	0	0	111
Class = "exon/intron"									
85%	3	2	0	0	3	2	0	0	5
80%	4	5	2	0	4	5	2	0	11
75%	4	5	2	0	4	5	2	0	11
70%	4	6	3	0	4	6	3	0	13
65%	5	8	5	1	5	8	5	1	19
60%	5	9	7	2	5	8	5	1	19
Class = "intron/exon"									
85%	2	1	0	0	2	1	0	0	3
80%	3	3	1	0	3	3	1	0	7
75%	3	3	1	0	3	3	1	0	7
70%	3	3	1	0	3	3	1	0	7
65%	3	3	1	0	3	3	1	0	7
60%	3	3	1	0	3	3	1	0	7

Table 2. Averaging summary of matched patterns against the test data.

Class	Matched patterns (traditional method)		Matched patterns (approximate method)		Difference (traditional vs approximate)	
	2-item patterns	3-item patterns	2-item patterns	3-item patterns	2-item patterns	3-item patterns
none	--	--	--	--	--	--
exon/intron	91.24%	84.35%	90.98%	83.27%	0.26	1.08
intron/exon	90.11%	87.62%	90.09%	86.89%	0.02	0.73

The comparative results shown in table 1 have been performed on the training data set. When matching the discovered patterns against the DNA patterns in the test data set, we found that the difference of patterns matched by our approximate method to the ones that matched by traditional method is only 0.52% (averaging from the difference values: 0.26, 1.08, 0.02, 0.73). We therefore conclude from this empirical study that the discovery of frequent patterns from randomly selected representatives from a data stream yields the patterns as complete and accurate as the standard method that finds patterns from the whole large data set.

5. Conclusions

Frequent pattern discovery is an essential operation for association analysis. The discovery process concerns an automatic extraction of interesting patterns and correlations from a large database. These patterns can reveal implicit relationships among set of objects (or items) that lead to the generation of association rules to be used for decision support, financial forecast, medical diagnosis, and many other applications. Current studies in association rule mining concentrate on how to effectively find all objects frequently co-occurring. Given m objects, there are as much as 2^m frequent patterns to consider. Frequent pattern discovery is thus a computationally expensive problem. For the case of data streaming, this problem is even harder because a continuously generated nature of stream does not allow a revisit on each data element, but the discovery process must produce results in a reasonable short period of time.

With such a strict requirement, we therefore propose an approximate approach to tackle the frequent pattern discovery over continuous stream problem. Our approximate algorithm is intended to be a pre-processing step prior to the discovery process. We propose a stochastic method to get a good guess of the stream characteristics, and then draw a set of representatives from the incoming stream. These representatives are subsequently used in the process of frequent pattern mining. Our design had been implemented with the functional programming paradigm and the experimental results confirm the efficiency and reliability of our method. For a massive database, parallel method is a solution for the scalability problem. That is the main direction of our future research.

Acknowledgement

This research has been supported by grants from the Thailand Research Fund and the National Research Council of Thailand. The authors have been supported by Suranaree University of Technology through the funding of the Data Engineering Research Unit and the Knowledge Engineering Research Unit.

References

- Agrawal R., Aggarwal C. and Prasad V., A tree projection algorithm for generation of frequent itemsets, *Journal of Parallel and Distributed Computing*, Vol. 61, pp. 350-371 (2001).
- Agrawal R., Imielinski T. and Swami A., Mining association rules between sets of items in large databases, In *Proc. ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'93)*, pp. 207-216 (1993).
- Agrawal R. and Srikant R., Fast algorithm for mining association rules in large databases, *Research Report RJ 9839*, IBM Almaden Research Center, San Jose, CA. (1994a).
- Agrawal R. and Srikant R., Fast algorithms for mining association rules, In *Proc. 1994 Int. Conf. Very Large Data Bases (VLDB'94)*, pp. 487-499 (1994b).
- Agrawal R. and Shafer J., Parallel mining of association rules: Design, implementation, and experience, *IEEE Trans. Knowledge and Data Engineering*, Vol. 8, pp. 962-969 (1996).
- Babcock B., Babu S., Datar M., Motwani R. and Widom J., Model and issues in data stream systems, In *Proc. ACM Symp. Principles of Database Systems (PODS'02)*, pp. 1-16 (2002).
- Cai Y., Pape G., Han J., Welge M. and Auvil L., MAIDS: Mining alarming incidents from data streams, In *Proc. Int. Conf. on Management of Data*, pp. 919-920 (2004).

- Chang J. and Lee W., A sliding window method for finding recently frequent itemsets over online data streams, *Journal of Information Science and Engineering*, Vol. 20, No. 4, pp. 753-762 (2004).
- Charikar M., Chen K. and Farach-Colton M., Finding frequent items in data streams, *Theoretical Computer Science*, Vol. 312, Issue 1, pp. 3-15 (2004).
- Cheung D., Han J., Ng V., Fu A. and Fu Y., A fast distributed algorithm for mining association rules, In *Proc. 1996 Int. Conf. Parallel and Distributed Information Systems*, pp. 31-44, (1996a).
- Cheung D., Han J., Ng V. and Wong C., Maintenance of discovered association rules in large databases: An incremental updating technique, In *Proc. 1996 Int. Conf. Data Engineering (ICDE'96)*, pp. 106-114 (1996b).
- Chi Y., Wang H., Yu P. and Muntz R., Moment: Maintaining closed frequent itemsets over a stream sliding window, In *Proc. IEEE Int. Conf. on Data Mining*, pp. 59-66 (2004).
- Coenen F. and Leng P., Partitioning strategies for distributed association rule mining, *The Knowledge Engineering Review*, Vol. 21, Issue 1, pp. 25-47 (2006).
- Cuzzocrea A., Leung C. and MacKinnon R., Mining constrained frequent itemsets from distributed uncertain data, *Future Generation Computer Systems*, Vol. 37, pp. 117-126 (2014).
- Elayyadi I., Benbernou S., Ouziri M. and Younas M., A tensor-based distributed discovery of missing association rules on the cloud. *Future Generation Computer Systems*, Vol. 35, pp. 49-56 (2014).
- Gaber M., Zaslavsky A. and Krishnaswamy S., Resource-aware knowledge discovery in data streams, In *Proc. Int. Workshop on Knowledge Discovery in Data Streams*, pp. 649-656 (2004).
- Gaber M., Zaslavsky A. and Krishnaswamy S., Mining data streams: A review, *ACM SIGMOD Record*, Vol. 34, Issue 2, pp. 18-26 (2005).
- Ghoting A. and Parthasarathy S., Facilitating interactive distributed data stream processing and mining, In *Proc. IEEE Int. Symposium on Parallel and Distributed Processing Systems* (2004).
- Grahne G. and Zhu J., Efficiently using prefix-trees in mining frequent itemsets, In *Proc. ICDM'03 Int. Workshop on Frequent Itemset Mining Implementations (FIMI'03)*, pp. 123-132 (2003).
- Guha S., Koudas N. and Shim K., Data streams and histograms, In *Proc. ACM Symposium on Theory of Computing*, pp. 471-475 (2001).
- Halatchev M. and Gruenwald L., Estimating missing values in related sensor data streams, In *Proc. Int. Conf. on Management of Data*, pp. 83-94 (2005).
- Han J. and Fu Y., Discovery of multiple-level association rules from large databases, In *Proc. 1995 Int. Conf. Very Large Data Bases (VLDB'95)*, pp. 420-431 (1995).
- Han J., Pei J. and Yin Y., Mining frequent patterns without candidate generation, In *Proc. 2000 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'00)*, pp. 1-12 (2000).

- Han J., Wang J., Lu Y. and Tzvetkov P., Mining top-k frequent closed patterns without minimum support, in *Proc. Int. Conf. on Data Mining*, pp. 211-218 (2002).
- Jiang M. and Gruenwald L., Research issues in data stream association mining, *ACM SIGMOD Record*, Vol. 35, Issue 1, pp. 14-19 (2006).
- Kargupta H., Bhargava R., Liu K., Powers M., Blair P., Bushra S., Dull J., Sarkar K., Klein M., Vasa M. and Handy D., VEDAS: A mobile and distributed data stream mining system for real-time vehicle monitoring, In *Proc. SIAM Int. Conf. on Data Mining*, pp. 300-311 (2004).
- Kerdprasop K., Kerdprasop N. and Sattayatham P., Density-biased clustering based on reservoir sampling, In *Proc. 16th Int. Workshop on Database and Expert Systems Applications (DEXA)*, pp. 1122-1126 (2005).
- Kerdprasop K., Kerdprasop N. and Sattayatham P., A Monte Carlo method to data stream analysis, *Enformatika Transactions on Engineering, Computing and Technology*, Vol.14, pp. 240-245 (2006).
- Li H., Lee S. and Shan M., An efficient algorithm for mining frequent itemsets over the entire history of data streams, In *Proc. Int. Workshop on Knowledge Discovery in Data Streams* (2004).
- Lin C., Chiu D., Wu Y. and Chen A., Mining frequent itemsets from data streams with a time-sensitive sliding window, In *Proc. SIAM Int. Conf. on Data Mining* (2005).
- Lin Y., Hu X., Li X., and Wu X., Mining stable patterns in multiple correlated databases, *Decision Support Systems*, Vol. 56, pp. 202-210 (2013).
- Liu J., Pan Y., Wang K. and Han J., Mining frequent item sets by opportunistic projection, In *Proc. 2002 ACM SIGKDD Int. Conf. Knowledge Discovery in Databases (KDD'02)*, pp. 239-248 (2002).
- Mao G., Wu X., Liu C., Zhu X., Chen G., Sun Y. and Liu X., Online mining of maximal frequent item sequences from data streams, *Technical Report CS-05-07*, University of Vermont, U.S.A. (2005).
- Park J., Chen M. and Yu P., An effective hash-based algorithm for mining association rules, In *Proc. 1995 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'95)*, pp. 175-186 (1995a).
- Park J., Chen M. and Yu P., Efficient parallel mining for association rules, In *Proc. 4th Int. Conf. Information and Knowledge Management*, pp. 31-36 (1995b).
- Pei J., Han J., Mortazavi-Asl B., Pinto H., Chen Q., Dayal U. and Hsu M., PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth, In *Proc. 2001 Int. Conf. Data Engineering (ICDE'01)*, pp. 215-224 (2001).
- Savasere A., Omiecinski E. and Navathe S., An efficient algorithm for mining association rules in large databases, In *Proc. 1995 Int. Conf. Very Large Data Bases (VLDB'95)*, pp. 432-443 (1995).
- Teng W., Chen M. and Yu P., Resource-aware mining with variable granularities in data streams, In *Proc. SIAM Int. Conf. on Data Mining* (2004).

Toivonen H., Sampling large databases for association rules, In *Proc. 1996 Int. Conf. Very Large Data Bases (VLDB'96)*, pp. 134-145 (1996).

Tseng F., Kuo Y. and Huang Y., Toward boosting distributed association rule mining by data de-clustering, *Information Sciences*, Vol. 180, pp. 4263-4289 (2010).

Vitter J., Random sampling with a reservoir, *ACM Transaction on Mathematical Software*, Vol. 11, No.1, pp. 37-57 (1985).

Yu J., Chong Z., Lu H. and Zhou A., False positive or false negative: Mining frequent itemsets from high speed transactional data streams, In *Proc. Int. Conf. on Very Large Databases* (2004).

Zaki M., Parthasarathy S., Ogihara M. and Li W., Parallel algorithm for discovery of association rules, *Data Mining and Knowledge Discovery*, Vol.1, pp. 343-374 (1997).

Zhu X., Li B., Wu X., He D. and Zhang C., CLAP: collaborative pattern mining for distributed information systems, *Decision Support Systems*, Vol. 52, Issue 1, pp. 40-51 (2011).



Concurrent Data Mining and Genetic Computing Implemented with Erlang Language

Kittisak Kerdprasop and Nittaya Kerdprasop

*Data Engineering Research Unit, School of Computer Engineering,
Suranaree University of Technology, Thailand
{kerdpras, nittaya}@sut.ac.th*

Abstract

The discovery process of data mining concerns an automatic extraction of interesting patterns and correlations from a large database. These patterns can reveal implicit relationships among set of objects that lead to the generation of actionable rules to be used for financial forecast, medical diagnosis, and many other useful applications. Current studies in data mining and genetic computing concentrate on how to effectively find all objects frequently co-occurring or correlated. For a massive database, parallel method is a solution for the scalability problem. In this paper, we present the design of parallel methods to the genetic algorithms, clustering, and association mining tasks. The implementation of the proposed method is based on the concurrent functional programming paradigm using the Erlang language that handles parallelism via a message passing mechanism. We test our implementations on the synthetic data sets and the real genetic data. The results show a good runtime improvement.

Keywords: *Concurrent programming, Erlang language, Concurrent genetic algorithms, Concurrent clustering, Concurrent association mining*

1. Introduction

Concurrent computing is a form of parallel task computation. A task-parallel method is commonly used in computer programming [12, 17, 23, 28] to speedup the computational time. Currently, there are two commonly used concurrent computing methods: thread and message passing. OpenMp is a well-known thread implementation [10], whereas MPI and MapReduce [30] are examples of message passing techniques. In this paper, we study parallelization of data mining and genetic computing tasks based on the message-passing method using Erlang language [5]. This language uses concurrent functional paradigm and communicates among hundreds of active processes via simple message passing built-in functions. As an example, to create multiple processes in Erlang, we use a spawn function as follows:

```
-module(example).  
-export([start/0]).  
start() ->  
    Pid1 = spawn(fun run/0),  
    io:format("New process ~p~n", [Pid1]),  
    Pid2 = spawn(fun run/0),  
    io:format("New process ~p~n", [Pid2]).  
run() -> io:format("Hello ! ~n", []).
```

The start function in a module example, which is the main process, creates two processes with identifiers Pid1 and Pid2, respectively. The newly created processes execute a run function that prints the word “Hello !” on the screen as the following:

```
New process <0.63.0>
Hello !
New process <0.64.0>
Hello !
```

The numbers <0.63.0> and <0.64.0> are identifiers of the newly created two processes. Each process then independently invokes the run function to print out a word “Hello !” on the screen. The processes in Erlang virtual machine are lightweight and do not share memory with other processes. Therefore, it is an ideal language to implement large scale parallelizing algorithms through the concurrent computation methods.

2. Concurrent implementation methods

2.1. Concurrent genetic algorithms

Genetic algorithms are search and optimization methods inspired by the natural selection process that causes biological evolution [11]. At the initial stage, genetic algorithms model a population of individuals by encoding each individual as a string of alphabets called a chromosome. Some of these individuals are possible solutions to a problem. To find good solution quickly, the algorithms emulate the strategy of nature, that is, survival of the fittest. Individuals that are more fit, as measured by the fitness function, are more likely to be selected for reproduction and recombination to create new chromosomes representing the next generation. Reproduction and recombination are normally achieved through the probabilistic selection mechanism together with the crossover and mutation operators.

As a consequence of their simple and yet effective search procedure, genetic algorithms have been successfully applied to solve different kinds of work [1, 3, 4]. Parallel computation for genetic algorithms has been proposed [7, 18, 24, 25] for at least two decades to speedup the computational time. Our work presented in this paper propose a simple scheme toward high performance computing using message passing mechanism, instead of a more sophisticated techniques appeared in the literature. The work of Bienz, *et al.*, [6] is close to ours, but their process interaction scheme is more tightly coupled than our scheme.

The implementation of genetic algorithms uses a simple mathematical problem: find the maximum squared number of an integer from the search space of mixed positive integers ranging from 1 to 16,777,127. The correct solution is 281,472,426,579,600. Main module of our program is the function go() that takes three parameters, that is, the population size, probability of mutation, and probability of crossover. Program source code in Erlang is given as follows:

```
go(PS, PM, PC) ->
    p([max_is, max()]),
    Popu = init(PS, space()),
    evol(PS, PM, PC, Popu, maxLoop(), false).

max() -> round(math:pow(2, bit()) - 1).

bit() -> 24 .                % 24=2**24 instances including 0
```

```
maxLoop()->150.
correct()-> 0.99999.
space()-> lists:seq(1,round(math:pow(2,bit()-1) ).
init(PS,L)->
    random:seed(erlang:now()),
    Pop = randW(L,PS) ,
    lists:map(fun encode/1,Pop).
randW(_,0)-> [];      % random population with replacement
randW(L,N)->
    [lists:nth(random:uniform(length(L)),L ) | randW(L,N-1)].
evol(PS,PM,PC,Popu,0,_)->
    p([in_each_evol,hd(Popu)]),
    hd(Popu) ;
evol(PS,PM,PC,Popu,_,true)->
    p([in_each_evol2,hd(Popu)]),
    hd(Popu) ;
evol(PS,PM,PC,Popu,Max,false)->
    PopuNew = xover(PM,PC,Popu)++Popu,
    Lout = sel(PS,PopuNew),
    [{Tmp,_,_}|_] = Lout,
    Percent=Tmp/max(),
    p([after_evol_loop , maxLoop()-Max+1,max,Tmp/max()]),
    OverThresh = Percent>correct(),
    evol(PS,PM,PC,Lout,Max-1,OverThresh).
sel(PS,Popu)-> % select good parent
    Lsort=lists:sort(fun ({_,_,X},{_,_,Y})-> X>Y end, Popu),
    {L1,_}=lists:split(PS,Lsort) ,
    L1 . % select best rank
xover(PM,PC,[])> [];
xover(PM,PC,[X1,X2|T])> xv(X1,X2,maybe(PC),PM)++xover(PM,PC,T) .
xv(X1,X2,false,PM)-> [X1,X2]; % no crossover
xv(X1,X2,true,PM)-> cross(X1,X2,PM) . % crossover
cross({_,X1,_},{_,X2,_},PM)->
    Rand=random:uniform(length(X1))-1,
    {L1,L11}= lists:split(Rand,X1),
    {L2,L22}= lists:split(Rand,X2),
    Xnew1= mutString(L1++L22,PM),
    Xnew2= mutString(L2++L11,PM), % mutate
    V1 = decode(Xnew1,bit()),
    V2= decode(Xnew2,bit()),
    [{V1,Xnew1,fitness(V1)},{V2,Xnew2,fitness(V2)}].
mutString([],PM)->[];
mutString([H|T],PM)->
```



```

    Prob=maybe (PM) ,
    if Prob-> [(H+1) rem 2 |mutString(T,PM) ]; % mutate
        true -> [H |mutString(T,PM) ] % no mutate
    end.

    maybe(Prob)-> random:uniform() < Prob.

    encode(N)-> { N , bitOf(N,bit()),fitness(N) }.

    decode([],_)->0;

    decode([H|T],B)-> round(H*math:pow(2,B-1))+decode(T,B-1) .

    bitOf(_,0)->[];

    bitOf(N,B)-> bitOf(N div 2,B-1)+[N rem 2].

    fitness(A)-> A*A .

    p(L)-> lists:foreach(fun(H)->io:format("~p ",[H])end,L) ,
        io:format("~n").
    
```

On the design of concurrent computation (Figure 1), we try to keep the message communication as simple as possible. The main process simply creates the child process and waits for the first best result to arrive. As soon as the main process receives the first solution, it will kill other processes that are still active. This problem has only one best solution, so we accept the first one. Implementation of this concurrent scheme is in Figure 2.

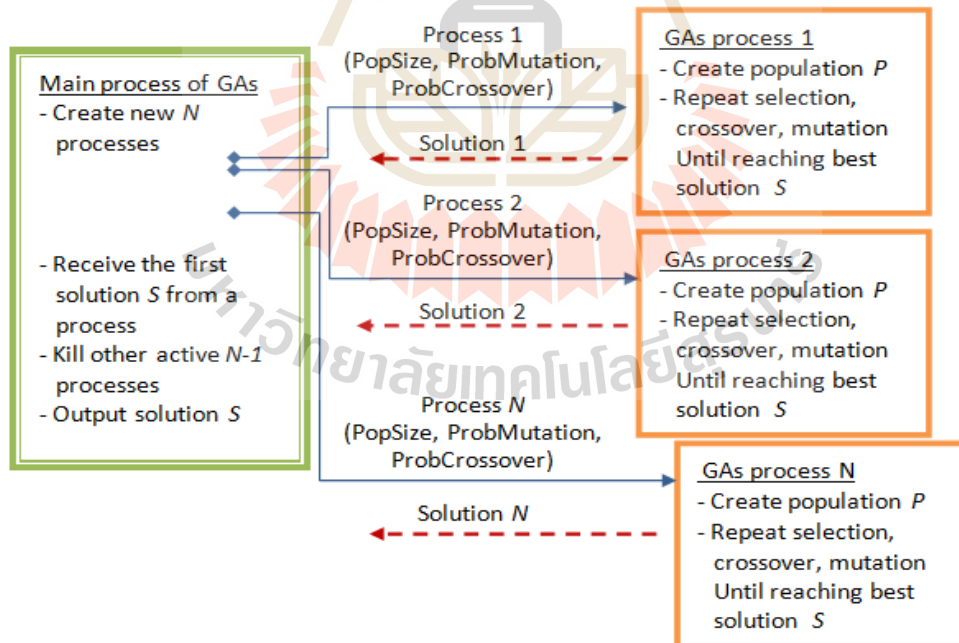
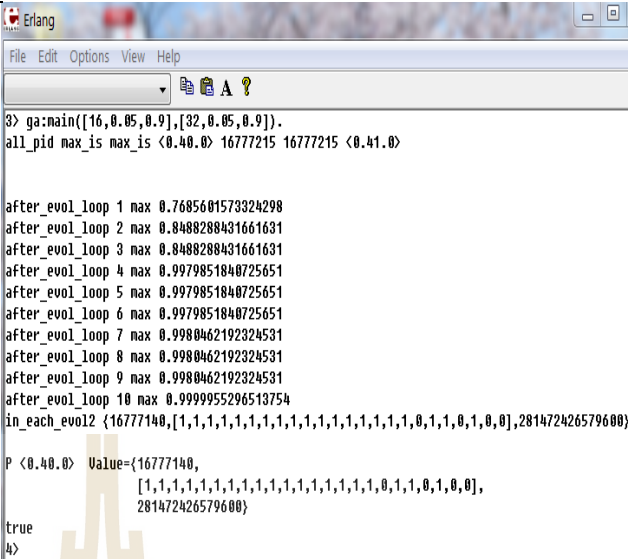


Figure 1. A message-passing model in concurrent genetic algorithms

```
-module(ga).
-compile(export_all).
main([P1,M1,C1],[P2,M2,C2]) ->
    Pid2=spawn(?MODULE, process,
               [P1,M1,C1]),
    Pid3=spawn(?MODULE, process,
               [P2,M2,C2]),
    Pid2 ! {self()},
    Pid3 ! {self()},
    p([all_pid,Pid2,Pid3]),
    receive
        {Pid,Msg}->io:format("P ~w
                             Value=~p~n", [Pid,Msg]),
        exit(Pid2,kill),
        exit(Pid3,kill)
    end.
process(PS,PM,PC) ->
    R = go(PS,PM,PC),
    receive
        {From}-> From!{self(),R}
    end.
```



```
3) ga:main([16,0.05,0.9],[32,0.05,0.9]).
all_pid max_is max_is <0.40.0> 16777215 16777215 <0.41.0>

after_evol_loop 1 max 0.7685601573324298
after_evol_loop 2 max 0.8488288431661631
after_evol_loop 3 max 0.8488288431661631
after_evol_loop 4 max 0.9979851848725651
after_evol_loop 5 max 0.9979851848725651
after_evol_loop 6 max 0.9979851848725651
after_evol_loop 7 max 0.9980462192324531
after_evol_loop 8 max 0.9980462192324531
after_evol_loop 9 max 0.9980462192324531
after_evol_loop 10 max 0.9999955296513754
in_each_evol12 {16777140,[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,0,1,0,0],281472426579600}

P <0.40.0> Value={16777140,
                 [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,0,1,0,0],
                 281472426579600}

true
4>
```

Figure 2. Implementation and running result of concurrent genetic algorithms with two active processes. The first process (process-id = <0.40.0>) has population size = 16, probability of mutation = 0.05, and probability of crossover = 0.9. The second process (process-id = <0.41.0>) has population size = 32, the other two parameters are the same as the first process

2.2. Concurrent clustering

Clustering is an unsupervised learning problem widely studied in many research areas such as statistics, machine learning, data mining, pattern recognition. The objective of clustering process is to partition a mixture of large dataset into smaller groups with a general criterion that data in the same group should be more similar or closer to each other than those in different groups. A serial k-means algorithm was proposed by J.B. MacQueen in 1967 [20] and since then it has gained much interest from data analysts. Despite its simplicity and great success, the k-means method is known to degrade when the dataset grows larger in terms of number of objects and dimensions [13, 15]. To obtain acceptable computational speed on huge datasets, most researchers turn to parallelizing scheme [9, 14, 16, 22, 27, 30].

The serial k-means algorithm [20] starts with the initialization phase of randomly selecting temporary k central points, or centroids. Then, iteratively assign data to the nearest cluster and then re-calculate the new central points of k clusters. The serial algorithm takes much computational time on calculating distances between each of N data points and the current K centroids. Then iteratively assign each data point to the closest cluster. We thus improve the computational efficiency by assigning P processes to handle the clustering task on a smaller group of N/P data points. The centroid update is responsible by the master process. In Figure 3, the PKM algorithm is the master process responsible for creating new parallel processes, sending centroids to the created processes, receiving the cluster assignment results, and recalculating the new centroids. The steps repeat as long as the old and the new centroids do not converge. The convergence criterion can be set through the function *difference(C, C')*. A screenshot of compiling and running the program (Erlang release R13B04) is given in Figure 4.

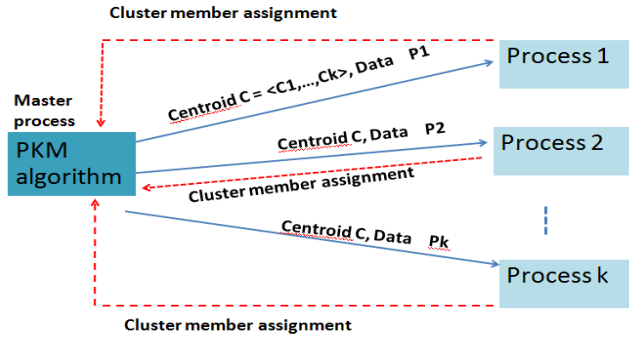


Figure 3. Communication between master and created processes in the concurrent k-means clustering

```

-module(pkm).
-import(lists,[seq/2,sum/1,flatten/1,
              split/2,nth/2]).
-import(io,[format/1,format/2]).
-import(random,[uniform/1]).

start(DataL,Cent,NumPar) ->
    CidL = myspawn(NumPar),
    LastC= myloop(CidL,Cent,
                 DataL,NumPar,1),
    format("~nCentroid=~w",[LastC]),
    LastC.

myspawn(0) -> [] ;
myspawn(N) -> [spawn(?MODULE,c,
                    [self()]) |
              myspawn(N-1) ].

myloop(CidL, Cent, DataL,
       NumPar, Count) ->
    mysend(Count,CidL,Cent,DataL),
    L=flatten(myrec(Count,NumPar)),
    C=calNewCent(Cent, L),
    format("~w.", [Count]),
    if Count >100 -> mystop(CidL),C;
        Cent/= C_ ->
            myloop(CidL,C_,DataL,
                  NumPar, Count+1);
        true -> mystop(CidL),C_
    end.

c(Sid) ->
    receive
    stop -> true;
    {LoopN,Cent,Data} ->
        L=locate(Data,Cent),
        Sid!{LoopN,L},c(Sid)
    end.
    
```

```

3> NumCent=4,CL=lists:sublist(D,NumCent).
[[{924,4436},{7231,9459},{5015,3114},{5975,9157}]
4> NumPar=8,DL=pkm:mysplit(length(D) div NumPar,D,NumPar).
[[{924,4436},
 {7231,9459},
 {5015,3114},
 {5975,9157},
 5> {TReal,RealCent}=timer:tc(pkm:start,[DL,CL,length(DL)]).
1.2.3.4.5.6.7.8.9.10.11.12.13.14.15.16.17.18.19.20.21.22.23.24.25.26.
.31.32.33.34.35.
Centroid=[[{2496.7755378358665,2510.583056046844},{7494.7329494421265,
388755},{7500.717667375798,2502.243794268552},{2493.4655902076956,750
396}][{129371000,
 [{2496.7755378358665,2510.583056046844},
 {7494.7329494421265,7500.258223388755},
 {7500.717667375798,2502.243794268552},
 {2493.4655902076956,7504.351034472396}]]
6> TReal.
129371000
    
```

Figure 4. Coding and series of line commands to create four initial centroids (command 3), then partition 800,000 data points into eight subgroups sending to the eight processors (command 4), parallel k-means (pkm) starts at command 5. The outputs of pkm are the number of iteration (i.e., 35) and the mean points of four clusters. A variable TReal is for displaying the running time of the whole process, which is 129371000 microseconds or 129.371 seconds, including send-receive messages between the master and the eight concurrent processes

2.3. Concurrent association mining applied to the splice site recognition problem

The splice site recognition problem can be formulated as the following. Given some part of unclassified genomic DNA sequences, decide whether this is an intron/exon border, an exon/intron border, or none of the two splice sites. To develop an accurate prediction model, a machine learning technique is usually applied. The learning task is that given sequences of genomic DNA with known splice junction labeled as either an intron/exon, an exon/intron, or none, the learning objective is to find a classification rule that can successfully predict the region of uncharacterized genomic DNA sequence.

Splice site prediction can be considered as a subproblem of gene prediction that aims at correctly recognizing gene from the given fragment of DNA sequence. The task of splice site prediction is to recognize the actual boundaries of the protein-coding regions in the DNA sequence. There are many computational techniques applied to tackle this problem. The direct method [8, 29] is to analyze the sequence to capture gene profile and identify specific features that can accurately predict the splice junctions. Researchers from the machine learning community prefer to attack this problem via a single or multiple classification learning algorithms [19, 21].

Our approach to solve the splice site recognition problem is different from those appeared in the literature in that our predictor is built from the association analysis technique [2], not the classification ones. The advantage of the proposed technique is that the prediction model can contain nucleotides at arbitrary position, not necessarily be the contiguous base sequences.

At the initial stage of our proposed method (named assoDNA), the training dataset with a mixture of exon/intron, intron/exon, and none of the two DNA sequence splice sites is separated into three subsets according to splice junction types. Each data subset is then processed through the same steps of frequent patterns and association analysis. Once the three data subsets are processed through the frequent pattern analysis method, the three sets of learning results (displayed as prediction rules) are finally combined and prioritized according to the confidence and support values. The proposed assoDNA method can be explained as a flow diagram shown in Figure 5.

The concurrent implementation of assoDNA is illustrated as follows:

```
-module(assoDNA_par) .
concurrent(P1, P2, P3) ->
    spawn(assoDNA_par, run, [self(),P1]),
    spawn(assoDNA_par, run, [self(),P2]),
    spawn(assoDNA_par,run,[self(),P3]),
    receive
        my_end -> ok
    end.

run(MasterID, InputL) ->
    R = main2(any, 3, InputL),
    file:delete("out.txt") ,
    AD = lists:last(R), [ADD|_] = AD,
    Rules = lists:sublist(R, length(R)-1),
    PrintRules = map(fun({D, S, Per, Class}) ->
        {to_Col3(notLast(D)),S,Per,transformBack(Class)} end,
        Rules),
    ADP=lists:map(fun(Data) -> {Data,checkRules(Data,Rules)} end, AD),
    ADPprint=map(fun({Data, V}) ->
```

```

    Predict=transformBack(V),
    {Data, [last(Data), Predict,
    mark(last(Data), Predict) ] } end, ADP),
    Predict=map(fun({F,S}) -> {to_Col3(notLast(F)),S} end, ADPprint),
    writeToFile(Predict),
    [_,Stop|_] = InputL,
    if Stop ==2 -> MasterID ! my_end ;
    true -> MasterID ! not_end
    end.
    
```

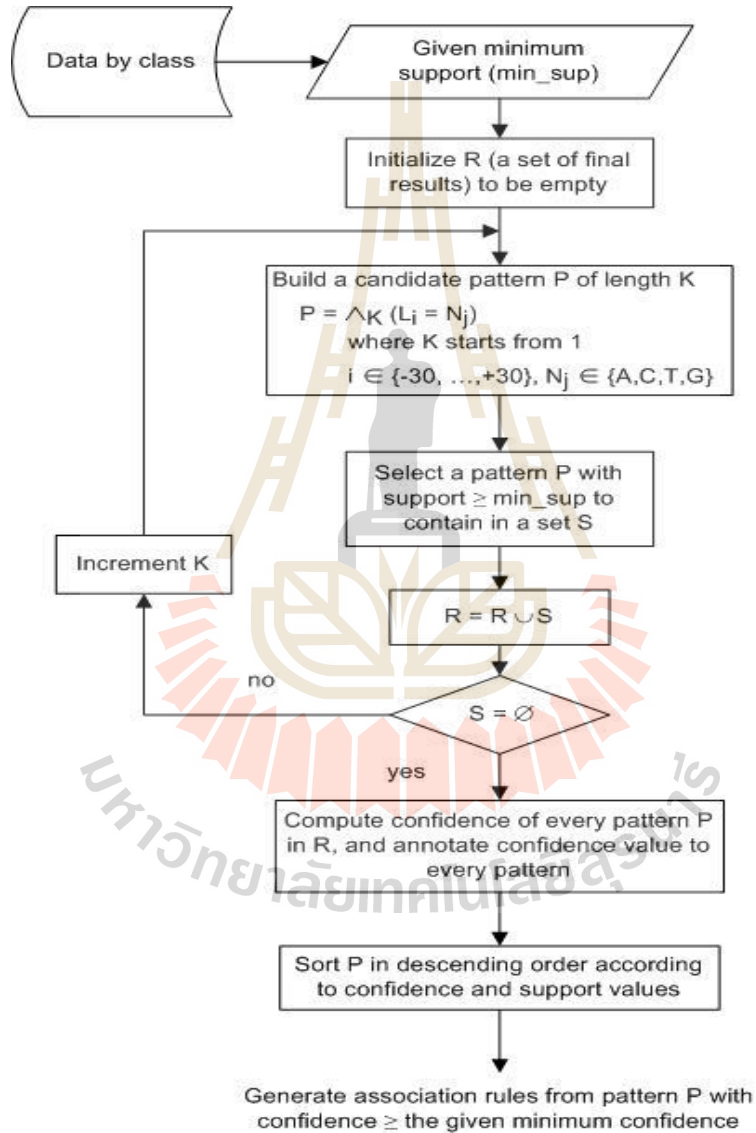


Figure 5. A flow diagram illustrating the assoDNA method

3. Performance Study Results

3.1. Performance of concurrent genetic algorithms

We design a series of experimentation to compare performance of sequential genetic algorithms against the concurrent implementation. The number of processes in the concurrent implementation has been varied from 2, 4, 8, 16, 32, 64, and 128. When the number of processes has been increased to 256, memory capacity is not enough for the Erlang system to reach the completion stage. If we, however, decrease the problem domain, the Erlang system can spawn more than hundreds of processes. To record running time of genetic algorithms, we use the following commands:

```
f(),  
T1 = erlang:now(),  
ga:main([8,0.05,0.8],[40,0.01,0.5]),  
T2 = erlang:now(),  
timer:now_diff(T2,T1)/1.0e6.
```

The $f()$ function is for clearing buffer. Function $now()$ is the clock function available in the Erlang shell. We start the concurrent process by calling function $main()$. In the above example, concurrent genetic algorithms with two processes have been invoked. The deduction of start time from the stop time will yield the running time. We also change the time unit from microsecond to second. The concurrent genetic algorithms coding can be easily adjusted to spawn more than two processes. Running time of 2 to 128 processes have been summarized and graphically shown in Figure 6. It can be seen from that concurrent genetic algorithms with 16 processes give the best computation performance. When the number of spawned processes is higher than a hundred, concurrency yields poorer performance than serial computation. This is mainly because every time the main process spawn a child process, there is an overhead cost of message passing. For this specific simple problem, we should not concurrent more than 16 processes. The optimal number of processes is however subjective and varied according to the problem domain. Empirical study is essential for the best parameter setting.

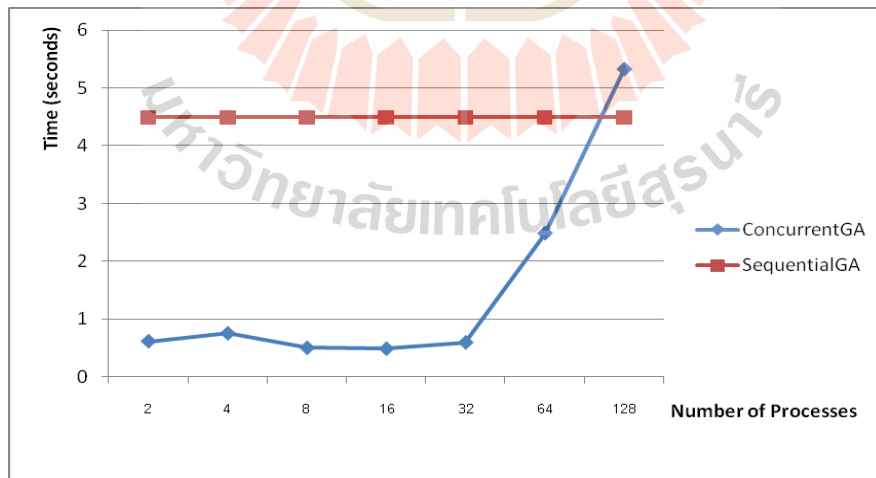


Figure 6. Computational time comparison of sequential versus concurrent genetic algorithms

3.2. Performance of concurrent clustering

We evaluate performances of the proposed PKM algorithm on synthetic two dimensional dataset. The computational speed of concurrent k-means as compared to serial k-means is given in Figure 7. It is noticeable that when dataset is small (N=50), running time of concurrent k-means is a little bit longer than the serial k-means. This is due to the overhead of spawning concurrent processes. At data size of 900,000 points, running time is unobservable because the machine is out of memory. Speedup advantage is very high (more than 30%) at dataset of size between 50,000 to 200,000 points.

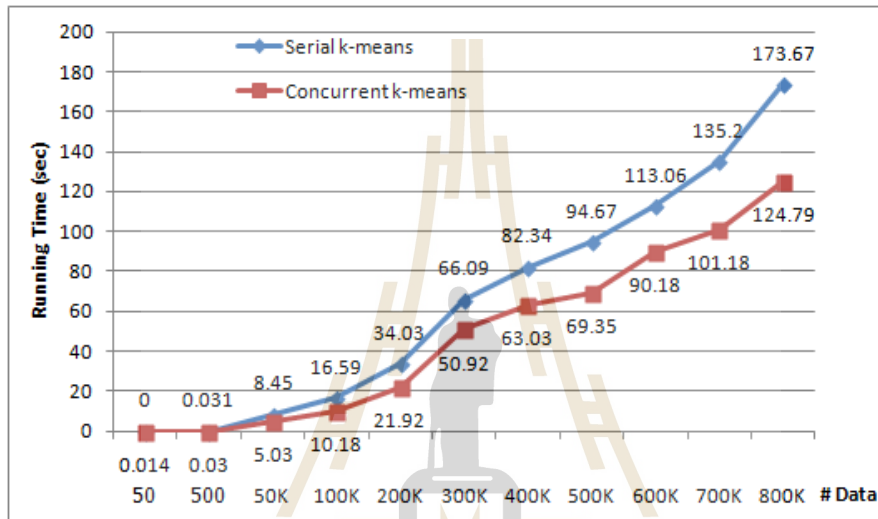


Figure 7. Running time comparisons of serial versus parallel k-means

3.3. Performance of concurrent association mining

The dataset used in this work is primate splice-junction gene sequences available at the UCI repository of machine learning databases [26]. This dataset are taken from GenBank 64.1 containing 3,190 DNA sequences. Each sequence is a window of 60 DNA base pairs starting at position -30 and ending at position +30 corresponding to the splice site location. The splice junction can be either a junction between intron and exon (intron/exon), a junction between exon and intron (exon/intron), or no junction at all (none).

To improve the computational performance of the proposed assoDNA method, we employ the concept of concurrent programming. Reduction in running time can be compared through the screenshots in Fig. 8 in which the last line on a upper screen is sequential running time (in a unit of microseconds), whereas the last line on a lower screen is concurrent running time. Time reduction in this example is around 46.29%.

```

Erlang
File Edit Options View Help
2> timer:tc(assoDNA_par.run,[self(),[1,1,80,1,2,80,1,3,80]]).

=====Read from file:"spliceDNA.DATA"=====
There are 1-3 ClassesClass ="none"
-----START---Apriori(in class=1,Min Support80%=800.0)---
[]

=====Read from file:"spliceDNA.DATA"=====
There are 1-3 ClassesClass ="exon/intron"
-----START---Apriori(in class=2,Min Support80%=400.0)---
K=2-[[["G(1)","G(5)"],427,85.39999999999999],
      [{"G(1)","T(2)"},494,98.8],
      [{"G(5)","T(2)"},424,84.8]], has 3 set

K=3-[[["G(1)","G(5)","T(2)"},424,84.8]], has 1 set

[[["TQ"],494],
 [{"GP"},499],
 [{"GT"},427],
 [{"GP","GT"},427],
 [{"GP","TQ"},494],
 [{"GT","TQ"},424],
 [{"GP","GT","TQ"},424]]

=====Read from file:"spliceDNA.DATA"=====
There are 1-3 ClassesClass ="intron/exon"
-----START---Apriori(in class=3,Min Support80%=400.0)---
K=2-[[["A(-2)","G(-1)"},496,99.2]], has 1 set

[[["AM"],497],[["GN"],498],[["AM","GN"],496]]
(9875000,not_end)

Erlang
File Edit Options View Help
3> f(),(T,_)=timer:tc(assoDNA_par.concurrent,[1,1,80],[1,2,80],[1,3,80]).

=====Read from file:"spliceDNA.DATA"=====
=====Read from file:"spliceDNA.DATA"=====
=====Read from file:"spliceDNA.DATA"=====
There are 1-3 ClassesClass ="exon/intron"
-----START---Apriori(in class=2,Min Support80%=400.0)---
There are 1-3 Classes
There are 1-3 ClassesClass ="none" Class ="intron/exon"
-----START---Apriori(in class=3,Min Support80%=400.0)---
-----START---Apriori(in class=1,Min Support80%=800.0)---
K=2-[[["G(1)","G(5)"],427,85.39999999999999],
      [{"G(1)","T(2)"},494,98.8],
      [{"G(5)","T(2)"},424,84.8]], has 3 set

K=3-[[["G(1)","G(5)","T(2)"},424,84.8]], has 1 set

[[["TQ"],494],
 [{"GP"},499],
 [{"GT"},427],
 [{"GP","GT"},427],
 [{"GP","TQ"},494],
 [{"GT","TQ"},424],
 [{"GP","GT","TQ"},424]]

K=2-[[["A(-2)","G(-1)"},496,99.2]], has 1 set

[[["AM"],497],[["GN"],498],[["AM","GN"],496]]
(5304000,ok)
    
```

Figure 8. Screenshots of sequential assoDNA (left) versus concurrent assoDNA (right)

4. Conclusion

Data mining and soft computing via genetic algorithms share a common goal of extracting patterns and useful information from a large collection of data. One important problem of such intelligent techniques is scalability due to huge amount of data to be processed. In this paper, we propose the design and implementation of concurrent computation to speedup the execution time over large amount of data. We investigate the robust search technique of genetic algorithms and propose that the algorithms can be improved via concurrency. The data mining tasks presented in this paper are clustering and association mining. Their performances also improved significantly with the concurrency scheme. From these promising results, we plan to further our study over distributed concurrent method.

Acknowledgements

This research was supported by the SUT Research and Development Fund, Suranaree University of Technology.

References

- [1] H. Adeli and N. Cheng, "Concurrent genetic algorithms for optimization of large structures", *Journal of Aerospace Engineering*, vol. 7, no. 3, (1994), pp. 276-296.
- [2] R. Agrawal, T. Imielinski and A. Swami, "Mining association rules between set of items in large databases", *Proceedings of the ACM SIGMOD International Conference on Management of Data*, (1993), pp. 207-216.
- [3] M. Al-Ansary and I. Deiab, "Concurrent optimization of design and machining tolerances using the genetic algorithms method", *International Journal of Machine Tools and Manufacture*, vol. 37, no. 12, (1997), pp. 1721-1731.
- [4] E. Alba, F. Chicano, M. Ferreira and J. Gomez-Pulido, "Finding deadlocks in large concurrent java programs using genetic algorithms", *Proceedings of the 10th International Conference on Genetic and Evolutionary Computation*, (2008), pp.1735-1742.
- [5] J. Armstrong, "Programming Erlang: Software for a Concurrent World", Raleigh, North Carolina, The Pragmatic Bookshelf, (2007).
- [6] A. Bienz, K. Fokle, Z. Keller, E. Zulkoski and S. Thede, "A generalized parallel genetic algorithm in Erlang", *Proceedings of Midstates Conference on Undergraduate Research in Computer Science and Mathematics*, (2011).
- [7] E. Cantu-Paz, "Markov chain models of parallel genetic algorithms", *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, (2000), pp. 216-226.
- [8] R. Dogan, L. Getoor and W. Wilbur, "Characterizing RNA secondary-structure features and their effects on splice-site prediction", *Proceedings of the 7th IEEE International Conference on Data Mining Workshops*, (2007), pp. 89-94.
- [9] R. Farivar, D. Rebolledo, E. Chan and R. Campbell, "A parallel implementation of k-means clustering on GPUs", *Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications*, (2008), pp. 340-345.
- [10] O. Ha and Y. Jun, "Monitoring of programs with nested parallelism using efficient thread labeling", *International Journal of u- and e-Service, Science and Technology*, vol. 4, no. 1, (2011), pp. 21-35.
- [11] J. Holland, "Adaptation in Natural and Artificial Systems", University of Michigan Press, (2004).
- [12] J. Hong, W. Sodsong, S. Chung, C. Kim, Y. Lim, S. Kim and B. Burgstaller, "Design, implementation and evaluation of a task-parallel JPEG decoder for the Libjpeg-turbo library", *International Journal of Multimedia and Ubiquitous Engineering*, vol. 7, no. 2, (2012), pp. 147-152.
- [13] M. Joshi, "Parallel k-means algorithm on distributed memory multiprocessors", Technical Report, University of Minnesota, (2003), pp. 1-12.
- [14] K. Kantabutra and A. Couch, "Parallel k-means clustering algorithm on NOWs", *NECTEC Technical Journal*, vol. 1, no. 6, (2000), pp. 243-248.
- [15] K. Kerdprasop, N. Kerdprasop and P. Sattayatham, "Weighted k-means for density-biased clustering", *Proceedings of the 7th International Conference on Data Warehousing and Knowledge Discovery*, (2005), pp. 488-497.
- [16] X. Li and Z. Fang, "Parallel clustering algorithms", *Parallel Computing*, vol. 11, no. 3, (1989), pp. 275-290.
- [17] Y. Li and L. Xiao, "Parallelization of two arithmetic operations over finite field $GF(2^n)$ ", *International Journal of Security and Its Applications*, vol. 6, no. 2, (2012), pp. 223-228.
- [18] D. Lim, Y. Ong, Y. Jin, B. Sendhoff, and B. Lee, "Efficient hierarchical parallel genetic algorithms using grid computing", *Future Generation Computer Systems*, vol. 23, (2007), pp. 658-670.
- [19] A. Lumini and L. Nanni, "Identifying splice-junction sequences by hierarchical multiclassifier", *Artificial Intelligence and Applications*, (2005), pp. 416-420.
- [20] J. MacQueen, "Some methods for classification and analysis of multivariate observations", *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, (1967), pp. 281-297.
- [21] C. Nantasenamat, T. Naenna, C. Isarankura-Na-Ayudhya and V. Prachayasittikul, "Recognition of DNA splice junction via machine learning approaches", *EXCLI Journal*, vol. 4, (2005), pp. 114-129.
- [22] A. Prasad, "Parallelization of k-means clustering algorithm", Project Report, University of Colorado, (2007), pp. 1-6.
- [23] P. Saito, R. Sabatine, D. Wolf and K. Branco, "An analysis of parallel approaches for a mobile robotic self-localization algorithm", *International Journal of Future Generation Communication and Networking*, vol. 2, no. 4, (2009), pp. 49-63.
- [24] O. Sehitoglu and G. Ucoluk, "Gene level concurrency in genetic algorithms", *Proceedings of International Symposium on Computer and Information Sciences*, (2003), pp. 976-983.

- [25] K. Tagawa, "A statistical study of concurrent differential evolution on multi-core CPUs", Proceedings of the Italian Workshop on Artificial Life and Evolutionary Computation, (2012), pp. 1-12.
- [26] The Machine Learning Database Repository, <http://mllearn.ics.uci.edu/databases/molecula-biology/splice-junction-gene-sequences>.
- [27] J. Tian, L. Zhu, S. Zhang and L. Liu, "Improvement and parallelism of k-means clustering algorithm", Tsinghua Science and Technology, vol. 10, no. 3, (2005), pp. 277-281.
- [28] H. Wu and C. Chiang, "A java prototype implementation of coordination for heterogeneous, distributed, and parallel programming", International Journal of Software Engineering and its Applications, vol. 6, no. 2, (2012), pp. 71-92.
- [29] Y. Yamada and K. Satou, "Prediction of genomic methylation status on CpG islands using DNA sequence features", WSEAS Trans on Biology and Biomedicine, vol. 5, no. 7, (2008), pp. 153-162.
- [30] W. Zhao, H. Ma and Q. He, "Parallel k-means clustering based on MapReduce", Proceedings of the 1st International Conference on Cloud Computing, (2009), pp. 674-679.

Authors



Kittisak Kerdprasop is an associate professor at the school of computer engineering, Suranaree University of Technology, Thailand. He received his bachelor degree in Mathematics from Srinakarinwirot University, Thailand, in 1986, master degree in computer science from the Prince of Songkla University, Thailand, in 1991 and doctoral degree in computer science from Nova Southeastern University, USA., in 1999. His current research includes Data mining, Artificial Intelligence, Functional and Logic Programming Languages, Computational Statistics.



Nittaya Kerdprasop is an associate professor at the school of computer engineering, Suranaree University of Technology, Thailand. She received her B.S. from Mahidol University, Thailand, in 1985, M.S. in computer science from the Prince of Songkla University, Thailand, in 1991 and Ph.D. in computer science from Nova Southeastern University, USA, in 1999. She is a member of ACM and IEEE Computer Society. Her research of interest includes Knowledge Discovery in Databases, Artificial Intelligence, Logic Programming, Deductive and Active Databases.

Building and Querying a Decision Tree Model with Constraint Logic Programming

Nittaya Kerdprasop, Fonthip Koongaew and Kittisak Kerdprasop

*Data Engineering Research Unit,
School of Computer Engineering,
Suranaree University of Technology, Thailand
nittaya@sut.ac.th*

Abstract

Decision tree induction has gained its popularity as an effective automated method for data classification mainly because of its simple, easy-to-understand, and noise-tolerant characteristics. The induced tree reveals the most informative attributes that can best characterize training data and accurately predict classes of unseen data. Despite its predictive power, the tree structure can be overly expanded or deeply grown when the training data do not show explicit patterns. Such bushy and deep trees are difficult to comprehend and interpret by humans. We thus propose a logic-based method to query over a complicate tree structure to extract only parts of the tree model that are really relevant to users' interest. The implementation using ECLiPSe constraint language to perform constrained search over a decision tree model is given in this paper. The illustrative examples on medical domains support our hypothesis regarding simplicity of constrained tree-based patterns.

Keywords: *Constraint data mining, Pattern induction, Querying, Classification tree, Decision tree induction, Constraint logic programming*

1. Introduction

A decision tree is a hierarchical structure comprising of nodes and edges. Each internal node, including the root of a tree, represents a decision choice. All possible decision choices are represented as branches from a node. The terminal decision outcome appears at the leaf node [13, 20, 24]. Machine learning and data mining communities consider the automatic process of building a decision tree from the training data with labeled decision outcomes as a classification problem (if the decision outcomes are continuous values rather than the nominal ones, it is referred to as a regression problem).

Given a training data set with decision attributes and the labeled outcome, the classification process aims at constructing an optimal classifier (or a tree) with minimum classification error. The tree-based classification process is thus composing of the tree-building phase and the pruning phase [12].

Building a decision tree from a set of training data is to partition data with mixing decision classes down the tree until each leaf node contains data instances with pure class. For a large data set with many attributes, constructed tree may contain branches that reflect chance occurrences, instead of the true relationship underlying the data subset. Many tree induction algorithms [5, 16, 18, 19] apply pruning strategies as

subsequent steps following the tree-building phase. A tree pruning operation, either pre-pruning or post-pruning, involves modifying a tree structure to be more simplified.

The built tree is considered corresponding to a collection of decision rules when traversing from the root node down to its leaves. A tree, or a set of decision rules, is normally applied as a classifier to help identifying appropriate decision on the future event or predicting class of unseen object. A classifier also serves as a generalized model of the training data set. Due to its simplicity and efficiency, decision tree induction has been applied in many research and application areas ranging from grid computing [4], finance [15], engineering [22], health care industry [1, 14], medicine [11], to bioinformatics [23].

Even with a tree pruning operation, a final tree structure can become a complex model when applying to the real world data with so many instances and attributes. General users and decision-makers normally prefer less complex decision trees. Many researchers solve this problem by simplifying tree structure with the trade off in classification accuracy [3], or applying some constraints during the tree-building phase [8, 9]. Our proposed method is different from most existing mechanism in that we deal with complexity problem after the tree induction phase.

We propose to construct a complete decision tree with the top-down induction approach [17]. Then we suggest that the users can manipulate the structure to be less complicate and truly reflect their interest by posing querying on this tree structure. Querying the tree model also appears in the literature [2, 6] but with quite a different purpose. Previous work on querying tree aims at extracting meta-data and statistical information from the model. Our work, on the other hand, focuses on serving users to extract only parts of the tree model that are of their interest.

We present the method and the detail of our implementation in Section 2. The prototype of our implementation based on the logic programming paradigm is also illustrated. Tree induction is normally implemented with SQL language [10, 21]; we demonstrate in this paper that it can be more effective to implement with constraint logic programming using ECLiPSe. Section 3 shows querying techniques. Efficiency of our implementation on medical data [7] is demonstrated in Section 4. Conclusion appears as the last section of this paper.

2. Building a Decision Tree Model with Logic Programming

We implement the decision tree induction method based on the ID3 algorithm [17] using logic programming paradigm and run with the ECLiPSe constraint programming system (<http://www.eclipseclp.org>). Program and data set are in the same format: that is, Horn clauses. Example of breast cancer data set [7] is shown in Figure 1. Format of a data set is *data([[data instances]+[attribute set]])*.

Program coding is given in Appendix. To run the program, users may simply call the predicate “run” as shown in Figure 2. The output of the program is a tree model that has been displayed as a textual format. Each branch of the tree has also been transformed as a decision rule.

```
data([ [age-"30-39", menopause-premeno, tumor_size-"30-34", inv_nodes-"0-2",node_caps-no,
deg_malig-3,breast-left,breast_quad-left_low,irradiat-no,class-no_recurrence_events],
...
[age-"40-49", menopause-premeno, tumor_size-"20-24", inv_nodes-"0-2",node_caps-no,
deg_malig-2,breast-right,breast_quad-right_up,irradiat-no,class-no_recurrence_events],
+
[ [class-no_recurrence_events, class-recurrence_events],
[age-"10-19", age-"20-29", age-"30-39", age-"40-49",age-"50-59", age-"60-69", age-
"70-79",age-"80-89",age-"90-99"],
[menopause-lt40, menopause-ge40, menopause-premeno],
[tumor_size-"0-4",tumor_size-"5-9",tumor_size-"10-14",tumor_size-"15-
19",tumor_size-"20-24",tumor_size-"25-29",tumor_size-"30-34",tumor_size-"35-
39",tumor_size-"40-44", tumor_size-"45-49",tumor_size-"50-54",tumor_size-"55-
59"],
[inv_nodes-"0-2",inv_nodes-"3-5",inv_nodes-"6-8",inv_nodes-"9-11",inv_nodes-"12-
14", inv_nodes-"15-17",inv_nodes-"18-20",inv_nodes-"21-23",inv_nodes-"24-
26",inv_nodes-"27-29",inv_nodes-"30-32",inv_nodes-"33-35",inv_nodes-"36-39"],
[node_caps-yes, node_caps-no],
[deg_malig-1, deg_malig-2, deg_malig-3],
[breast-left, breast-right],
[breast_quad-left_up, breast_quad-left_low, breast_quad-right_up, breast_quad-
right_low, breast_quad-central],
[irradiat-yes, irradiat-no]
]).
```

Figure 1. Breast Cancer Data Set in a Horn Clause Format

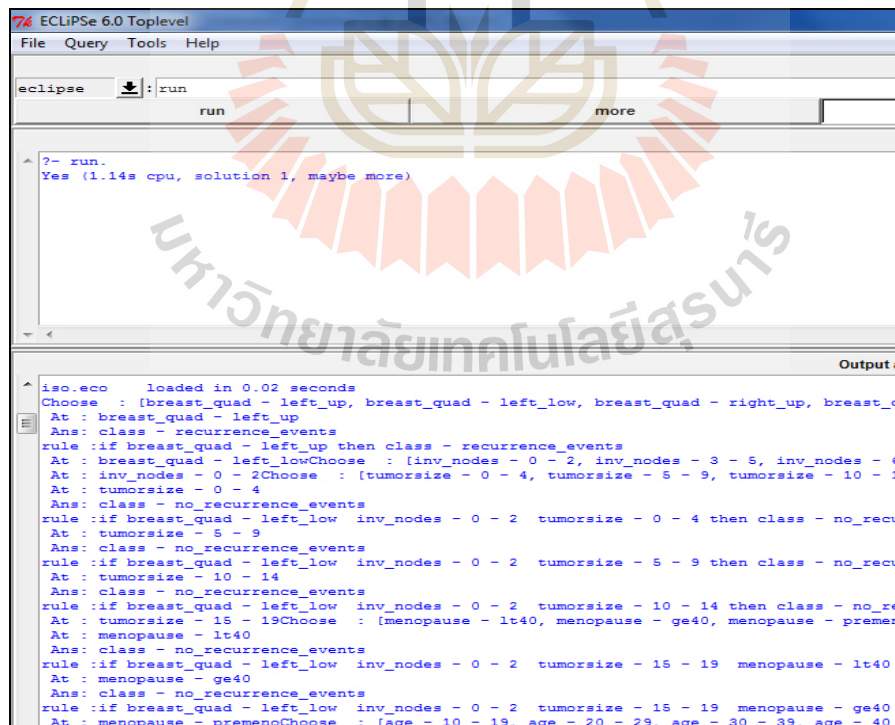


Figure 2. A Screenshot of Program Running on the Breast Cancer Data Set

3. Querying a Tree Model

Once a decision tree model has been created as shown in Figure 2, user can then query the model with 7 different styles of constraints:

findRule([]) : to display all rules extracted from a decision tree model.

findRule([X]) : to display only rules that are relevant to the attribute X (such as *irradiat* for rules that contain the attribute “irradiat” or *irradiat – yes* as a query to show all rules with “irradiat with value yes”); number of attributes is not limited.

findRule([\+X]) : to display all rules except the ones with attribute X (\+ means “not”).

findRule([X1,X2]) : to display all rules that satisfy the condition X1 AND X2 (negation \+ can also be applied to the attributes X1, X2).

findRuleOr([X1],[X2]) : to display all rules satisfied either the attribute X1, or X2 (negation \+ can also be applied to the attributes X1, X2).

findRuleOr([\+X1],[\+X2]) : to display all rules extracted from a decision tree model

findRuleOr([X1,\+X2],[\+X3,X4]) : to display all rules that satisfy the compound operations “(X1 AND (NOT X2)) OR ((NOT X3) AND X4)”.

We show the result of *findRule([])* querying over a tree model induced from the breast cancer data set in Figure 3. Then constraining the result with the query *findRule([class - recurrence_events])*. The query result is shown in Figure 4.

```
if breast_quad - left_low inv_nodes - 15-17 age - 40-49 then class -  
if breast_quad - left_low inv_nodes - 15-17 age - 50-59 then class -  
if breast_quad - left_low inv_nodes - 24-26 then class - recurrence_ev  
if breast_quad - right_up tumor_size - 0-4 then class - no_recurrence_  
if breast_quad - right_up tumor_size - 5-9 then class - no_recurrence_  
if breast_quad - right_up tumor_size - 10-14 then class - no_recurrenc  
if breast_quad - right_up tumor_size - 15-19 then class - no_recurrenc  
if breast_quad - right_up tumor_size - 20-24 inv_nodes - 0-2 then cla  
if breast_quad - right_up tumor_size - 20-24 inv_nodes - 3-5 then cla  
if breast_quad - right_up tumor_size - 25-29 deg_malig - 1 then class  
if breast_quad - right_up tumor_size - 25-29 deg_malig - 2 age - 40-  
if breast_quad - right_up tumor_size - 25-29 deg_malig - 2 age - 50-  
if breast_quad - right_up tumor_size - 25-29 deg_malig - 2 age - 50-  
if breast_quad - right_up tumor_size - 25-29 deg_malig - 2 age - 60-  
if breast_quad - right_up tumor_size - 25-29 deg_malig - 3 age - 30-  
if breast_quad - right_up tumor_size - 25-29 deg_malig - 3 age - 40-  
if breast_quad - right_up tumor_size - 25-29 deg_malig - 3 age - 50-  
if breast_quad - right_up tumor_size - 25-29 deg_malig - 3 age - 60-  
if breast_quad - right_up tumor_size - 30-34 deg_malig - 1 then class  
if breast_quad - right_up tumor_size - 30-34 deg_malig - 2 node_caps  
if breast_quad - right_up tumor_size - 30-34 deg_malig - 2 node_caps  
if breast_quad - right_up tumor_size - 30-34 deg_malig - 2 node_caps  
if breast_quad - right_up tumor_size - 30-34 deg_malig - 2 node_caps  
if breast_quad - right_up tumor_size - 30-34 deg_malig - 2 node_caps
```

Figure 3. A Set of Rules Obtained from the Query *findRule([])*

```

if breast_quad - left_low inv_nodes - 0-2 tumor_size - 30-34 age - 50-59 breast - left me
if breast_quad - left_low inv_nodes - 0-2 tumor_size - 35-39 age - 30-39 then class - recur
if breast_quad - left_low inv_nodes - 0-2 tumor_size - 35-39 age - 50-59 deg_malig - 2 the
if breast_quad - left_low inv_nodes - 0-2 tumor_size - 40-44 age - 40-49 deg_malig - 1 the
if breast_quad - left_low inv_nodes - 0-2 tumor_size - 40-44 age - 60-69 then class - recur
if breast_quad - left_low inv_nodes - 0-2 tumor_size - 50-54 breast - right then class - re
if breast_quad - left_low inv_nodes - 3-5 deg_malig - 2 age - 30-39 then class - recurrence
if breast_quad - left_low inv_nodes - 3-5 deg_malig - 2 age - 40-49 breast - left then cla
if breast_quad - left_low inv_nodes - 3-5 deg_malig - 2 age - 60-69 then class - recurrence
if breast_quad - left_low inv_nodes - 3-5 deg_malig - 3 age - 30-39 then class - recurrence
if breast_quad - left_low inv_nodes - 3-5 deg_malig - 3 age - 40-49 then class - recurrence
if breast_quad - left_low inv_nodes - 3-5 deg_malig - 3 age - 50-59 then class - recurrence
if breast_quad - left_low inv_nodes - 3-5 deg_malig - 3 age - 60-69 tumor_size - 40-44 the
if breast_quad - left_low inv_nodes - 6-8 tumor_size - 15-19 then class - recurrence_events
if breast_quad - left_low inv_nodes - 6-8 tumor_size - 25-29 then class - recurrence_events
if breast_quad - left_low inv_nodes - 6-8 tumor_size - 35-39 then class - recurrence_events
if breast_quad - left_low inv_nodes - 6-8 tumor_size - 40-44 then class - recurrence_events
if breast_quad - left_low inv_nodes - 9-11 age - 30-39 then class - recurrence_events
if breast_quad - left_low inv_nodes - 9-11 age - 70-79 then class - recurrence_events
if breast_quad - left_low inv_nodes - 15-17 age - 40-49 then class - recurrence_events
if breast_quad - left_low inv_nodes - 24-26 then class - recurrence_events
if breast_quad - right_up tumor_size - 20-24 inv_nodes - 3-5 then class - recurrence_events
if breast_quad - right_up tumor_size - 25-29 deg_malig - 2 age - 50-59 breast - left then
if breast_quad - right_up tumor_size - 25-29 deg_malig - 2 age - 60-69 then class - recur
if breast_quad - right_up tumor_size - 25-29 deg_malig - 3 age - 30-39 then class - recur
if breast_quad - right_up tumor_size - 25-29 deg_malig - 3 age - 40-49 then class - recur
if breast_quad - right_up tumor_size - 25-29 deg_malig - 3 age - 60-69 then class - recur
if breast_quad - right_up tumor_size - 30-34 deg_malig - 2 node_caps - yes age - 40-49 the
if breast_quad - right_up tumor_size - 30-34 deg_malig - 2 node_caps - yes age - 50-59 the
if breast_quad - right_up tumor_size - 30-34 deg_malig - 2 node_caps - yes age - 60-69 in
if breast_quad - right_up tumor_size - 30-34 deg_malig - 3 age - 40-49 node_caps - yes the
if breast_quad - right_up tumor_size - 30-34 deg_malig - 3 age - 50-59 then class - recur

```

Figure 4. Result of Constrained Search with the Query *findRule* ([class - recurrence_events])

4. Experimentation and Results

To test the performance of the proposed method to query over a discrete tree model, we use the standard UCI data repository [7] including the hepatitis data set (155 instances and 15 attributes), breast cancer data set (286 instances, 10 attributes), and thyroid disease data set (2800 instances and 16 attributes). Query result over a hepatitis data set with the query is shown in Figure 5. For each data set we test the system with seven different kinds of queries as summarized in the followings.

Hepatitis data set

```

findRule([])
findRule([bilirubin - "0.1-1.0"])
findRule([\+bilirubin - "0.1-1.0"])
findRule([bilirubin - "0.1-1.0", class - live])
findRule([\+bilirubin - "0.1-1.0", class - live])
findRuleOr([[ascites - yes],[bilirubin - "0.1-1.0,age - "31-40"]]) % Version 1
findRuleOr([ [ ascites - yes],[bilirubin - "0.1-1.0",age - "31-40"]]) % Version 2

```

Breast cancer data set

```

findRule([])
findRule([breast_quad - right_up])
findRule([\+breast_quad - right_up])
findRule([breast_quad - right_up, class - no_recurrence_events])
findRule([\+breast_quad - right_up, \+class - no_recurrence_events])
findRuleOr([[breast_quad - left_low, inv_nodes - "6-8" ], [breast_quad - left_low,
inv_nodes - "6-8", tumor_size - "15-19"]]) % Version 1

```

```
findRuleOr([[breast_quad - left_low , inv_nodes - "6-8" ], [breast_quad - left_low,  
inv_nodes - "6-8", tumor_size - "15-19"]]) % Version 2
```

Thyroid disease data set

```
findRule([])  
findRule([pregnant - false])  
findRule([\+pregnant - false])  
findRule([query_hyperthyroid - true , query_hypothyroid - true , sex - male ,  
on_antithyroid_medication - false])  
findRule([\+query_on_thyroxine - false , \+on_antithyroid_medication , \+class -  
sick])  
findRuleOr([[query_hyperthyroid - true , query_hypothyroid - false] , [\+  
query_hypothyroid - true] , [\+class - negative]]) % Version 1  
findRuleOr([[query_hyperthyroid - true , query_hypothyroid - false] , [\+  
query_hypothyroid - true] , [\+class - negative]]) % Version 2
```

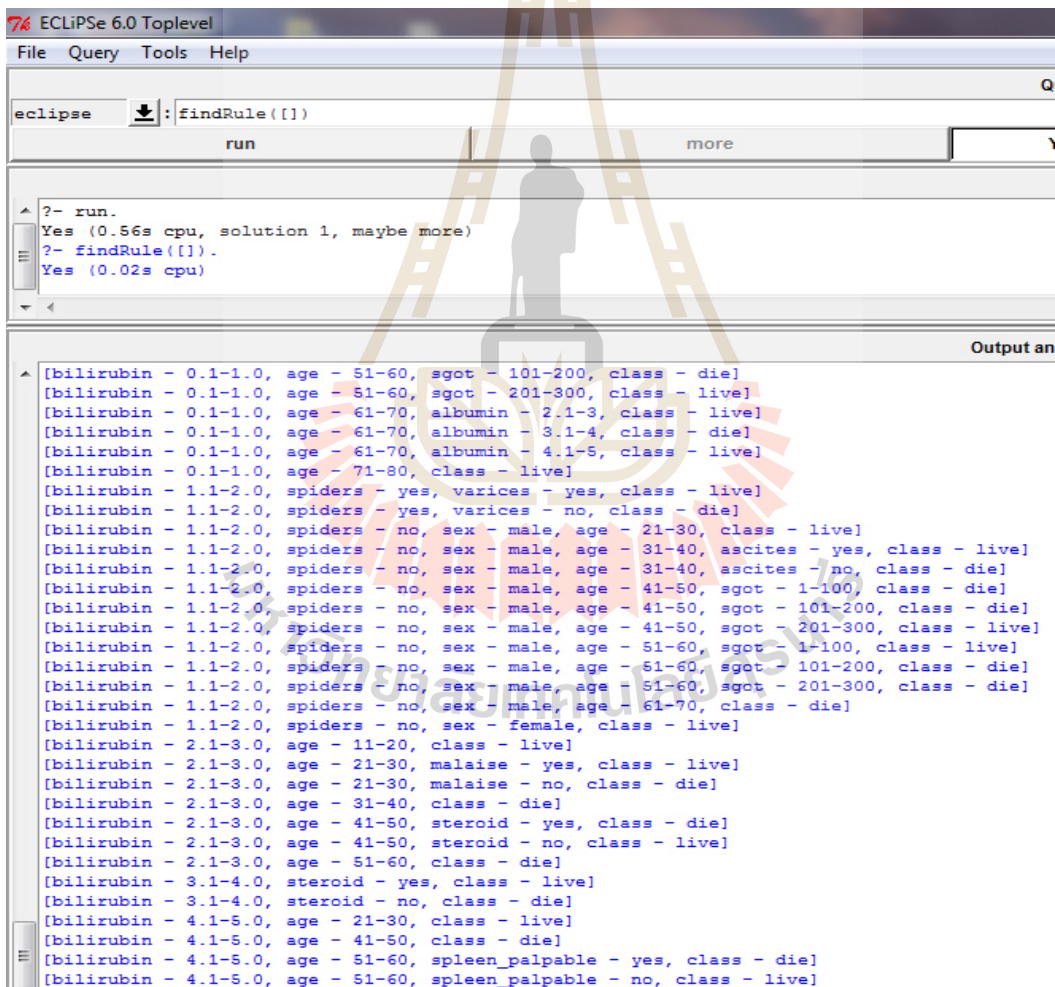


Figure 5. Running Result of Querying Hepatitis Data Model with the Query *findRule([])*

As a demonstration of querying the model, we show only one example. For the query “*findRuleOr([\+referral_source-svhc, \+ query_hypothyroid - false, class - sick], [\+ query_hypothyroid - false]])*”, its result is as follows:

- Rule 1: **if** referral_source - other sick - false tumor - false query_hypothyroid - true sex - female on_thyroxine - false query_hyperthyroid - false psych - false query_on_thyroxine - false on_antithyroid_medication - false pregnant - false thyroid_surgery - false lithium - false goitre - false hypopituitary - false **then** class - negative
- Rule 2: **if** referral_source - other sick - false tumor - false query_hypothyroid - true sex - female on_thyroxine - false query_hyperthyroid - false psych - true **then** class - negative
- Rule 3: **if** referral_source - other sick - false tumor - false query_hypothyroid - true sex - female on_thyroxine - false query_hyperthyroid - true **then** class - negative
- Rule 4: **if** referral_source - other sick - false tumor - false query_hypothyroid - true sex - female on_thyroxine - true **then** class - negative
- Rule 5: **if** referral_source - other sick - false tumor - false query_hypothyroid - true sex - male query_hyperthyroid - false on_thyroxine - false query_on_thyroxine - false on_antithyroid_medication - false pregnant - false thyroid_surgery - false lithium - false goitre - false hypopituitary - false psych - false **then** class - negative
- Rule 6: **if** referral_source - other sick - false tumor - false query_hypothyroid - true sex - male query_hyperthyroid - false on_thyroxine - true **then** class - negative
- Rule 7: **if** referral_source - other sick - false tumor - false query_hypothyroid - true sex - male query_hyperthyroid - true on_antithyroid_medication - false **then** class - sick
- Rule 8: **if** referral_source - other sick - false tumor - false query_hypothyroid - true sex - male query_hyperthyroid - true on_antithyroid_medication - true **then** class - negative
- Rule 9: **if** referral_source - other sick - false tumor - true **then** class - negative
- Rule 10: **if** referral_source - other sick - true sex - female query_hyperthyroid - false query_hypothyroid - true **then** class - negative
- Rule 11: **if** referral_source - other sick - true sex - female query_hyperthyroid - true **then** class - negative
- Rule 12: **if** referral_source - other sick - true sex - male **then** class - negative
- Rule 13: **if** referral_source - stmw **then** class - negative

- Rule 14: **if** referral_source - svhc query_hypothyroid - true on_thyroxine - false sick - false sex - female psych - false
then class - negative
- Rule 15: **if** referral_source - svhc query_hypothyroid - true on_thyroxine - false sick - false sex - female psych - true query_on_thyroxine - false on_antithyroid_medication - false pregnant - false thyroid_surgery - false query_hyperthyroid - false lithium - false goitre - false tumor - false hypopituitary - false
then class - sick
- Rule 16: **if** referral_source - svhc query_hypothyroid - true on_thyroxine - false sick - false sex - male
then class - negative
- Rule 17: **if** referral_source - svhc query_hypothyroid - true on_thyroxine - false sick - true
then class - sick
- Rule 18: **if** referral_source - svhc query_hypothyroid - true on_thyroxine - true
then class - sick
- Rule 19: **if** referral_source - svhd query_hypothyroid - true sex - female
then class - negative
- Rule 20: **if** referral_source - svhd query_hypothyroid - true sex - male
then class - sick
- Rule 21: **if** referral_source - svi query_hypothyroid - true sex - female sick - false on_thyroxine - false query_on_thyroxine - false on_antithyroid_medication - false pregnant - false thyroid_surgery - false query_hyperthyroid - false lithium - false goitre - false tumor - false hypopituitary - false psych - false
then class - negative
- Rule 22: **if** referral_source - svi query_hypothyroid - true sex - female sick - false on_thyroxine - true query_on_thyroxine - false on_antithyroid_medication - false pregnant - false thyroid_surgery - false query_hyperthyroid - false lithium - false goitre - false tumor - false hypopituitary - false psych - false
then class - sick
- Rule 23: **if** referral_source - svi query_hypothyroid - true sex - female sick - true
then class - negative
- Rule 24: **if** referral_source - svi query_hypothyroid - true sex - male sick - false on_thyroxine - false query_on_thyroxine - false on_antithyroid_medication - false pregnant - false thyroid_surgery - false query_hyperthyroid - false lithium - false goitre - false tumor - false hypopituitary - false psych - false
then class - negative
- Rule 25: **if** referral_source - svi query_hypothyroid - true sex - male sick - true
then class - sick

Performance of running results in terms of rule reduction, that is the simplification of a tree model, can be graphically shown in Figures 6-8.

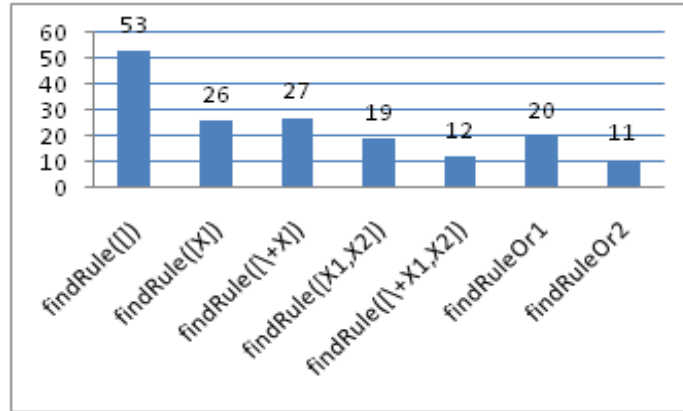


Figure 6. Performance in Terms of Model Reduction of Hepatitis Data Set

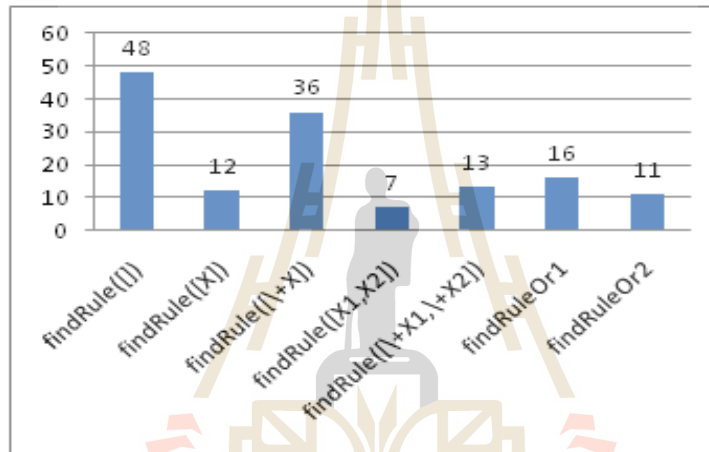


Figure 7. Performance in Terms of Model Reduction of Breast Cancer Data Set

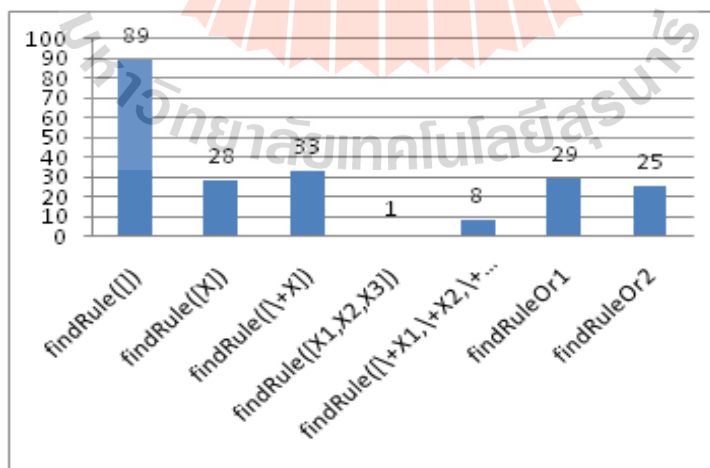


Figure 8. Performance in Terms of Model Reduction of Thyroid Disease Data Set

5. Conclusion

Classification is a data mining task that aims to induce general concept from training data. The induced concept not only explains major characteristics of the underlying data, but also acts as a classification model to predict classes of unseen data. Several learning algorithms have been proposed to induce classification concept, but the most applicable algorithm is decision tree induction. The main reason for its popularity is a simple and understandable form of a tree that has been used to represent the induced concept.

Despite its simplicity and efficiency, it could be a problem when communicating sophisticated concept as a large tree model to general users who are not an expert in decision science or computer technology. Large tree model is difficult to comprehend at a glance. Therefore, simplifying tree structure is necessary for conveying concept model to novice users. Many researchers propose a constraint-based approach during the tree-building phase to make a tree structure more simplified. We, however, consider tree simplification as a post-process of decision tree induction. We propose to grow a full decision tree. Then, apply users' preferences as a constrained search over a tree model. Only branches of a tree model that correspond to the user-specified constraints are displayed in a simple form of decision rules to the users.

The implemented prototype is expected to ease users in searching for useful knowledge from the tree model. The usability test with users who are practitioners in the field is nevertheless essential to confirm our assumption. In our future work, we also intend to consider other aspects of pushing constraints in the tree induction process.

Acknowledgements

This research was supported by the SUT Research and Development Fund, Suranaree University of Technology.

Appendix

A source code of decision tree with findRule predicates to query a tree model, implemented with constraint logic programming language ECLiPSe, is given as follows.

```
%% Program Discrete-Tree Induction with findRule queries
:- lib(listut).
:- lib(sd).
:-dynamic rule_me/1.
:-dynamic allrule/1.

append_me([H | T],L,[H | RT]) :- append_me(T,L,RT).
append_me([],L,L).

findRuleOr([H | T]) :- ( findRule(H) -> true ; ! ),
    findRuleOr(T).
findRuleOr([]).

findRule(X) :- allrule(L),
    findRule(X,L).

findRule(_,[]).
findRule(X,[H | T]) :- (findQuery(X,H)-> split_rule(H) ; true),
    findRule(X,T).

findQuery([\ +X | TX],H) :- (findAtt(X,H) -> false ; (findAttLabel(X,H) -> false ; true) ),
    findQuery(TX,H).
```

```

findQuery([X|TX],H) :- (findAtt(X,H) -> true ; (findAttLabel(X,H) -> true ; false ) ),
    findQuery(TX,H).
findQuery([],_).

findAtt(X,[X-_|_]).
findAtt(X,[_|T]) :- findAtt(X,T).
findAtt(_,[]) :- false.

findAttLabel(X,[X|_]).
findAttLabel(X,[_|T]) :- findAttLabel(X,T).
findAttLabel(_,[]) :- false.

run :- retractall(rule_me(_)),
    retractall(allrule(_)),
    compile(" / C / Users / ASUS / Desktop / id3_ok / data-sickthyroid_OK.txt"),
    data(Data+Attrs),
    main(Data,Attrs,[]),
    retractall(rule_me(_)),
    findall(X,rule_me(X),R),
    assert(allrule(R)).

main([],_).
main(_,_).
main(Data, Attrs,OldAttr) :- all_info(Data+Attrs, R1),
    (\+hasOneClass(R1) -> (maplist(avg_info,R1,Out),
        chooseMin(Out, nil/11,OO),
        OO=O/_,
        writeln('Choose ':OO),
        [listut]:delete(Attrs,O,NewAttrs),
        (foreach(X,O),param(Data,NewAttrs,OldAttr) do
            (filterData([X],Data,NewData),
            write(' At':X),
            append_me(OldAttr,[X],OAL),
            main(NewData,NewAttrs,OAL))
        ) ; writeln(""),
        getlast_goal(Data,Att-Ans),
        write(" Ans: ") , writeln(Att-Ans),
        append_me(OldAttr,[Att-Ans],NOAL),
        write('rule :'),
        split_rule(NOAL)),
        assert(rule_me(NOAL)).

split_rule(NOAL) :- mem_last(NOAL,L,RL),write('if'),
    (foreach(RLL,RL) do write(' '),write(RLL),write(' ')),
    write('then'),write(' '),writeln(L).

%last member in list
%mem_last([1,2,3,5,6],L,RL).
mem_last([H],H,[]).
mem_last([H|T],L,[H|RL]) :- mem_last(T,L,RL).

%mem(Data+Attrs),all_info(Data+Attrs,R).
all_info(Data+[P|Attr],R):-maplist(info(Data,P), Attr, R).

%mem(Data+_), info(Data,[p-y,p-n],[o-s,o-c,o-r], R).
%R=[[2,3], [4,0], [3,2]]
info(Data, P, O, O-R) :- maplist(info1(Data,P),O,R).
info1(Data, P, O, R) :- maplist(mcount(Data,O),P,R).

mcount(Data,O,P,Sum) :- foreach(L,Data),fromto(0,I,R,Sum),param(O,P) do
    ((member(O,L), member(P,L))->R is I+1; R is I).

%avg_info(o-[[2,3],[4,0],[3,2]],R).
avg_info(O-LL,O/Info):- flatten(LL,L), sumlist(L, Sum),
    (foreach(X,LL), fromto(0,I,N,Info),param(Sum) do
        (sumlist(X,SumX),

```

```

(Sum>0 -> Ratio is SumX/Sum, ! ; Ratio is 0),
logInfo(X,InfoSub),
N is I+(Ratio*InfoSub)
)).

sumlist(L,[],L):-!.
sumlist([],[],[]):-!.
sumlist([H1 | T1],[H2 | T2],[HR | TR]) :- HR is H1+H2,sumlist(T1,T2,TR).

sumlists([],R,R):-!.
sumlists([H | T],PR,NR) :- sumlist(H,PR,R), sumlists(T,R,NR).

hasOneClass([]):-!.
hasOneClass([_VL | T]) :- sumlists(VL,[],NR), hasOneClass(T,NR).

hasOneClass([],[]):-!.
hasOneClass([_VL | T],PR) :- sumlists(VL,PR,NR), hasOneClass(T,NR).
hasOneClass([],[H | T]) :- (H=0 -> hasOneClass([],T) ; find(T)).

find([]):-!.
find([H | T]) :- (H=0 -> find(T) ; false).

% get the Last data in Fist of list in list
getlast_goal([H | _],R) :- getlast(H,R).
getlast([H],H).
getlast([_ | T],R) :- getlast(T,R).

%filterData
filterData(_,[],[]):-!.
filterData(L,[H | Data],[H | R]) :- msubset(L,H,!, filterData(L,Data,R).
filterData(L,[H | Data],R) :- \+msubset(L,H,!, filterData(L,Data,R).

%-----
msubset(S1,S2) :- foreach(X,S1), param(S2) do member(X,S2), !.
allmem([H],L) :- member(H,L), !.
allmem([H | T],L) :- member(H,L), allmem(T,L).
logInfo(XL, R) :- sumlist(XL,Sum), Sum==0, R=99, !.
logInfo(XL, R) :- sumlist(XL,Sum),
    (foreach(X,XL), fromto(0,S,N,R), param(Sum) do
        ( Ratio is X/Sum,
          (Ratio>0->[iso]:log(Ratio,Log) ; Log is 1), %log(0) is undefined
          [iso]:log(2,Base2),
          N is S-(Ratio*(Log/Base2) ) ) ).

chooseMin([],O/Tmp,O/Tmp).
chooseMin([A/H | T],O/Tmp,Min) :- (H<Tmp -> NextMin = A/H ; NextMin = O/Tmp),
    chooseMin(T,NextMin,Min).

% ===== End of Program =====

```

References

- [1] M. H. M. Adnan, W. Husain and N. A. Rashid, "Hybrid approaches using decision tree, naive Bayes, means and euclidean distances for childhood obesity prediction", International Journal of Software Engineering and Its Applications, vol. 6, no. 3, (2012), pp. 99-106.
- [2] Y. Ben-Asher and I. Newman, "Decision trees with AND, OR queries", Proceedings of the 10th Annual Structure in Complexity Theory Conference, Minneapolis, Minnesota, (1995) June 19-22, pp. 74.
- [3] M. Bohanec and I. Bratko, "Trading accuracy for simplicity in decision trees", Machine Learning, vol. 15, no. 3, (1994), pp. 223-250.
- [4] A. Bouyer, M. Karimi, M. Jalali and M. N. MD SAP, "A new approach for selecting best resources nodes by using fuzzy decision tree in grid resource broker", International Journal of Grid and Distributer Computing, vol. 1, no. 1, (2008), pp. 49-62.
- [5] L. Breiman, J. Freidman, R. Olshen and C. Stone, "Classification and Regression Trees", Belmont, California: Wadsworth, (1984).

- [6] L. Fang and K. LeFevre, "Splash: ad-hoc querying of data and statistical models", Proceedings of the 13th International Conference on Extending Database Technology, Lausanne, Switzerland, (2010) March 22-26, pp. 275-286.
- [7] A. Frank and A. Asuncion, "UCI Machine Learning Repository", [http://archive.ics.uci.edu/ml], Irvine, University of California, School of Information and Computer Science, (2010).
- [8] M. Garofalakis, D. Hyun, R. Rastogi and K. Shim, "Efficient algorithms for constructing decision trees with constraints", Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Boston, MA, (2000) August 20-23, pp. 335-339.
- [9] M. Garofalakis and R. Rastogi, "Scalable data mining with model constraints", ACM SIGKDD Explorations Newsletter, vol. 2, no. 2, (2000), pp. 39-48.
- [10] A. Hinneburg, D. Habich and W. Lehner, "COMBI-operator-database support for data mining applications", Proceedings of the 29th International Conference on Very Large Data Bases, Berlin, Germany, September 9-12, (2003), pp. 429-439.
- [11] N. Kerdprasop and K. Kerdprasop, "Knowledge induction from medical databases with higher-order programming", WSEAS Transactions on Information Science & Applications, vol. 6, no. 10, (2009), pp. 1719-1728.
- [12] N. Kerdprasop and K. Kerdprasop, "The development of discrete decision tree induction for categorical data", International Journal of Mathematics and Computers in Simulations, vol. 5, no. 6, (2011), pp. 499-509.
- [13] H. Kim and G. J. Koehler, "Theory and practice of decision tree induction", Omega International Journal of Management Science, vol. 23, no. 6, (1995), pp. 637-652.
- [14] M. Ko and K.-M. Osei-Bryson, "Reexamining the impact of information technology investment on productivity using regression tree and multivariate adaptive regression splines (MARS)", Information Technology and Management, vol. 9, no. 4, (2008), pp. 285-299.
- [15] G. Nie, W. Rowe, L. Zhang, Y. Tian and Y. Shi, "Credit card churn forecasting by logistic regression and decision tree", *Expert Systems with Applications*, vol. 38, no. 12, (2011), pp. 15273-15285.
- [16] K.-M. Osei-Bryson, "Post-pruning in regression tree induction: an integrated approach", *Expert Systems with Applications*, vol. 34, no. 2, (2008), pp. 1481-1490.
- [17] J. R. Quinlan, "Induction of decision tree", *Machine Learning*, vol. 1, (1986), pp. 81-106.
- [18] J. R. Quinlan, "Simplifying decision tree", *Knowledge Acquisition for Knowledge Based Systems*, vol. 1, B. Gaines and J. Boose, Eds., Academic Press, (1989).
- [19] J. R. Quinlan, "C4.5: Programs for Machine Learning", Morgan Kaufmann, (1992).
- [20] L. Rokach and O. Maimon, "Top-down induction of decision trees classifiers-a survey", *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, vol. 35, no. 4, (2005), pp. 476-487.
- [21] K.-U. Sattler and O. Dunemann, "SQL database primitives for decision tree classifiers", Proceedings of the 10th International Conference on Information and Knowledge Management, Atlanta, Georgia, (2001) November 5-10, pp. 379-386.
- [22] S. M. Shaaban and H. Nabwey, "A decision tree approach for steam turbine-generator fault diagnosis", *International Journal of Advanced Science and Technology*, vol. 51, (2013) February, pp. 59-66.
- [23] G. Stiglic, S. Kocbek, I. Pernek and P. Kokol, "Comprehensive decision tree models in bioinformatics, PLoS ONE", vol. 7, no. 3, article e33812, doi:10.1371/journal.pone.0033812, (2012).
- [24] H. S. Yazdi and N. Salehi-Moghaddami, "Multi branch decision tree: a new splitting criterion", *International Journal of Advanced Science and Technology*, vol. 45, (2012) August, pp. 91-106.

Authors



Nittaya Kerdprasop is an associate professor at the School of Computer Engineering, Suranaree University of Technology, Thailand. She received her bachelor degree in Radiation Techniques from Mahidol University, Thailand, in 1985, master degree in Computer Science from the Prince of Songkla University, Thailand, in 1991 and doctoral degree in Computer Science from Nova Southeastern University, U.S.A, in 1999. Her research interest includes Knowledge Discovery in Databases, Artificial Intelligence, and Logic Programming.



Fonthip Koongaew is a computer engineer and research assistant. She received her bachelor degree in Computer Engineering from Suranaree University of Technology (SUT), Thailand, in 2010, and master degree in Computer Engineering from SUT in 2012. Her current research includes Constraint Data Mining, Classification Mining, and Logic Programming Languages.



Kittisak Kerdprasop is an associate professor and chair of the School of Computer Engineering, Suranaree University of Technology, Thailand. He received his bachelor degree in Mathematics from Srinakarinwirot University, Thailand, in 1986, master degree in Computer Science from the Prince of Songkla University, Thailand, in 1991 and doctoral degree in Computer Science from Nova Southeastern University, U.S.A., in 1999. His current research includes Data mining, Artificial Intelligence, Functional Programming Language, and Computational Statistics.



Knowledge Engineering Process for a Rapid Prototyping of Inductive Expert System

Nittaya Kerdprasop and Kittisak Kerdprasop

Data Engineering Research Unit, School of Computer Engineering,
Suranaree University of Technology, 111 University Avenue,
Nakhon Ratchasima 30000 Thailand

Abstract

The main characteristic of current expert systems is the separation of a knowledge base that may be changed from one application to another from the inference engine that still remains the same across applications. The delay in the development of many expert systems is due to the difficulty in acquiring and eliciting knowledge from the human domain experts. The concept of inductive expert system is thus been devised to overcome such bottleneck by incorporating automatic knowledge acquisition module in the system. According to this new concept, knowledge can now be induced or learned in an automatic way from archived databases that are normally available in most organizations. In this paper, we propose an architecture of the inductive expert system that includes the knowledge engine part to automatically forming expert rules from the stored data. We explain the automatic knowledge creation technique through a simple running example, then followed by a real application. We also provide our Prolog source code in appendices for knowledge engineers to apply our technique as a rapid prototyping of their own expert systems.

Keywords: *Expert Systems, Intelligent Knowledge Base, Machine Learning, Knowledge Engineering.*

1. Introduction

Since the release of DENDRAL in the 1960s from the Stanford Heuristic Programming Project [5] as the first practical knowledge-driven program, expert systems have enormously proliferated and been applied to all areas of computer-based problem solving. The inventors of DENDRAL system have introduced the novel and important concept of knowledge base separation in that the content of knowledge could be added and refined independently from the program module, called the inference engine, that interprets and uses that knowledge. The loosely coupling of a knowledge base and an inference engine is an influential concept to all successor rule-based expert systems such as MYCIN [10], INTERNIST-1 [6], and many others.

Since the 1980s expert systems, also called knowledge-based systems, have shifted from the medical and scientific application domains to various areas. In manufacturing and other engineering applications, rule-based expert systems are commonly applied to solve optimization problems, plan manufacturing scheduling, diagnose equipment failures, and use in almost every stage of the manufacturing process [2]. The increasing popularity of rule-based expert systems is due to the simplicity of the *if-then* rules that are easy to comprehend by humans. Many expert system tools such as Clips and Jess are available as a rule engine to facilitate rule generation for a knowledge base. These tools help facilitating knowledge representation, but knowledge acquisition and elicitation are still the labor-intensive tasks facing most knowledge engineers.

Modern expert system development process has thus moved toward the automating methodology by applying intelligent knowledge extraction techniques. Such intelligent techniques can be acquired through the machine learning and data mining technologies. There have been increasing numbers of research work attempting to apply learning techniques to automatically extract and elicit knowledge [1], [3], [4], [7], [8], [11]. These attempts have pushed the current expert system technology to the next generation of an inductive expert system in the sense that besides the knowledge base and the inference engine, the system now includes the learning component.

The research work presented in this paper takes the same direction as most researchers in an attempt to automate knowledge extraction and elicitation with machine learning and data mining techniques. Our work, however, is different from others in that not only proposing an architecture of the learnable inductive expert system and experimenting with some learning algorithms, but we also design and develop a full complement of the rule-based expert system. The work presented in this paper covers the knowledge mining from existing databases, knowledge transfer as a set of rules to be stored in the knowledge base, and knowledge reasoning through a logic-based

inference engine. Program source code for the whole process also provided in appendices.

2. A Framework for Automatic Knowledge Base Creation

We design (in Fig. 1) an architecture of the inductive expert system to include the knowledge engine facility. This part of the system requires a machine learning algorithm and a training dataset. The learning algorithm used in our work is based on the ID3 algorithm [9] because the structure of induced tree is appropriate for generating reasoning and explanation in the expert system shell. The induced knowledge is to be generated in a format of decision rules incorporated with probabilistic values. This value is intended to be used as the degree of potential applicability of each decision rule. The probabilistic values are indeed the coverage values of decision rules and can be computed as a proportion of $(\text{number of instances at leaf nodes}) / (\text{total data instances in a training dataset})$.

The steps graphically shown in Fig.2 are the process to generate decision rules to be stored in the knowledge base. These rules are to be used by the inference engine for giving recommendation to users. Consulting rules are for reasoning and giving explanation when requested by the users.

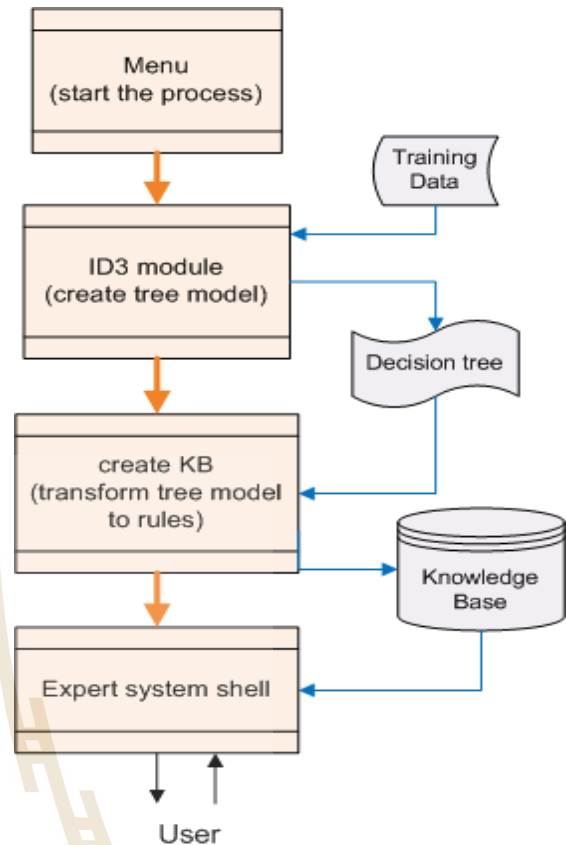


Fig. 2 Automatic knowledge engineering process.

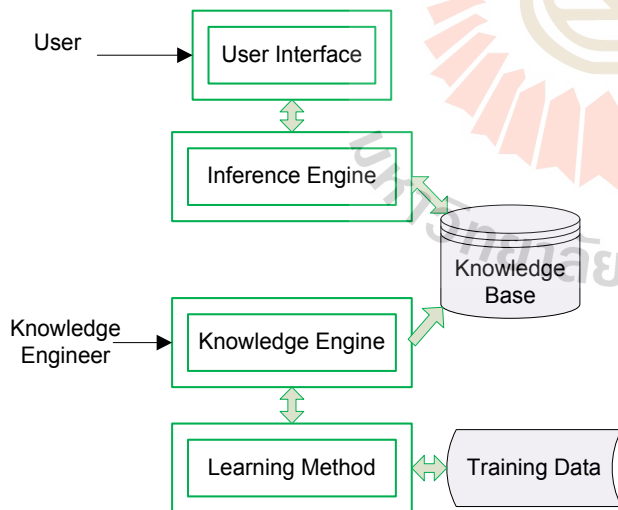


Fig. 1 Architecture of the inductive expert system.

3. Running Example

3.1 Training Data for Building a Tree Model

To explain the idea proposed in the previous section, we provide a running example through a simple training dataset as illustrated in Fig. 3. The given data contain information regarding color and shape of three objects and their classified class as either yes (the right object), or no (the wrong one). Our objective is to learn a decision model from this small dataset and extract a model in a form of a decision tree that to be helpful in identifying objects in the future with unknown class. The first step is converting data format to fit the program. Most data in the databases are represented as table. Appropriate format as required by our Prolog program is the one shown below the table in Fig.3. This converted data has been saved in a file 'shape.pl', and is to be used as a training dataset in the next step.

color	shape	class
red	round	yes
blue	polygon	yes
green	square	no

```
attribute(color, [red, green, blue]).
attribute(shape, [round, polygon, square]).
attribute(class, [yes, no]).

instance(1, class=yes, [color=red, shape=round]).
instance(2, class=yes, [color=blue, shape=polygon]).
instance(3, class=no, [color=green, shape=square]).
```

Fig. 3 A sample shape dataset that contains three instances.

3.2 Tree Model and a Transformed Knowledge Base Rule

Once the training dataset has been prepared, the next step is to build a tree model from the data. This can be done through invoking the program 'id3menu.pl.' A small dialog box will be popped up (as shown in Fig. 4) to ask the file name of training data. The parameter 'MinProb' is for pruning a tree model. The more the value, the shorter the tree model. Default value of this parameter is 0.001, which should be small enough for most moderate size data.

When user clicks the 'Enter' button, the dialog box disappears and the program starts building a tree model. This model is actually a data structure of nodes and edges (as illustrated in Fig.5). User will then be asked to input the file name to store the model. In this example, we store a model in the file named 'shape.knb'. Content of this file (displayed in Fig.6) is automatically created by the 'id3menu.pl' program. The program traverses the tree model and converts the structures of nodes and edges into rules. The created file, 'shape.knb', is a knowledge base induced from the training data and can be consulted by the inference engine of the expert system shell.

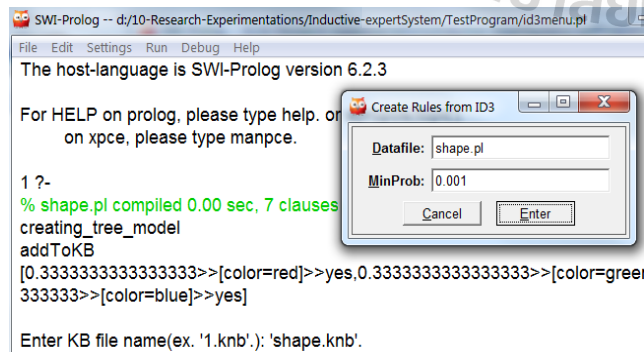


Fig. 4 A snapshot of parameter setting and output of the program id3menu.pl.

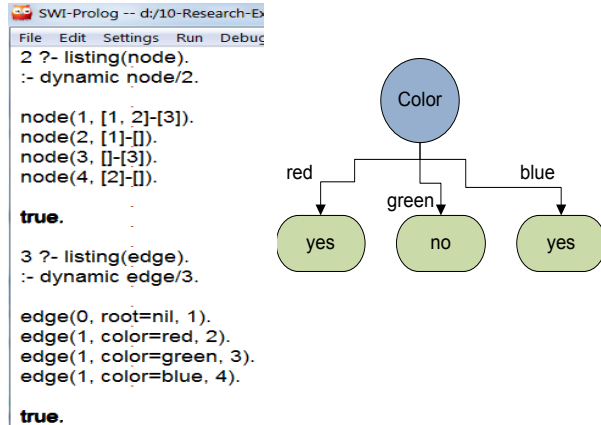


Fig. 5 A tree model in a form of node and edge structures (left) and its interpretation in a graphical form (right).

```
shape - Notepad
File Edit Format View Help
% Knowledge base automatically created for expert shell.

% top_goal is where the inference starts.

top_goal(X,V) :- type(X,V).

% Generated rules:

type(yes,0.3333333333333333):-color(red).
type(no,0.3333333333333333):-color(green).
type(yes,0.3333333333333333):-color(blue).

% Generated menu:

color(X):-menuask(color,X,[red,green,blue]).
shape(X):-menuask(shape,X,[round,polygon,square]).
class(X):-menuask(class,X,[yes,no]).

% end of automatic KB creation
```

Fig. 6 A knowledge base 'shape.knb' that is automatically generated from a tree model.

3.3 Knowledge Consulting Through the Expert System

To consult a knowledge base, user needs a second program named 'expertshell.pl'. After running this program (by double-clicking at the file name), the prompt sign '1 ?' will appear on the screen. User can now start commanding the expert system by typing 'expertshell.' and press enter. The system will greet with simple advice (as in Fig.7). This expert shell can work with any knowledge base. Therefore, user has to specify the file name of the knowledge base. It is 'shape.knb' in this example. Once the knowledge base has been loaded, user may start the consulting process by typing the command 'solve.' (Note that every command in Prolog ends with a full-stop.)

```
SWI-Prolog -- d:/10-Research-Experimentations/Inductive-expertSystem/TestProgra
File Edit Settings Run Debug Help
1 ?- expertshell.
This is the Easy Expert System shell.
Type help. load. solve. why. quit.
at the prompt.

expert-shell> load.
Enter file name in single quotes (ex. '1.knb'): 'shape.knb'.
% shape.knb compiled 0.00 sec, 8 clauses

expert-shell> solve.

What is the value for color?
[1-red,2-green,3-blue]

Enter the choice> 1.

The answer is __yes__ with probability 0.3333333333333333

expert-shell> solve.

What is the value for color?
[1-red,2-green,3-blue]

Enter the choice> 2.

The answer is __no__ with probability 0.3333333333333333
```

Fig. 7 An interaction with the expert system shell using a knowledge base 'shape.knb'.

The expert shell starts asking questions as suggested by information stored in the knowledge base. Thus, the order and content of questions can vary according to the knowledge base currently applicable to the expert shell. After the system provides appropriate answer, user may ask for explanation by typing a command 'why.'

4. Experimentation

The experimentation with real data is to confirm the efficiency of the proposed automatic knowledge base creation method. For the purpose of demonstration, we use a car evaluation data set obtain from the UCI repository (<http://archive.ics.uci.edu/ml>). In this dataset, each car is to be evaluated as acceptable or unacceptable based on the buying price, price of maintenance, number of doors, capacity in terms of persons to carry, the size of luggage boot, and the estimated safety of the car. The data set has been formatted as Prolog clauses and saved in a file named 'car.pl'. The created knowledge base is illustrated in Fig. 8, and consulting this knowledge base through the expert system shell is shown in Fig.9.

```
car - Notepad
File Edit Format View Help

% Knowledge base for expert shell. -- written by Data mining postprocess
% top_goal where the inference starts.

top_goal(X,V) :- type(X,V).

type(acc,0.32):-safety(high),persons(4). % generated rule
type(unacc,0.18):-safety(low). % generated rule
type(acc,0.14):-safety(high),persons(more). % generated rule
type(unacc,0.1):-safety(med),maint(vhigh). % generated rule
type(acc,0.08):-safety(med),maint(high). % generated rule
type(acc,0.08):-safety(med),maint(med). % generated rule
type(unacc,0.08):-safety(high),persons(2). % generated rule
type(acc,0.06):-safety(med),maint(low). % generated rule

buying(X):-menuask(buying,X,[vhigh,high,med,low]). %generated menu
maint(X):-menuask(maint,X,[vhigh,high,med,low]). %generated menu
doors(X):-menuask(doors,X,[2,3,4,'5more']). %generated menu
persons(X):-menuask(persons,X,[2,4,more]). %generated menu
lug_boot(X):-menuask(lug_boot,X,[small,med,big]). %generated menu
safety(X):-menuask(safety,X,[low,med,high]). %generated menu
class(X):-menuask(class,X,[unacc,acc]). %generated menu
```

Fig. 8 An automatically created knowledge base 'car.knb'.

```
SWI-Prolog -- d:/10-Research-Experimentations/Manufacturing-
File Edit Settings Run Debug Help
1 ?- expertshell.
This is the Easy Expert System shell.
Type help. load. solve. why. quit. or 99.
at the prompt.

expert-shell> load.
Enter file name in single quotes (ex. '1.knb'): 'car.knb'.
% car.knb compiled 0.00 sec, 5,888 bytes

expert-shell> solve.

What is the value for safety?
[1-low,2-med,3-high,99-exitShell]
Enter the choice> 2.

What is the value for maint?
[1-vhigh,2-high,3-med,4-low,99-exitShell]
Enter the choice> 4.

The answer is __acc__ with probability 0.06

expert-shell> why.

The answer is ...acc... with probability = 0.06.
The known storage are
[maint(low),safety(med)]

expert-shell> |
```

Fig. 9 Consulting 'car.knb' through the expert system shell.

5. Conclusion

Artificial intelligence, specifically expert systems, has played an important role in solving complex engineering and manufacturing problems. Knowledge base and inference procedures have been employed to solve the problems that require significant human expertise and domain-specific knowledge. The required knowledge has to be elicited by knowledge engineers. It is a labor-intensive task, and thus a bottle neck in building intelligent systems. We propose to apply data mining technique as a major step in a knowledge engine component of the inductive expert system to assist the knowledge elicitation task. The proposed technique is a novel method for automating knowledge acquisition that help supporting intelligent manufacturing systems. Knowledge in our tool can be discovered from the stored data using the decision tree induction algorithm. The learned tree structure is then transformed to a rule set that can be integrated into the knowledge base. The implementation of our knowledge acquisition tool is based on the logic programming scheme that has been proven appropriate for inferring and reasoning answers and recommendations from the existing knowledge base.

Appendix A. Source Code for Automatic Knowledge Base Creation

The source code provided here is for learning a tree model from training data and then transform the model to be a rule set to store in the knowledge base. The given ID3 module is capable of learning model of binary classes such as yes/no, true/false, acceptable/unacceptable. For training data with multiple classes, the module needs some modification. This program should be saved in a single file, named "id3menu.pl". To run the program, user may double click at the file name in the directory where it has been saved. The knowledge base will be automatically created and stored in the same directory with the file name such as 'shape.knb', and this program can now be closed. The created knowledge base will be used later by the expert system shell, which is another Prolog program.

```
/*-----id3menu.pl-----*/
id3menu:-
    new(Dialog,dialog('Create Rules from ID3')),
    send_list(Dialog, append,
        [ new(D1, text_item(datafile, '*.pl')),
          new(Per, text_item(minProb, '0.001')),
          button(cancel, message(Dialog, destroy)),
          button(enter, and(message(@prolog, callId3,
            D1?selection, Per?selection), message(Dialog, destroy) )) ),
          send(Dialog, open).
```

```
callId3(Dfile,Per) :- term_to_atom(Per1,Per),
                    consult(Dfile), createKB(Per1).

:-id3menu.

% ----- Create KB rules -----
createKB(Min) :- init(AllAttr,EdgeList), getnode(N),
                create_edge(N,AllAttr,EdgeList), addAllKnowledge,
                selectRule(Min,Res), writeln(Res), nl,
                write('Enter KB file name(ex. "1.knb"): '),
                read(F), tell(F), writeHeadF, format('~n% Generated
rules::~n'),
                maplist(createRule1,Res), nl,
                format('~n% Generated menu::~n'),
                writeTailF, told, writeln(endProcess).

writeHeadF :-
    format('% Knowledge base automatically created for expert
shell.'),
    format('~n~n% top_goal is where the inference starts::~n'),
    format('~ntop_goal(X,V) :- type(X,V)::n').

writeTailF :-
    findall(_, (attribute(S,L),
    format('~nw(X):-menuask(~w,X,~w).',[S,S,L])),_),
    format('~n~n% end of automatic KB creation').

transform1([X=V],[Res]) :-
    atomic_list_concat([X,','V,'],Res1),
    term_to_atom(Res,Res1),!.

transform1([X=V|T],[Res|T1]) :-
    atomic_list_concat([X,','V,'],Res1),
    term_to_atom(Res,Res1), transform1(T,T1).

createRule1(I) :- I = Z>>X>>Y,
    transform1(X, BodyL),
    format('~ntype(~w,~w):-',[Y,Z]),
    myformat(BodyL), !.

myformat([X]) :- write(X), write(','),!.

myformat([H|T]) :- write(H), write(','), myformat(T).

addAllKnowledge :-
    findall([A], pathFromRootToLeaf(A,_), Res),
    retractall(_>>_>>_), maplist(apply(assert),Res),
    write(addToKB), nl. % add to knowledge base
selectRule(V,Res) :-
    findall(N>>X>>Class,(X>>Class>>N,N>=V),Res1),
    sort(Res1,Res2), reverse(Res2,Res).

path(A,[H|T],C) :- edge(A,H,B), path(B,T,C).
path(C,[],C) :- !.

pathFromRootToLeaf(V>>Class>>Num, C) :-
    path(1,V,C), node(C,Value1-Value2),
    (Value1=[] ; Value2=[]),
    (Value1=[] -> length(Value2,Numb) ; length(Value1,Numb)),
    total+Total, Num is Numb/Total, hasClass(C1,C2),
    (Value1=[]->Class=C2;Class=C1).

%----- ID3 (work only with data with 2 classes) -----
:- dynamic current_node/1,node/2,edge/3,hasClass/2,type/2.
```

```
init(AllAttr,[root-nil/PB-NB]) :-
    writeln(creating_tree_model), retractall(hasClass(_,_)),
    attribute( class,[ Y1, Y2]), assert(hasClass(Y1,Y2)),
    retractall(node(_,_)), retractall(current_node(_)),
    retractall(type(_,_)), retractall(edge(_,_)),
    assert(current_node(0)), hasClass(C1,C2),
    findall(X,attribute(X,_),AllAttr1),
    delete(AllAttr1,class,AllAttr),
    findall(X2,instance(X2,class=C1,_),PB),
    findall(X3,instance(X3,class=C2,_),NB),
    length(PB,N1), length(NB,N2), N is N1+N2,
    retractall(total+_), apply(assert,[total+N]).

getnode(X) :- current_node(X), X1 is X+1,
    retractall(current_node(_)),
    assert(current_node(X1)), X1 <4000. % limit at 4000 nodes

create_edge(_,[_]) :- !.
create_edge(_,[_]) :- !.
create_edge(N, AllAttr, EdgeList) :- create_nodes(N, AllAttr,
EdgeList).
create_nodes(N, AllAttr, [H1-H2/PB-NB|T] ) :-
    getnode(N1),
    assert(edge(N,H1=H2,N1)), assert(node(N1,PB-NB)),
    append(PB, NB, AllInst),
    ( PB\==[], NB\==[] )-> (cand_node(AllAttr, AllInst, AllSplit),
        min_cand(AllSplit, [V, MinAttr, Split]),
        delete(AllAttr,MinAttr,Attr2),
        create_edge(N1,Attr2,Split)) ; true ),
    create_nodes(N,AllAttr,T).
create_nodes(_,[_]) :- !.
create_nodes(_,[_]) :- !.
min_cand([H|T], Min) :- min_cand(T, H, Min).
min_cand([], Min, Min).
min_cand([H|T], Min0, Min) :- H = [V,_], Min0 = [V0,_],
    ( V<V0 -> Min1=H ; Min1=Min0),
    min_cand(T, Min1, Min).
cand_node([H|T], CurlInstL, [[Val, H, SplitL] | OtherAttr]) :-
    info(H, CurlInstL, Val, SplitL),
    cand_node(T, CurlInstL, OtherAttr).
cand_node([],_[]) :- !.
cand_node(_,[_],[_]).
info(A,CurlInstL,R,Split) :- attribute(A,L),
    maplist(concat3(A,=), L, L1),
    suminfo(L1, CurlInstL, R, Split).
concat3(A,B,C,R) :- atom_concat(A,B,R1), atom_concat(R1,C,R).
suminfo([H|T], CurlInstL, R, [Split | ST]) :-
    AllBag = CurlInstL, hasClass(C1,C2),
    term_to_atom(H1,H),
    findall(X1,(instance(X1,_L1),member(X1,CurlInstL),
        member(H1,L1)), BagGro),
    findall(X2,(instance(X2,class=C1,L2),
        member(X2,CurlInstL), member(H1,L2)), BagPos),
    findall(X3,(instance(X3,class=C2,L3),member(X3,CurlInstL),
        member(H1,L3)), BagNeg),
```

```
(H11=H22) = H1,
length(AllBag,Nall), length(BagGro,NGro),
length(BagPos,NPos), length(BagNeg,NNeg),
Split = H11-H22/BagPos-BagNeg,
suminfo(T,CurlInstL,R1,ST),
( NPos is 0 *->L1 = 0; L1 is (log(NPos/NGro)/log(2)) ),
( 0 is NNeg *->L2 = 0; L2 is (log(NNeg/NGro)/log(2)) ),
( NGro is 0 -> R = 999;
    R is (NGro/Nall)*(-(NPos/NGro)*L1-(NNeg/NGro)*L2)+R1 ) .
suminfo([],_0,[]).
% ----- End of KB Creation Process -----
```

Appendix B. Expert System Shell in Prolog

```
% ----- expertshell.pl -----
% To run this program call 'expertshell.'
% then call 'load.' and input a file name such as 'file.knb'.
% Start consulting the expert system with the command 'solve.'
:-dynamic known/1, answer/2.
expertshell :-
    greeting, repeat, nl, write('expert-shell> '), read(X), do(X),
    X == quit, writeln('>>>>Goodbye, see you later<<<<'), !.
greeting :-
    write('This is the Easy Expert System shell. '), nl,
    native_help.
do(help) :- native_help, !.
do(load) :- load_kb, !.
do(solve) :- solve, !.
do(why) :- why, !.
do(quit).
do(X) :- write(X), write(' is not a legal command. '), nl, fail.
native_help :- write('Type help. load. solve. why. quit. '),
    nl, write('at the prompt. '), nl.
load_kb :- write('Enter file name in single quotes (ex. "1.knb".): '),
    read(F), reconsult(F).
solve :- retractall(known(_)),retractall(answer(_)),
    top_goal(X,V),
    format('The answer is __~w__ with probability ~w',[X,V]),
    assert(answer(X,V)), nl.
solve :- write('No answer found. '), nl.
menuask(Pred,Value,Menu) :-
    menuask(Pred,Menu),
    atomic_list_concat([Pred,(','),Value,']'),X),
    term_to_atom(T,X), known(T),!.
menuask(Pred,_):-
    atomic_list_concat([Pred,(','),']'),X),
    term_to_atom(T,X), known(T), !.
menuask(Attribute,Menu):-
    nl, write('What is the value for '), write(Attribute), write('?'),
    nl, addchoice(Menu,MenuRes), writeln(MenuRes), nl,
    write('Enter the choice> '), read(C), nl,
    member(C-V,MenuRes),
    atomic_list_concat([Attribute,(','),V,']'),X,
```

```
term_to_atom(T,X), asserta(known(T)).

why :- answer(A,V),
    format('~nThe answer is ...~w... with probability =
        ~w.~n',[A,V]),
    findall( X , known(X),Result),
    writeln('The known storage are'), writeln(Result).

addchoice(X,Res) :- length(X,Len),
    numlist(1,Len,NumL), map(NumL,X,Res).

map([],[],[]).
map([H|T], [X|TT], [H-X|T1]) :- map(T, TT, T1).

% ----- END OF EXPERT SYSTEM SHELL -----
```

Acknowledgment

This research was supported by the SUT Research and Development Fund, Suranaree University of Technology.

References

- [1] F. Alonso, L. Martinez, A. Perez, and J.P. Valente, "Cooperation between expert knowledge and data mining discovered knowledge: Lesson learned," *Expert Systems with Applications*, vol.39, 2012, pp.7524-7535.
- [2] A. B. Badiru, *Expert Systems: Applications in Engineering and Manufacturing*, Prentice Hall, 1992.
- [3] N. Kerdprasop and K. Kerdprasop, "Higher order programming to mine knowledge for a modern medical expert system," *International Journal of Computer Science Issues*, vol. 8, no. 3, 2011, pp. 64-72.
- [4] C. Leon, F. Biscarri, I. Monedero, J. I. Guerrero, J. Biscarri, and R. Millan, "Integrated expert system applied to the analysis of non-technical losses in power utilities," *Expert Systems with Applications*, vol.38, 2011, pp.10274-10285.
- [5] R. K. Lindsay, B. G. Buchanan, E. A. Feigenbaum, and J. Lederberg, "DENDRAL: A case study of the first expert system for scientific hypothesis formation," *Artificial Intelligence*, vol.61. no.2, 1993, pp.209-261.
- [6] R. A. Miller, H. E. Pople, and J. D. Myer, "INTERNIST-1, An experimental computer-based diagnostic consultant for general internal medicine," *New England Journal of Medicine*, vol.307, no.8, 1982, pp.468-476.
- [7] A. H. Mohammad and N. A. M. Al Saiyd, "A framework for expert knowledge acquisition," *International Journal of Computer Science and Network Security*, vol.10, no.11, 2010, pp.145-151.
- [8] R. A. Perez, L. O. Hall, S. Romaniuk, and J. T. Lilkendey, "Inductive learning for expert systems in manufacturing," *Proceedings of the 25th Hawaii International Conference on System Sciences*, 1992, pp.14-25.
- [9] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol.1, no.1, 1986, pp.81-106.
- [10] E. H. Shortliffe, *Computer-Based Medical Consultations: MYCIN*, Elsevier, 1976.
- [11] T. Witkowski, P. Antczak, and A. Antczak, "Machine learning-based classification in manufacturing system," *Proceedings of the 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, 2011, pp.580-585.

Nittaya Kerdprasop is an associate professor with the school of computer engineering, Suranaree University of Technology, Thailand. She received her B.S. from Mahidol University, Thailand, in 1985, M.S. in computer science from the Prince of Songkla University, Thailand, in 1991, and Ph.D. in computer science from Nova Southeastern University, U.S.A., in 1999. She is a member of ACM and IEEE Computer Society. Her research of interest includes Knowledge Discovery in Databases, Artificial Intelligence, Logic Programming, Deductive and Active Databases.

Kittisak Kerdprasop is an associate professor and chair of computer engineering school, Suranaree University of Technology, Thailand. He received his bachelor degree in Mathematics from Srinakarinwirot University, Thailand, in 1986, master degree in computer science from the Prince of Songkla University, Thailand, in 1991, and doctoral degree in computer science from Nova Southeastern University, U.S.A., in 1999. His current research includes Data mining, Artificial Intelligence, Functional and Logic Programming, and Computational Statistics.

Visual Data Mining and the Creation of Inductive Knowledge Base

NITTAYA KERDPRASOP and KITTISAK KERDPRASOP

Data Engineering Research Unit,
School of Computer Engineering, Suranaree University of Technology,
111 University Avenue, Nakhon Ratchasima 30000
THAILAND
nittaya@sut.ac.th, kittisakThailand@gmail.com

Abstract: - Visual data mining is a sub-discipline of data mining, which is a main analysis step of the process called knowledge discovery in databases or KDD. The objective of KDD is to automatically discover potentially useful knowledge that are hidden in large volume. In this paper, we demonstrate the use of KNIME system to do visual data mining. We also show the subsequent deployment of discovered knowledge as an intelligent resource of a rule-based expert system. Such knowledge is automatically derived from the stored data, rather than manually elicited from the human experts as in traditional expert systems. We therefore call such knowledge resource as the inductive knowledge base. The case study to derive car evaluation knowledge shown in this paper as a running example is expected to be a demonstration of data mining technique and application that can support the advancement of intelligent mechanical analysis.

Key-Words: - Visual data mining, Inductive learning, Knowledge base creation, Inductive expert system.

1 Introduction

Visual data mining is an automatic and intelligent data analysis technique that utilizes visualization as a means to communicate between user and the computer to explore data and to extract hidden patterns from stored databases [1], [2]. The main benefit of visualization is that it allows easy understanding for novice users and it is also natural to human perception. Recent trend in intelligent manufacturing and other engineering fields [3], [4] is to apply data mining techniques to automatically identify patterns and causal relationships that are too obscure and unobvious to be detected by human's eyes.

Applying data mining technique to high dimensional and large amount data is however not a straightforward task because the induced patterns are normally low accurate if the input data are not well prepared or not in an appropriate form. Numerous available learning algorithms and many data preparation techniques supported by most data mining systems are also a hindrance to users who are unfamiliar with the knowledge discovery process.

We thus illustrate in this paper a natural way to do data mining through visualization. We also propose a semi-automatic technique to transfer the data mining output to be a knowledge base content in the inductive expert system.

The main characteristic of current expert systems is the separation of a knowledge base that may be changed from one application to another from the inference engine that still remains the same across applications. The delay in the development of many expert systems is due to the difficulty in acquiring and eliciting knowledge from the human domain experts.

The concept of inductive expert system is thus been devised to overcome such bottleneck by incorporating automatic knowledge acquisition module in the system. According to this new concept, knowledge can now be induced or learned in an automatic way from archived databases that are normally available in most organizations. In this paper, we propose an architecture of the inductive expert system that includes the knowledge engine part to automatically forming expert rules from the stored data.

2 Related Work

Since the release of DENDRAL in the 1960s from the Stanford Heuristic Programming Project [5] as the first practical knowledge-driven program, expert systems have enormously proliferated and been applied to all areas of computer-based problem solving. The inventors of DENDRAL system have introduced the novel and important concept of knowledge base separation in that the content of

knowledge could be added and refined independently from the program module. This module is called the inference engine responsible for interpreting and using the knowledge. The loosely coupling of a knowledge base and an inference engine is an influential concept to all successor rule-based expert systems such as MYCIN [6], INTERNIST-1 [7], and many others.

Since the 1980s expert systems, also called knowledge-based systems, have shifted from the medical and scientific application domains to various areas. In manufacturing, mechanical analysis, and other engineering applications, rule-based expert systems are commonly applied to solve optimization problems, diagnose equipment failures, plan manufacturing scheduling, and other stages of the manufacturing process [8].

The increasing popularity of rule-based expert systems is due to the simplicity of the *if-then* rules that are easy to comprehend by humans. Many expert system tools such as Clips and Jess are available as a rule engine to facilitate rule generation for a knowledge base. These tools help facilitating the part of knowledge representation, but knowledge acquisition and elicitation are still the labor-intensive tasks for most knowledge engineers.

Modern expert system development process has thus moved toward the automating methodology by applying intelligent knowledge extraction techniques. Such intelligent techniques can be acquired through the machine learning and data mining technologies. There have been increasing numbers of research work attempting to apply learning techniques to automatically extract and elicit knowledge [9], [10], [11], [12]. These attempts have pushed the current expert system technology to the next generation of an inductive expert system in the sense that besides the knowledge base and the inference engine, the system now includes the learning component.

The research work presented in this paper takes the same direction as most researchers in an attempt to automate knowledge extraction and elicitation with machine learning and data mining techniques. Our work, however, is different from others in that not only proposing an architecture of the learnable inductive expert system, but we also cover the knowledge mining from existing databases, knowledge transfer as a set of rules to be stored in the knowledge base, and knowledge reasoning through a logic-based inference engine.

3 A Framework Toward Automatic Knowledge Base Creation

We design (in Figure 1) an architecture of the inductive expert system to include the knowledge engine facility. This part of the system requires a machine learning algorithm and a training dataset. The learning algorithm used in our work is the ID3 algorithm [13] because of its efficiency. Moreover, the structure of the induced tree is appropriate for generating reasoning and explanation part in the expert system shell.

The induced knowledge as a decision tree is subsequently to be transformed into a format of decision rules that are also incorporated each rule with the probabilistic value. This value is intended to be used as the degree of potential applicability of each decision rule. The probabilistic values are indeed the coverage values of decision rules and can be computed as a proportion of (*number of instances at leaf nodes*) / (*total data instances in a training dataset*).

The steps graphically shown in Figure 2 are the process to generate decision rules to be stored in the knowledge base. These rules are to be used by the inference engine for giving recommendation to users. Consulting rules are for reasoning and giving explanation when requested by the users.

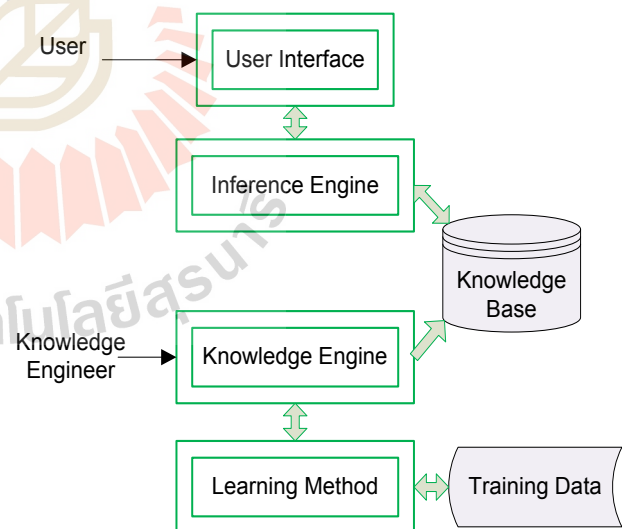


Figure 1. Architecture of the inductive expert system.

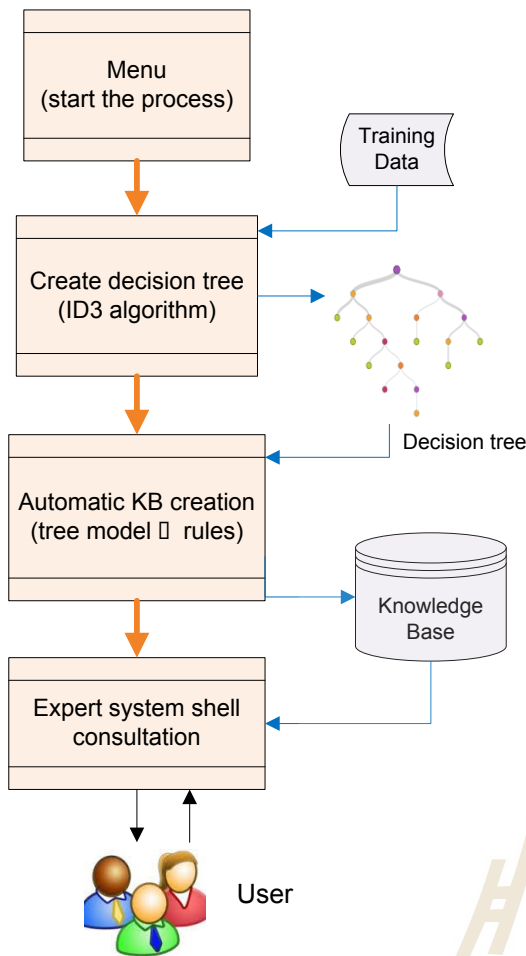


Figure 2. Automatic knowledge engineering process.

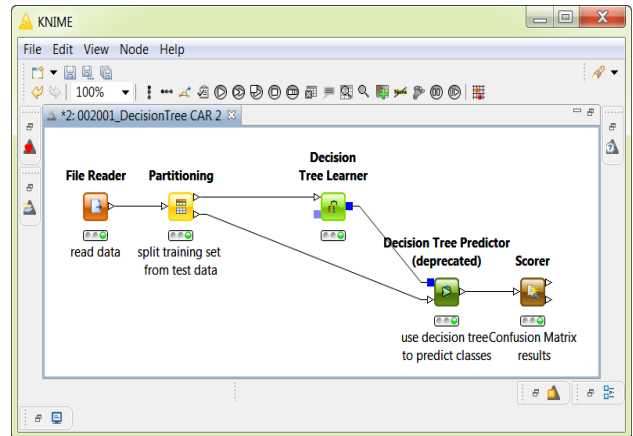


Figure 3. Data mining process through the connection of visual icons in KNIME system.

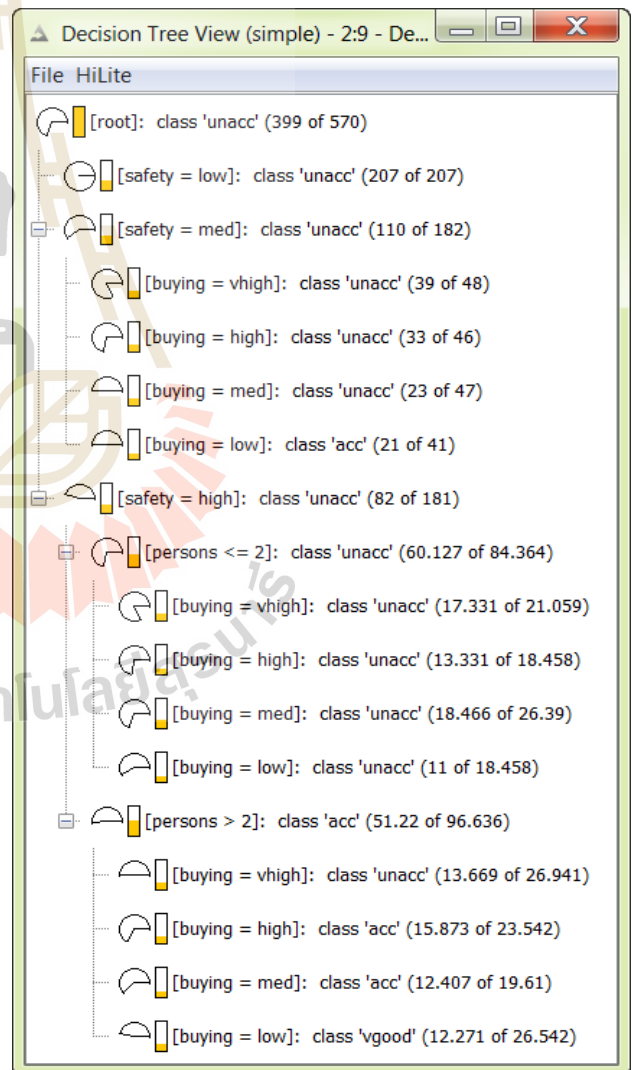


Figure 4. A decision tree model to classify safety of a car as unacc/acc/vgood.

4 Experimentation and Results

4.1 Car Evaluation Data Set and the Visual Data Mining

The purpose of this experimentation is to illustrate the proposed automatic knowledge base creation method with real data. We use a car evaluation data set obtain from the UCI repository (<https://archive.ics.uci.edu/ml/datasets/Car+Evaluation>). In this data set, each car is to be evaluated as either very good (vgood), acceptable (acc), or unacceptable (unacc) based on the buying price, price of maintenance, number of doors, capacity in terms of persons to carry, the size of luggage boot, and the estimated safety of the car.

The data set has been mined with the visual data mining tool named KNIME [14]. The visual data mining process is illustrated in Figure 3, and the mining result as decision tree is shown in Figure 4.

4.2 Knowledge Base Creation and Consultation Through the Expert System Shell

Once the decision tree has been built and output by the KNIME tool, we develop the program to traverse the tree model and convert the structures of nodes and edges into rules. The created file, 'car.knb', is a knowledge base induced from the training data and can be consulted by the inference engine of our simple expert system shell. The created knowledge base content is illustrated in Figure 5, and example steps of consulting this knowledge base through the expert system shell is shown in Figure 6.

User starts commanding the expert system by typing 'expertshell' and press enter. The system will greet with simple advice. This expert shell is implemented with the Prolog language and can work with any knowledge base. Therefore, user has to specify the file name of the knowledge base. It is 'car.knb' in this example. Once the knowledge base has been loaded, user may start the consulting process by typing the command 'solve'.

```

car - Notepad
File Edit Format View Help
% Knowledge base for expert shell. -- written by Data mining postprocess
% top_goal where the inference starts.

top_goal(X,V) :- type(X,V).

type(acc,0.32):-safety(high),persons(4). % generated rule
type(unacc,0.18):-safety(low). % generated rule
type(acc,0.14):-safety(high),persons(more). % generated rule
type(unacc,0.1):-safety(med),maint(vhigh). % generated rule
type(acc,0.08):-safety(med),maint(high). % generated rule
type(acc,0.08):-safety(med),maint(med). % generated rule
type(unacc,0.08):-safety(high),persons(2). % generated rule
type(acc,0.06):-safety(med),maint(low). % generated rule

buying(X):-menuask(buying,X,[vhigh,high,med,low]). %generated menu
maint(X):-menuask(maint,X,[vhigh,high,med,low]). %generated menu
doors(X):-menuask(doors,X,[2,3,4,'5more']). %generated menu
persons(X):-menuask(persons,X,[2,4,more]). %generated menu
lug_boot(X):-menuask(lug_boot,X,[small,med,big]). %generated menu
safety(X):-menuask(safety,X,[low,med,high]). %generated menu
class(X):-menuask(class,X,[unacc,acc]). %generated menu
    
```

Figure 5. An automatically created knowledge base 'car.knb'.

```

SWI-Prolog -- d:/10-Research-Experimentations/Manufacturing-
File Edit Settings Run Debug Help
1 ?- expertshell.
This is the Easy Expert System shell.
Type help. load. solve. why. quit. or 99.
at the prompt.
expert-shell> load.
Enter file name in single quotes (ex. '1.knb'): 'car.knb'.
% car.knb compiled 0.00 sec, 5,888 bytes
expert-shell> solve.

What is the value for safety?
[1-low,2-med,3-high,99-exitShell]
Enter the choice> 2.

What is the value for maint?
[1-vhigh,2-high,3-med,4-low,99-exitShell]
Enter the choice> 4.
The answer is __acc__ with probability 0.06
expert-shell> why.

The answer is ...acc... with probability = 0.06.
The known storage are
[maint(low),safety(med)]
expert-shell> |
    
```

Figure 6. Consulting 'car.knb' through the Prolog expert system shell.

To consult a knowledge base, user needs to invoke a program named 'expertshell.pl'. After running this program, the prompt sign '1 ?' will appear on the screen (as shown in Figure 6). User can now start commanding the expert system by typing 'expertshell.' and press enter. After that the prompt sign will be changed to 'expert-shell>'.

This expert shell is a general tool in the sense that it can work with any knowledge base that is encoded with the appropriate rule-based format. Therefore, user has to specify the file name of the knowledge base of his/her intention. It is 'cart.knb' in this example. Once the knowledge base has been loaded, user may start the consulting process by typing the command 'solve.' (Note that every command in Prolog has to end with a full-stop.)

The expert shell starts asking questions as suggested by information stored in the knowledge base. Thus, the order and content of questions can vary according to the knowledge base currently applicable to the expert shell and also depending on the answer given by user. After the system provides appropriate answer or suggestion, user may ask for explanation by typing a command 'why.' The system will respond with the phrase 'The known storage are' and then followed by the answers stored in the working storage.

The coding of expert system shell is so concise that it can be listed at the end of this section. To test this simple program, interested reader just create a knowledge base 'car.knb' as shown in Figure 5 and then run this expert system shell with the Prolog interpreter. (We use SWI-Prolog in this example.)

```
% ----- expertshell.pl -----
% To run this program call 'expertshell.'
% then call 'load.' and input a file name such as 'file.knb'.
% Start consulting the expert system with the command 'solve.'
```

```
:-dynamic known/1, answer/2.
```

```
expertshell :-
```

```
    greeting,
    repeat,
    nl, write('expert-shell> '),
    read(X),
    do(X),
    X == quit,
    writeln('>>>Goodbye, see you later<<<<'), !.
```

```
greeting :-
```

```
    write('This is the Easy Expert System shell.'), nl,
    native_help.
```

```
do(help) :- native_help, !.
```

```
do(load) :- load_kb, !.
```

```
do(solve) :- solve, !.
```

```
do(why) :- why, !.
```

```
do(quit).
```

```
do(X) :- write(X), write(' is not a legal command.'), nl, fail.
```

```
native_help :- write('Type help. load. solve. why. quit.'),
    nl, write('at the prompt.'), nl.
```

```
load_kb :- write('Enter file name in single quotes (ex. "1.knb").: '),
    read(F), consult(F).
```

```
solve :-
```

```
    retractall(known(_)),
    retractall(answer(_,_)),
    top_goal(X,V),
    format('The answer is __~w__ with probability ~w',[X,V]),
    assert(answer(X,V)), nl.
```

```
solve :- write('No answer found.'), nl.
```

```
menuask(Pred,Value,Menu) :-
```

```
    menuask(Pred,Menu),
    atomic_list_concat([Pred,'(',Value,')'],X),
    term_to_atom(T,X), known(T),!.
```

```
menuask(Pred,_):-
```

```
    atomic_list_concat([Pred,'(',')'],X),
    term_to_atom(T,X), known(T), !.
```

```
menuask(Attribute,Menu):-
```

```
    nl, write('What is the value for '),
    write(Attribute), write('?'), nl,
    addchoice(Menu,MenuRes),
    writeln(MenuRes), nl,
    write('Enter the choice> '), read(C), nl,
    member(C-V,MenuRes),
    atomic_list_concat([Attribute,'(',V,')'],X),
    term_to_atom(T,X),
    asserta(known(T)) .
```

```
why :- answer(A,V),
```

```
    format('~nThe answer is ...~w... with probability =
    ~w.~n',[A,V]),
    findall(X, known(X),Result),
    writeln('The known storage are'), writeln(Result).
```

```
addchoice(X,Res) :-
```

```
    length(X,Len),
    numlist(1,Len,NumL),
    map(NumL,X,Res).
```

```
map([],[],[]).
```

```
map([H|T], [X|TT], [H-X|T1]) :- map(T, TT, T1).
```

```
% ----- END OF EXPERT SYSTEM SHELL -----
```

5 Conclusion

Artificial intelligence, specifically expert systems, has played an important role in solving complex engineering and manufacturing problems for more than four decades. Knowledge base and inference procedures have been employed to solve the problems that require significant human expertise and domain-specific knowledge. The required knowledge has to be elicited by knowledge engineers. It is a labor-intensive task, and thus a bottle neck in building intelligent systems.

We propose to apply data mining technique as a major step in a knowledge engine component of the inductive expert system to assist the knowledge elicitation task. The proposed technique is a novel method for automating knowledge acquisition that help supporting intelligent manufacturing systems. We demonstrate knowledge mining through the visual tool called KNIME, which has many visualization features to support users who are not an expert in data mining.

Knowledge as a learned tree structure is then transformed by our Prolog program to be a rule set that can be integrated into the knowledge base. The implementation of our knowledge acquisition tool is based on the logic programming scheme that has been proven appropriate for inferring and reasoning

answers and recommendations from the existing knowledge base.

Acknowledgment

This work has been supported by grants from the National Research Council of Thailand (NRCT) and Suranaree University of Technology via the funding of Data Engineering Research Unit.

References:

- [1] S. Simoff, M. Bohlen and A. Mazeika (Eds.), *Visual Data Mining: Theory, Techniques and Tools for Visual Analytics*, Springer, 2008.
- [2] D. Keim, Information visualization and visual data mining, *IEEE Transactions on Visualization and Computer Graphics*, 7, 1, 2002, pp.100-107.
- [3] N. Ruschin-Rimini, O. Maimon and R. Romano, Visual analysis of quality-related manufacturing data using fractal geometry, *Journal of Intelligent manufacturing*, 23, 3, 2012, pp.481-495.
- [4] J. Fodor, R. Klempous and C. Suárez Araujo: *Recent Advances in Intelligent Engineering Systems*, Springer, 2012.
- [5] R. Lindsay, B. Buchanan, E. Feigenbaum and J. Lederberg, DENDRAL: A case study of the first expert system for scientific hypothesis formation, *Artificial Intelligence*, 61, 2, 1993, pp.209-261.
- [6] E. Shortliffe, *Computer-Based Medical Consultations: MYCIN*, Elsevier, 1976.
- [7] R. Miller, H. Pople and J. Myer, INTERNIST-1, An experimental computer-based diagnostic consultant for general internal medicine, *New England Journal of Medicine*, 307, 8, 1982, pp.468-476.
- [8] A. Badiru, *Expert Systems: Applications in Engineering and Manufacturing*, Prentice-Hall, 1992.
- [9] A. Mohammad and N. Al Saiyd, A framework for expert knowledge acquisition, *International Journal of Computer Science and Network Security*, 10, 11, 2010, pp.145-151.
- [10] T. Witkowski, P. Antezak and A. Antezak, Machine learning-based classification in manufacturing system, *Proceedings of the 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, Czech Republic, 2011, pp.580-585.
- [11] C. Leon, F. Biscarri, I. Monedero, J. Guerrero, J. Biscarri and R. Millan, Integrated expert system applied to the analysis of non-technical losses in power utilities, *Expert Systems with Applications*, 38, 2011, pp.10274-10285.
- [12] F. Alonso, L. Martinez, A. Perez and J. Valente, Cooperation between expert knowledge and data mining discovered knowledge: Lesson learned, *Expert Systems with Applications*, 39, 2012, pp.7524-7535.
- [13] J. Quinlan, Induction of decision trees, *Machine Learning*, 1, 1, 1986, pp.81-106.
- [14] M. Berthold, N. Cebron, F. Dill, T. Gabriel, T. Kotter, T. Meini, P. Ohl, C. Sieb, K. Thiel and B. Wiswedel, KNIME: The Konstanz Information Miner, *Proceedings of the 31st Annual Conference of the Gesellschaft für Klassifikation e.V.*, Albert-Ludwigs-Universität Freiburg, 2007, pp.319-326.

Knowledge Mining and Its Application to Support Computational Health Informatics

Nittaya Kerdprasop and Kittisak Kerdprasop

Data Engineering Research Unit, School of Computer Engineering,
Suranaree University of Technology, Nakhon Ratchasima 30000, Thailand

Abstract

Knowledge mining is the process of deriving new and useful knowledge from vast volumes of data and other background information. Derived knowledge can possibly be patterns of data represented in summarized form, relationships among data represented as rules, or representatives of data subgroups represented by majority values. In health and medical domains, knowledge has been discovered in different forms such as association, classification rules, clustering and segmentation, trend or temporal pattern analysis. The discovered knowledge can facilitate computational health informatics in various aspects such as supporting expert decision, making estimation of future trends, diagnosing the causes of illness, and predicting severity of the symptoms. The term *computational health informatics* refers to an emerging field of research focusing on the devise of novel computational techniques to support healthcare and medical applications. In this paper, we propose a knowledge mining method based on the declarative programming scheme, which is a programming paradigm that we argue to be more appropriate for the knowledge intensive tasks than the imperative and object-oriented paradigms. Easy knowledge transfer to the knowledge base content is demonstrated in this paper to confirm the appropriateness of a high-level declarative scheme. Our designed system includes four major knowledge mining tasks: data classification, association mining, sequence analysis, and clustering. We demonstrate knowledge deployment through the automatic generation of knowledge base to support medical decision. The knowledge deployment example illustrates advantage of the high-level declarative scheme to facilitate current computational health informatics.

Keywords: Knowledge Mining, Computational Health Informatics, Healthcare Decision Support System, Declarative Programming

Introduction

Healthcare organizations at the present day regularly generate huge amount of data in electronic form stored in their databases. These data are valuable resource for automatic discovering of useful knowledge, or knowledge mining, to gain insight knowledge potentially useful for a better patient-care decision support. Such implicit knowledge is a valuable asset to most organizations as a substantial source to enhance understanding of data relationships and support better decisions to increase organizational competency. During the past decades there has been an increasing interest in devising database and learning technologies to automatically induce implicit knowledge from clinical and health data (Kretschmann et al., 2001; Lin et al., 2001; Bratsas et al., 2007; Huang et al., 2007; Ghazavi & Liao, 2008; Noren et al., 2008; Zhuang et al., 2009). Most of these work used imperative and object-oriented programming styles. These programming styles are suitable for general software development that requires a nice form of graphical user interface. But for a specific kind of intelligent software development, fast knowledge deduction and accurate induction features are more important than a friendly GUI (Truemper, 2004). We thus design our knowledge mining system based on the declarative paradigm.

This paper presents the design of an intelligent system that has the capability of inducing knowledge from the stored data and automatically ranking and selecting induced

knowledge to be included in the knowledge base. Our system includes data mining engines, that is, the classifier and other miners, as a main feature for extracting hidden knowledge. We, however, broaden typical data mining system by incorporating knowledge management functions with the inclusion of additional features to elicit and store selected knowledge in the knowledge base. Demonstration of the system deployment has been done through the knowledge induction application to support medical and health informatics.

The Design and Implementation of a Declarative Knowledge Mining System

Implicit knowledge acquisition can be achieved through the availability of the knowledge-mining system. *Knowledge mining* is the discovery of hidden knowledge stored possibly in various forms and places in large data repositories. Automatic knowledge acquisition can be achieved through the availability of the data mining engines. The discovered knowledge facilitates expert decision support, data exploration and explanation, estimation of future trends, and prediction of future outcomes based on present data. In this section, we present the design (Figure 1) of a knowledge-mining system named SUT-Miner to support a high-level decision in medical domains.

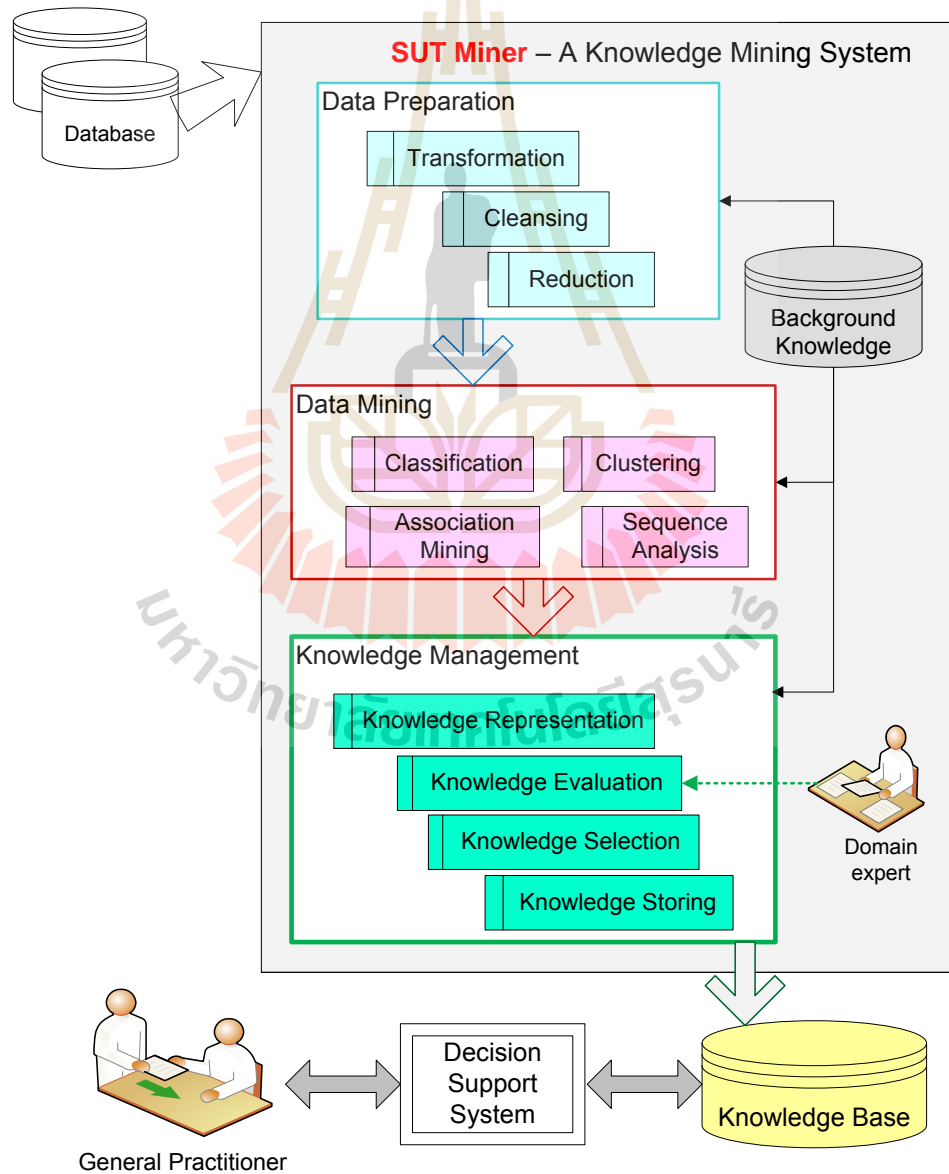


Figure 1. Architecture of the knowledge mining system

Generally, the proposed system can also be applicable to any domain that requires a knowledge-based decision support. A rapid prototyping of the proposed system has been implemented with a declarative style using first-order and second-order Horn clauses (Nadathur & Miller, 1990; Naish, 1996; Kramer & Widmer, 2001). The intuitive idea of our design and implementation is that for such a complicated knowledge-based system program coding should be done declaratively at a high level to alleviate the burden of programmers. The advantages of declarative style are thus the decrease in program development time and the increases in expressiveness of knowledge representation and efficiency of knowledge utilization.

Knowledge induction through various data mining engines is the back-end of the system responsible for acquiring and discovering new and useful knowledge. Usefulness is to be validated at the final step by human experts of the field. Discovered knowledge is stored in the knowledge base to be applied to solve new cases or create new knowledge in the knowledge inferring phase, which is the front-end of the decision support system.

The SUT-Miner system obtains input from heterogeneous data sources. Therefore, redundancy, incompleteness, and noise (*i.e.*, random error) can be expected from the input data. The data preparation component has been designed to clean, transform the format, and select only relevant data with suitable feature selection techniques and possibly sampling to reduce data instances. The data mining component is for performing various mining tasks. Currently, we design and implement four different mining modules, *i.e.*, classification, association mining, sequence analysis, and clustering. Some background knowledge can be adopted to guide the mining methodology selection.

The knowledge management is a novel component that we include in our design to move data mining methodology towards the actual applicability for most general users. In our implementation, knowledge selection module applies some heuristics to rank and elicit knowledge based on possibility of future application. Knowledge evaluation and selection should, however, be confirmed by human experts for correctness and usefulness.

To illustrate working example of the proposed SUT-Miner system, we include in this section some coding of declarative programming based on the Prolog language. The program coding is the classification module using decision-tree induction method known as ID3.

```

mainId3(Min) :-
    init(AllAttr,EdgeList),
    getnode(N),
    create_edge_onelevel(N,AllAttr,EdgeList),
    addKnowledge,
    selectRule(Min,Res),
    maplist(writeln,Res).
create_edge_onelevel(_,_,[]):-!.
create_edge_onelevel(_,[],_):-!.
create_edge_onelevel(N,AllAttr,EdgeList) :- create_nodes(N,AllAttr,EdgeList).
create_nodes(N,AllAttr,[H1-H2/PB-NB|T]) :-
    getnode(N1),
    assert(edge(N,H1=H2,N1)),
    assert(node(N1,PB-NB)),
    append(PB,NB,AllInst),
    ( (PB\==[], NB\==[]) ->
        (cand_node(AllAttr,AllInst,AllSplit),
        min_cand(AllSplite,[V,MinAttr,Split]),
        delete(AllAttr,MinAttr,Attr2),
        create_edge_onelevel( N1,Attr2,Split))
    ; true ),
    create_nodes(N,AllAttr,T).
create_nodes(_,_,[]):-!.

```



```
create_nodes(,[],):-!
```

```
addKnowledge :-
```

```
    findall([A],pathFromRootToLeaf(A,_),Res),  
    retractall(_>>_>>_),  
    maplist(apply(assert),Res).
```

```
selectRule(V,Res):-
```

```
    findall(N>>X>>Class,(X>>Class>>N,N>=V),Res1),  
    sort(Res1,Res2),  
    reverse(Res2,Res).
```

Deployment: Automatic Generation of a Knowledge Base

To demonstrate knowledge deployment, we call the ID3 classification module as shown in the previous section. The pop-up window will appear (Figure 2) on the screen for user to specify data file to be mined. The parameter 'MinProb' is for rule selection. In our classification module, we convert the decision tree, which is the output of the ID3 algorithm (Quinlan, 1986), to be in a form of decision rules. Each rule has been augmented with the probability value, which can be computed as a proportion of:

(number of instances at leaf node) / (total data instances in a data set).

User can specify the minimum probability value for selecting only rules that have high possibility for applying to predict future event. The default of this parameter is 0.001. In main module, the predicates **addKnowledge** and **selectRule(V,Res)** are invoked to compute probability along each tree branch to generate probabilistic rules and then select only rules that could occur at the probability level higher than the specified threshold.



Figure 2. Pop-up window for specifying data filename and minimum probability threshold

In the shown example, we use a data named breast-cancer-wisconsin (available at <http://www.ics.uci.edu/~mlearn/MLRepository.html>). The data had been transformed to be in a first-order clausal format (as shown in Figure 3). The model to be induced from this dataset aims at predicting breast cancer images as either in a benign or malignant stage.

The decision rules induced from this dataset are then automatically transform to be a knowledge base as illustrated in Figure 4. The deployment of this knowledge base has been shown through the inference steps of an expert shell (Figure 5).

```

% data: breast-cancer-wisconsin
attribute(clump_thickness,      [1,2,3,4,5,6,7,8,9,10]).
attribute(uniformity_of_cell_size, [1,2,3,4,5,6,7,8,9,10]).
attribute(uniformity_of_cell_shape, [1,2,3,4,5,6,7,8,9,10]).
attribute(marginal_adhesion,    [1,2,3,4,5,6,7,8,9,10]).
attribute(single_epithelial_cell_size, [1,2,3,4,5,6,7,8,9,10]).
attribute(bare_nuclei,         [1,2,3,4,5,6,7,8,9,10]).
attribute(bland_chromatin,     [1,2,3,4,5,6,7,8,9,10]).
attribute(normal_nucleoli,     [1,2,3,4,5,6,7,8,9,10]).
attribute(mitoses,            [1,2,3,4,5,6,7,8,9,10]).
attribute(class, [benign, malignant]).
instance(1,class=benign,[clump_thickness=5,clump_thickness=1,uniformity_of_cell_size=1,uniformity_of_cell_size=1,uniformity_of_cell_shape=2,uniformity_of_cell_shape=1,marginal_adhesion=3,marginal_adhesion=1,single_epithelial_cell_size=1]).
instance(2,class=benign,[clump_thickness=5,clump_thickness=4,uniformity_of_cell_size=4,uniformity_of_cell_size=5,uniformity_of_cell_shape=7,uniformity_of_cell_shape=10,marginal_adhesion=3,marginal_adhesion=2,single_epithelial_cell_size=1]).
...
instance(699,class=malignant,[clump_thickness=4,clump_thickness=8,uniformity_of_cell_size=8,uniformity_of_cell_size=5,uniformity_of_cell_shape=4,uniformity_of_cell_shape=5,marginal_adhesion=10,marginal_adhesion=4,single_epithelial_cell_size=1]).

```

Figure 3. Breast-cancer-wisconsin dataset in a format of first-order logic clauses.

```

% Knowledge base automatically created for expert shell.
% top_goal is where the inference starts.
top_goal(X,V) :- type(X,V).

% Generated rules:
type(malignant,0.1430615164520744):-
    uniformity_of_cell_shape(10),clump_thickness(10).
type(malignant,0.060085836909871244):-
    uniformity_of_cell_shape(10),clump_thickness(8).
type(benign,0.04721030042918455):-
    uniformity_of_cell_shape(3),clump_thickness(1).
...
type(malignant,0.001430615164520744):-
    uniformity_of_cell_shape(1),marginal_adhesion(4),clump_thickness(2),uniformity_of_cell_size(1).

% Generated menu:
clump_thickness(X):-menuask(clump_thickness,X,[1,2,3,4,5,6,7,8,9,10]).
uniformity_of_cell_size(X):-menuask(uniformity_of_cell_size,X,[1,2,3,4,5,6,7,8,9,10]).
uniformity_of_cell_shape(X):-
    menuask(uniformity_of_cell_shape,X,[1,2,3,4,5,6,7,8,9,10]).
marginal_adhesion(X):-menuask(marginal_adhesion,X,[1,2,3,4,5,6,7,8,9,10]).
single_epithelial_cell_size(X):-
    menuask(single_epithelial_cell_size,X,[1,2,3,4,5,6,7,8,9,10]).
bare_nuclei(X):-menuask(bare_nuclei,X,[1,2,3,4,5,6,7,8,9,10]).
bland_chromatin(X):-menuask(bland_chromatin,X,[1,2,3,4,5,6,7,8,9,10]).
normal_nucleoli(X):-menuask(normal_nucleoli,X,[1,2,3,4,5,6,7,8,9,10]).
mitoses(X):-menuask(mitoses,X,[1,2,3,4,5,6,7,8,9,10]).
class(X):-menuask(class,X,[benign,malignant]).
% end of automatic KB creation

```

Figure 4. Generated knowledge base content and menu to appear in the expert system shell.

```
SWI-Prolog -- d:/10-Research-Experimentations/Inductive-expertSystem/TestProgram/expertshell.pl
File Edit Settings Run Debug Help

1 ?- expertshell.
This is the Easy Expert System shell.
Type help. load. solve. why. quit.
at the prompt.

expert-shell> load.
Enter file name in single quotes (ex. '1.knb'): 'breast-cancer-wisconsin.knb'.
% breast-cancer-wisconsin.knb compiled 0.05 sec, 492 clauses

expert-shell> solve.

What is the value for uniformity_of_cell_shape?
[1-1,2-2,3-3,4-4,5-5,6-6,7-7,8-8,9-9,10-10]
Enter the choice> 8.

What is the value for marginal_adhesion?
[1-1,2-2,3-3,4-4,5-5,6-6,7-7,8-8,9-9,10-10]
Enter the choice> 10.

The answer is __malignant__ with probability 0.017167381974248927

expert-shell> why.

The answer is ...malignant... with probability = 0.017167381974248927.
The known storage are
[marginal_adhesion(10),uniformity_of_cell_shape(8)]

expert-shell> |
```

Figure 5. Inference steps of the expert system shell through the automatically created knowledge base content

Conclusions

Healthcare organizations regularly generate huge amount of data in electronic form and store in heterogeneous databases. These data are a valuable resource for automatic discovering of useful knowledge, known as knowledge mining or data mining, to support high-level decisions.

In this paper we have proposed the design and implementation of SUT-Miner, a declarative knowledge mining system. The system is intended to support automatic knowledge acquisition in medical domains that require new knowledge to support better decisions as well as to enhance comprehension of stored data. The system is also applicable to any domain that requires a knowledge-based decision support. A rapid prototyping of the proposed system is provided in a declarative style using first-order and second-order Horn clauses. The proposed knowledge discovery environment is composed of tools and methods

suitable for various kinds of knowledge discovery tasks including data classification, association discovery, sequence analysis, and data clustering.

Acknowledgment

This research has been conducted at the data engineering research unit, school of computer engineering, Suranaree University of Technology (SUT). This research unit is financially supported by research grant from SUT.

References

1. Bratsas, C., Koutkias, V., Kaimakamis, E., Bamidis, P., Pangalos, G., & Maglaveras, N. (2007). KnowBaSICS-M: An ontology-based system for semantic management of medical problems and computerised algorithmic solutions. *Computer Methods and Programs in Biomedicine*, 83, 39-51.
2. Ghazavi, S., & Liao, T. (2008). Medical data mining by fuzzy modeling with selected features. *Artificial Intelligence in Medicine*, 43, 3, 195-206.
3. Huang, M., Chen, M., & Lee, S. (2007). Integrating data mining with case-based reasoning for chronic diseases prognosis and diagnosis, *Expert Systems with Applications*, 32, 856-867.
4. Kramer, S., & Widmer, G. (2001). Inducing classification and regression trees in first order logic. In S. Dzeroski & N. Lavrac (Eds.), *Relational Data Mining*, pp.140-159, Springer.
5. Kretschmann, E., Fleischmann, W, & Apweiler, R. (2001). Automatic rule generation for protein annotation with the C4.5 data mining algorithm applied on SWISS-PROT. *Bioinformatics*, 17, 10, 920-926.
6. Lin, F., Chou, S., Pan, S., & Chen, Y. (2001). Mining time dependency patterns in clinical pathways. *International Journal of Medical Informatics*, 62, 1, 11-25.
7. Nadathur, G., & Miller, D. (1990). Higher-order Horn clauses. *Journal of the ACM*, 37, 777-814.
8. Naish, L. (1996). Higher-order logic programming in Prolog. *Technical Report 96/2*, Department of Computer Science, University of Melbourne, Australia.
9. Noren, G., Bate, A., Hopstadius, J., Star, K., & Edwards, I. (2008). Temporal pattern discovery for trends and transient effects: Its application to patient records. *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Las Vegas, U.S.A., pp.963-971.
10. Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, 1, 81-106.
11. Truemper, K. (2004). *Design of logic-based intelligent systems*. John Wiley & Sons, New Jersey.
12. Zhuang, Z., Churilov, L., & Burstein, F. (2009). Combining data mining and case-based reasoning for intelligent decision support for pathology ordering by general practitioners. *European Journal of Operational Research*, 195, 3, 662-675.

ภาคผนวก ข

รหัสต้นฉบับของโปรแกรม

1. โปรแกรมการค้นพบรูปแบบที่ปรากฏบ่อยในข้อมูลดีเอ็นเอ (Program to discover frequent patterns in DNA data) ทะเบียนลิขสิทธิ์เลขที่ ว1. 5346 ให้ไว้ ณ วันที่ 8 เมษายน 2558
2. โปรแกรมแบบขนานเพื่อค้นหาความสัมพันธ์ (Parallel association mining program) ทะเบียนลิขสิทธิ์เลขที่ ว1. 5347 ให้ไว้ ณ วันที่ 8 เมษายน 2558



รลข.01

ทะเบียนข้อมูลเลขที่ ว1. 5346

หนังสือรับรองการแจ้งข้อมูล
ลิขสิทธิ์
ออกให้เพื่อแสดงว่า
มหาวิทยาลัยเทคโนโลยีสุรนารี

ได้แจ้งข้อมูลลิขสิทธิ์ ประเภทงาน วรรณกรรม

ลักษณะงาน โปรแกรมคอมพิวเตอร์

ชื่อผลงาน โปรแกรมการค้นพบรูปแบบที่ปรากฏบ่อยในข้อมูลดีเอ็นเอ

ไว้ต่อกรมทรัพย์สินทางปัญญา ตามคำขอแจ้งข้อมูลลิขสิทธิ์ เลขที่ 322071

เมื่อวันที่ 1 เดือน เมษายน พ.ศ. 2558

ให้ไว้ ณ วันที่ 8 เดือน เมษายน พ.ศ. 2558

ลงชื่อ.....

นางสาวศิริวรรณ นพรัถ
นักวิชาการพาณิชย์ปฏิบัติการ
ปฏิบัติราชการแทนผู้อำนวยการสำนักลิขสิทธิ์

หมายเหตุ

1. เอกสารนี้มิได้รับรองความเป็นเจ้าของลิขสิทธิ์
2. การเปลี่ยนแปลงรายการข้างต้น ให้ดูด้านหลัง

ชื่อภาษาไทย	การค้นพบรูปแบบที่ปรากฏบ่อยในข้อมูลดีเอ็นเอ
ชื่อภาษาอังกฤษ	Program to discover frequent patterns in DNA data
ทะเบียนข้อมูลเลขที่	ว1. 5346
ให้ไว้ ณ วันที่	8 เมษายน พ.ศ. 2558
คำอธิบายโปรแกรมโดยย่อ	<p>โปรแกรมการค้นพบรูปแบบที่ปรากฏบ่อยนี้พัฒนาด้วยภาษาเออแลง (Erlang) เนื่องจากเป็นภาษาเชิงฟังก์ชันที่มีความเหมาะสมกับงานที่จะต้องระบุรูปแบบแพทเทิร์นเป็นจำนวนมาก ซึ่งภาษาในแนวทางนี้จะช่วยในลดเวลาในการพัฒนาซอร์สโค้ดลงเป็นอันมากเนื่องจากภาษาเออแลงมีลักษณะของภาษาเชิงประกาศ (declarative) ที่ช่วยให้การเขียนและอ่านโค้ดทำได้ง่ายกว่าภาษาฮาสเกิล</p> <p>โปรแกรมสำหรับงานการค้นพบรูปแบบที่ปรากฏบ่อยนี้ เขียนขึ้นเพื่อค้นพบรูปแบบที่ปรากฏบ่อยในสายรหัสพันธุกรรมหรือดีเอ็นเอ โดยใช้ข้อมูล DNA-nominal จากฐานข้อมูล UCI (http://archive.ics.uci.edu/ml/datasets/) ไฟล์โปรแกรมจะเป็นชื่อเดียวกันกับชื่อโมดูลและมีส่วนขยายเป็น erl ดังนั้นโปรแกรมนี้จึงถูกบันทึกในชื่อ “assoDNA.erl” ทำหน้าที่รู้จำและจำแนกดีเอ็นเอส่วนบรรจรหัส (exon) ออกจากส่วนที่ไม่มีรหัสสำหรับสร้างโปรตีน (intron) รวมถึงความสามารถในการจำแนกจุดเชื่อมต่อ (splice sites) ของเอ็กซอนและอินทรอน งานในลักษณะนี้เรียกว่าการทำนายยีน (gene prediction) หรือการรู้จำรูปแบบของยีน (gene recognition) การทำนายยีน หรือการรู้จำรูปแบบของยีน เป็นปัญหาวิจัยที่สำคัญในงานชีวสารสนเทศ เนื่องจากความรู้ที่ได้จะเป็นพื้นฐานสำคัญของการวิเคราะห์ด้านอื่น เช่น ชนิด ปริมาณ และความสมบูรณ์ของโปรตีนที่จะได้จากกระบวนการสังเคราะห์</p>

```

% Program to discover frequent patterns in DNA data
%
% Created by Kittisak Kerdprasop and Nittaya Kerdprasop
%       Data Engineering Research Unit,
%       Suranaree University of Technology, Thailand.
%
% Start the program by calling
%       c(assoDNA,[export_all]).
% followed by
%       assoDNA:main2().
%
% The program is written in Erlang language
%

-module(assoDNA).
-import(lists, [seq/2, sum/1, flatten/1, split/2, nth/2, map/2, last/1]).
-import(io, [format/1, format/2]).
-import(ordsets, [to_list/1, from_list/1, is_subset/2, union/1]).
% Using this program:
%       c(assoDNA, [export_all]).
%       assoDNA:main2().
% input(_)->[[1,3,4],[2,3,5],[1,2,3,5],[2,5]].
% allItems() ->[1,2,3,4,5]. maxRec()->5. per()->50.
% -----
% input(_)->[["a","c","d"],["b","c","e"],["a","b","c","e"],["b","e"]].
% allItems() ->[["a","b","c","d","e"]].

% maxRec()->5. per()->50.
%-----

input(_ ) -> FileNs=["DNA-nominal.data","DNA-nominal.test"],
    format("~nFile 1.~p 2.~p ",FileNs),
    {_,FNo}=io:read(" :Choose> "),
    FileN=lists:nth(FNo,FileNs),
    format("Read from file:~p",[FileN]),
    readfile(FileN).

allItems() -> [F++[S]||F<- ["A","C","T","G"], S<-seq(048+1,048+60)].

readfile(FileName) -> {ok, Binary} = file:read_file(FileName),
    Lines = string:tokens(erlang:binary_to_list(Binary), "\n\r"),
    L=lists:map(fun(X) -> string:tokens(X," ") end,Lines),
    AD=[addCol(EachL,1) || EachL <-L],
    S=splitClass(["none","exon/intron","intron/exon"], AD),
    AllData=[extract(LL) || LL <-S],
    format("~nThere are 1~w Classes", [length(AllData)]),
    {_,ClassNo}=io:read(" :Choose> "),
    {Class,Data}=lists:nth(ClassNo,AllData),
    io:format("Class =~p ",[Class]),
    Data.

splitClass([],_) -> [];
splitClass([H|T],L) -> [lists:filter(fun(X)->lists:last(X)==H end,L)|splitClass(T,L)].

addCol([X],_) -> [X] ; % except the last
addCol([H|T],N) -> Col=048+N, [ H++[Col] | addCol(T,N+1)].

extract([H|T]) -> {last(H),extract2([H|T])}.
extract2(LL) -> [Rec--[last(Rec)]||Rec<-LL].

main() -> AllInput=input(1111),
    DB=myToSet(AllInput),Total=length(AllInput),{_,Per}=io:read("input percent> "),
    MinSup=Total*Per/100,
    format("~nTotal=~w ,~w% MinSup=~w",[Total,Per,MinSup]),
    apriori(DB,allItems(),MinSup).

```



```

myToSet(L) -> [from_list(X)||X<-L].
myToList(SL) -> [to_list(S)||S<-SL].

apriori(DB, Items, Min) ->
  C1=[{from_list([X]),findSup(from_list([X]),DB)}|| X<-Items ],
  L1=[{FS,Sup} || {FS,Sup}<-C1,Sup>=Min],
  % print L1
  LkPrint=[ {to_list(FS),Sup,Sup/length(DB)*100} || {FS,Sup}<-L1],
  format("~nK=~w~p, has ~w set ",[1,LkPrint,length(LkPrint)]),
  K=2, LS=[FS||{FS,_}<-L1],
  aprioriLoop(L1,DB,LS,K,Min) .

findSup(_, []) ->0;
findSup(Set, DB) -> [H|T]=DB,
  Cond = is_subset(Set,H),
  if Cond->1+findSup(Set,T);
  true -> findSup(Set,T)
end.

aprioriLoop(AllL,_,[],_,_) -> AllL;
aprioriLoop(AllL,_,[],_,_) -> AllL;
aprioriLoop(AllL,DB,LS,K,Min) -> Com=combi(LS),
  C = myDistinct(usedCombi(Com,K)),
  Ck=[{X,findSup(X,DB)}|| X<-C_],
  Lk=[ {FS,Sup} || {FS,Sup}<-Ck,Sup>=Min],
  LkS=[FS||{FS,_}<-Lk],
  LkPrint=[ {to_list(FS),Sup,Sup/length(DB)*100} || {FS,Sup}<-Lk],
  format("~nK=~w~p, has ~w set ",[K,LkPrint,length(LkPrint)]),
  aprioriLoop(AllL++Lk,DB,LkS,K+1,Min) .

myDistinct(List) -> to_list(from_list(List)).
combi([H|T]) -> [[H,Te] || Te<-T]++ combi(T);
combi([]) -> [].
usedCombi([H|T], K) -> Union=union(H),
  Len=ordsets:size(Union),
  if Len==K -> [Union|usedCombi(T,K)];
  true -> usedCombi(T,K)
end ;
usedCombi([],_) -> [].
shift([H|T]) ->T++[H].

genR(_,Max,Max) -> [];
genR(L,N,Max) -> {H,T} = lists:split(N,L),[{H,T}]++genR(L,N+1,Max).

genRule(_,0,_) -> [];
genRule(L,Count,Len) -> genR(L,1,Len)++genRule(shift(L),Count-1,Len).

set(X) -> from_list(X).
list(X) -> to_list(X).

searchL(Set, [{Set,Val}|_]) -> Val;
searchL(Set, [{_Another,_}|T]) -> searchL(Set,T);
searchL(_Set,[]) -> 1 .
findConf({H,B},AllL)-> {H,B,searchL(set(H++B),AllL)/searchL(set(H),AllL)}.
sortConf({__,_},Conf1),({__,_},Conf2) -> Conf1>Conf2.

main2() ->
  format("~n-----START-----"),
  AllL=main(),
  AllAsso2=[list(X)|| {X,_} <-AllL,length(list(X))>1 ],
  AllRuleGen=lists:flatten([genRule(L,length(L),length(L))||L<- AllAsso2]),
  AllRuleConf=[findConf(X,AllL)||X<-AllRuleGen],
  format("~n~n AllRule=~w ,~nThere are ~w rules",
  ", [AllRuleConf,length(AllRuleConf)]),
  lists:sort({assoDNA,sortConf},AllRuleConf),
  format("~n-----") .

```

```

% ---Test Module-----
% c(assoDNA,[export_all]).
% assoDNA:findSupOf(["AM","GN"]).
% assoDNA:findPos(["AM","GN"]).
findPos([])->ok;
findPos([H|T])->[F,S]=H,format("~p~w--",[F],S-48),findPos(T).
subList(L1,L)->length(L--L1)==length(L)-length(L1).
findSup1(L1,LL)->lists:filter(fun(L)->subList(L1,L) end,LL) .
findSupOf(Lfind)-> format("\DNA-nominal.data","\DNA-nominal.test\" "),
  {_,FileName}=io:read(" Start NewJob FileName> "),
  {ok, Binary} = file:read_file(FileName),
  Lines = string:tokens(erlang:binary_to_list(Binary), "\n\r"),
  L=lists:map(fun(X) -> string:tokens(X," ") end,Lines),
  AD=[addCol(EachL,1) || EachL <-L],
  S1=splitClass(["none","exon/intron","intron/exon"],AD) ,
  [{_,CL1},{_,CL2},{_,CL3}] = [extract(LL1) || LL1 <-S1],
  OLen=[length(CL1),length(CL2),length(CL3)],
  Re=findSup1(Lfind,AD),
  S=splitClass(["none","exon/intron","intron/exon"],Re),
  AllData=[extract(LL) || LL <-S],
  myprint(OLen,AllData).

myprint([],_) -> endOfPrint;
myprint([OH|OT],[{C,LL}|T]) ->
  format("~nClass:~p has ~w = ~w percentOf
  ~w",[C,length(LL),length(LL)/OH*100,OH]),
  myprint(OT,T).

% ===== END OF DNA FREQUENT PATTERN DISCOVERY PROGRAM ===== %

```



รลข.01

ทะเบียนข้อมูลเลขที่ ว1. 5347

หนังสือรับรองการแจ้งข้อมูล
ลิขสิทธิ์
ออกให้เพื่อแสดงว่า
มหาวิทยาลัยเทคโนโลยีสุรนารี

ได้แจ้งข้อมูลลิขสิทธิ์ ประเภทงาน วรรณกรรม

ลักษณะงาน โปรแกรมคอมพิวเตอร์

ชื่อผลงาน โปรแกรมแบบขนานเพื่อค้นหาความสัมพันธ์

ไว้ต่อกรมทรัพย์สินทางปัญญา ตามคำขอแจ้งข้อมูลลิขสิทธิ์ เลขที่ 322072

เมื่อวันที่ 1 เดือน เมษายน พ.ศ. 2558

ให้ไว้ ณ วันที่ 8 เดือน เมษายน พ.ศ. 2558

ลงชื่อ.....

นางสาวศิริวรรณ นพรัถ์

นักวิชาการพาณิชย์ปฏิบัติการ

ปฏิบัติราชการแทนผู้อำนวยการสำนักลิขสิทธิ์

หมายเหตุ

1. เอกสารนี้มิได้รับรองความเป็นเจ้าของลิขสิทธิ์
2. การเปลี่ยนแปลงรายการข้างต้น ให้ดูด้านหลัง

ชื่อภาษาไทย	โปรแกรมแบบขนานเพื่อค้นหาความสัมพันธ์
ชื่อภาษาอังกฤษ	Parallel association mining program
ทะเบียนข้อมูลเลขที่	ว1. 5347
ให้ไว้ ณ วันที่	8 เมษายน พ.ศ. 2558
คำอธิบายโปรแกรมโดยย่อ	<p>โปรแกรมแบบขนานเพื่อค้นหาความสัมพันธ์นี้พัฒนาด้วยแนวทางการโปรแกรมแบบพร้อมกัน (concurrent program) ด้วยภาษาเอแอลง โดยใช้ในแอปพลิเคชันการวิเคราะห์รูปแบบสายรหัสพันธุกรรมใน 3 ตำแหน่งคือ ส่วนเอ็กซอนที่เชื่อมต่อกับอินทรอน ส่วนอินทรอนที่เชื่อมต่อกับเอ็กซอน และส่วนที่ไม่ใช่ทั้งเอ็กซอนและอินทรอน ข้อมูลนี้คือข้อมูล primate splice-junction gene sequences ได้มาจากฐานข้อมูล GenBank 64.1 และดาวน์โหลดได้จาก UCI machine learning repository (http://mlern.ics.uci.edu/databases/molecular-biology/splice-junction-gene-sequences)</p> <p>การค้นหาแบบรหัสพันธุกรรมในสามส่วนนั้นจะทำพร้อมกัน เพื่อเพิ่มความเร็วในการทำงาน รูปแบบของรหัสพันธุกรรมเป็นรูปแบบเชิงสัมพันธ์ (association patterns) ที่สามารถกำหนดค่า support ที่เป็นเกณฑ์การปรากฏบ่อยของรูปแบบ ผลลัพธ์ของโปรแกรมจะเป็นการรายงานตำแหน่งนิวคลีโอไทด์ที่มักจะปรากฏในแต่ละกรณีของ splice-junction ในการพัฒนาโปรแกรมยังได้เสนอแนวทางการประมวลผลแบบลำดับเพื่อใช้เปรียบเทียบความเร็วในการทำงานกับแบบขนาน</p>

```

% Parallel Association Mining Program
%
% Created by Kittisak Kerdprasop and Nittaya Kerdprasop
%       Data Engineering Research Unit,
%       Suranaree University of Technology, Thailand.
%
% Start the program by calling
%       c(assoDNA_par,[export_all]).
%
% This Erlang language program is written to mine association rules concurrently.
%

-module(assoDNA_par) .
-import(lists, [sublist/2, seq/2, sum/1, flatten/1, split/2, nth/2, map/2, last/1]) .
-import(io, [format/1, format/2]) .
-import(ordsets, [to_list/1, from_list/1, is_subset/2, union/1]) .

% Using this program:
%% c(assoDNA_par, [export_all]) .
%% File1,Class,MinSup=80%
%% timer:tc(assoDNA_par,run,[self(),[1,1,80,1,2,80,1,3,80]]). %%11265000 microsec
%% f(), {T,_}=timer:tc(assoDNA_par,concurrent,[1,1,80],[1,2,80],[1,3,80]).
%%5422000 microsec

%% File1,Class,MinSup=60%
%% timer:tc(assoDNA_par,run,[self(),[1,1,60,1,2,60,1,3,60]]). %%15125000 microsec
%% f(), {T,_}=timer:tc(assoDNA_par,concurrent,[1,1,60],[1,2,60],[1,3,60]).
%%9125000 microsec

%% File1,Class,MinSup=40%
%% timer:tc(assoDNA_par,run,[self(),[1,1,40,1,2,40,1,3,40]]). %%39609000 microsec
%% f(), {T,_}=timer:tc(assoDNA_par,concurrent,[1,1,40],[1,2,40],[1,3,40]).
%%18657000 microsec

concurrent(P1,P2,P3)->
    spawn(assoDNA_par,run,[self(),P1]),
    spawn(assoDNA_par,run,[self(),P2]),
    spawn(assoDNA_par,run,[self(),P3]),
    receive
        my_end-> ok
    end.
fileNames()-> ["spliceDNA.DATA","spliceDNA-Test.TEST"].

input(InputL)-> FileNs=fileNames(),
    [FNo,ClassNo,_]=InputL,FileN=lists:nth(FNo,FileNs),
    format("~n=====Read from file:~p=====",[FileN]),
    {ok,Binary}=file:read_file(FileN),
    Lines=string:tokens(erlang:binary_to_list(Binary),"\\n\\r"),
    L=lists:map(fun(X)->string:tokens(X," ")end,Lines),
    AD=[addCol(EachL,1)||EachL<-L], % all data
    S=splitClass(["none","exon/intron","intron/exon"],AD),
    AllData=[extract(my_no_value,LL)||LL<-S],
    format("~nThere are 1-~w Classes",[length(AllData)]),
    {Class,Data}=lists:nth(ClassNo,AllData),
    io:format("Class =~p",[Class]),
    {AD,Data,FNo,ClassNo}.

% all available of gene at any Col for finding C1
allItems()-> [F++[S]||F<-["A","C","T","G","D","N","S","R"],
    S<-seq(048+1,048+30)++seq(048+32,048+60)].

addCol([X],_)->[X]; % except the last
addCol([H|T],N) when N<31->Col=048+N,[H++[Col]|addCol(T,N+1)];
addCol([H|T],N)->Col=049+N,[H++[Col]|addCol(T,N+1)]. % skip Col=31

to_Col2(LL)->lists:map(fun([A,Col])->[A]++integer_to_list(Col-048)end,LL).

```

```

% to_Col3("exon/intron")->"exon/intron";
to_Col3(LL)->
  lists:map(fun([A,Col])->[A]++("++integer_to_list(Col-048-31)++)" end,LL).

apriori(DB,Items,Min)->
  C1=[{from_list([X]),findSup(from_list([X]),DB)} || X<-Items ],
  L1=[{FS,Sup} || {FS,Sup}<-C1,Sup>=Min],
  LkPrint=[ {to_list(FS),Sup,Sup/length(DB)*100} || {FS,Sup}<-L1],
  K=2, LS=[FS||{FS,_}<-L1],
  aprioriLoopPar(L1,DB,LS,K,Min) .

findSup(_,[])->0;
findSup(Set,DB)->[H|T]=DB,
  Cond =is_subset(Set,H),
  if Cond->1+findSup(Set,T);
  true -> findSup(Set,T)
end.

aprioriLoopPar(AllL,_,[],_,_) ->format("~n~p~n",[AllL]),AllL; % return final Set
aprioriLoopPar(AllL,_,[],_,_) ->format("~n~p~n",[AllL]),AllL;
aprioriLoopPar(AllL,DB,LS,K,Min)->
  Com=combi(LS),
  C_=myDistinct(usedCombi(Com,K)),
  Ck=[{X,findSup(X,DB)} || X<-C_],
  Lk=[ {FS,Sup} || {FS,Sup}<-Ck,Sup>=Min],
  LkS=[FS||{FS,_}<-Lk],
  LkPrint=[ {to_list(FS),Sup,Sup/length(DB)*100} || {FS,Sup}<-Lk],
  LkPrint3=lists:map(fun({F,S,T})->{to_Col3(F),S,T} end,LkPrint),
  format("~nK=~w~p, has ~w set ~n ",[K,LkPrint3,length(LkPrint)]),
  aprioriLoopPar(AllL++Lk,DB,LkS,K+1,Min) .
%%----
aprioriLoop(AllL,_,[],_,_) -> format("~n~p~n",[AllL]),AllL; % return final Set
aprioriLoop(AllL,_,[],_,_) -> format("~n~p~n",[AllL]),AllL;
aprioriLoop(AllL,DB,LS,K,Min)->
  Com=combi(LS),
  C_=myDistinct(usedCombi(Com,K)),
  Ck=[{X,findSup(X,DB)} || X<-C_],
  Lk=[ {FS,Sup} || {FS,Sup}<-Ck,Sup>=Min],
  LkS=[FS||{FS,_}<-Lk],
  LkPrint=[ {to_list(FS),Sup,Sup/length(DB)*100} || {FS,Sup}<-Lk],
  LkPrint3=lists:map(fun({F,S,T})->{to_Col3(F),S,T} end,LkPrint),
  format("~nK=~w~p, has ~w set ~n ",[K,LkPrint3,length(LkPrint)]),
  aprioriLoop(AllL++Lk,DB,LkS,K+1,Min) .
%%----
myDistinct(List)->to_list(from_list(List)).
combi([H|T])->[H,Te] || Te<-T]++ combi(T);
combi([])->[].
usedCombi([H|T],K)-> Union=union(H),
  Len=ordsets:size(Union),
  if Len==K -> [Union|usedCombi(T,K)];
  true -> usedCombi(T,K)
end ;
usedCombi([],_)->[].

shift([H|T])->T++[H].
genR(_,Max,Max)->[];
genR(L,N,Max)->{H,T}=lists:split(N,L),
  [{H,T}]++genR(L,N+1,Max) .

% genRule([2,3,5],3,3) .
genRule(_,0,_)->[];
genRule(L,Count,Len)->genR(L,1,Len)++genRule(shift(L),Count-1,Len) .

set(X)->from_list(X) . list(X)->to_list(X) .
searchL(Set,[{Set,Val}|_]) -> Val;
searchL(Set,[{Another,_}|T]) -> searchL(Set,T);

```

```

searchL(_Set,[])-> 1 . % Cannot find Set

findConf({H,B},AllL) -> {H,B,searchL(set(H++B),AllL)/searchL(set(H),AllL) }.

sortConf({_,_},Conf1){_,_},Conf2)-> Conf1>Conf2.

% FNo=1 is DATA,ThisClass=2 is "exon/intron"
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% MAIN : run() %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% run(ID,[1,2,80]).

run(MasterID,InputL)->
  R=main2(any,3,InputL),file:delete("out.txt"), %HHHHHH
  AD=lists:last(R), % all data
  [ADD|_]=AD, % c(128,[ADD,lists:last(AD)])
  Rules=lists:sublist(R,length(R)-1),
  PrintRules=map(fun({D,S,Per,Class})->
    {to_Col3(notLast(D)),S,Per, transformBack(Class)} end, Rules),
  ADP=lists:map(fun(Data)-> {Data,checkRules(Data, Rules)} end,AD),
  ADPprint=map(fun({Data,V})-> Predict=transformBack(V),
    {Data,[last(Data),Predict, mark(last(Data),Predict)]} end,ADP),
  Predict=map(fun({F,S})->{to_Col3(notLast(F)),S} end, ADPprint),
  writeToFile(Predict),
  [_|Stop|_]=InputL,
  if Stop ==2 -> MasterID!my_end ;
  true -> MasterID!not_end
end.

% write to text file
writeToFile(Data)-> {ok,FP}=file:open("out.txt",[append]),
  io:format(FP,"~p",[Data]),
  file:close(FP).

mark(V1,V2)->V1==V2.
first([])-> no_first;
first([H|T])-> H.
notLast(F)-> sublist(F,length(F)-1).

checkRules(D,[])-> 1;
checkRules(D,[H|T])-> DataL=lists:sublist(D,length(D)-1), %no class attached
  {L,S,Con,C}=H,Condition=subList(L,DataL),
  if Condition-> C ;
  true ->checkRules(D,T)
end.

transformBack(1)-> "none";
transformBack(2)-> "exon/intron";
transformBack(3)-> "intron/exon".

main2(_|AD,[])->[AD];
main2(_|_|InputL)-> {AD,AllInput,FNo,ThisClass}=input(InputL), %%%HHH
  DB = myToSet(AllInput),Total=length(AllInput),
  [_|_|Per|LT] = InputL, % = 80,
  MinSup=Total*Per/100,
  format("~n-----START---Apriori(in class=~p,Min Support~p%~p)---
    ",[ThisClass,Per,MinSup]),
  AllL=apriori(DB,allItems(),MinSup), % find asso of this class
  LL=lists:map(fun({CodeL,_})->CodeL end,AllL) %extract code list
  [{"AM"},{"AM"},{"CL"} ...],
  ConfLL= lists:map(fun({Lfind,V})->
  {Lfind,V,myfindSupOf(Lfind,FNo,ThisClass),ThisClass} end,AllL),
  SortByConf=lists:sort(fun({A,S1,K1,_},{B,S2,K2,_})-> K1>=K2 end,ConfLL),
  Filter=lists:filter(fun({_|_|K1,_})->
    K1>=85 end,SortByConf), % 85=max% confidence
  Filter++main2(any,AD,LT) .

%%%
---Modules For Testing-----
% c(assoDNA_par,[export_all]), assoDNA_par:findSupOf(["AM","GN"]).
% myfindSupOf(["AM","GN"],1,2).
% assoDNA:myfindSupOf(["GN","GP","GT","TQ"],1,2). %support=63,conf= 95%
% assoDNA:myfindSupOf(["GN","GP","GT"],1,2). % support=64,conf=82 %

```

```

myfindSupOf(Lfind,FNo,ThisClass)->
  FileNs=fileNames()
  ,FileName=lists:nth(FNo,FileNs),
  {ok, Binary} = file:read_file(FileName),
  Lines = string:tokens(erlang:binary_to_list(Binary), "\n\r"),
  L=lists:map(fun(X) -> string:tokens(X," ") end,Lines)
  ,AD=[addCol(EachL,1) || EachL <-L]
  ,S1=splitClass(["none","exon/intron","intron/exon"],AD)
  ,[{_,CL1},{_,CL2},{_,CL3}] = [extract(my_no_value,LL1) || LL1 <-S1]
  ,OLen=[length(CL1),length(CL2),length(CL3)]
  ,Re=findSup1(Lfind,AD) % may be []
  ,S=splitClass(allClass(),Re)
  ,[LC1,LC2,LC3]=S,[C1,C2,C3]=allClass()
  ,AllData=[extract(C1,LC1), extract(C2,LC2),extract(C3,LC3)]
  ,{Cc1,Lc1}=lists:nth(ThisClass,AllData) % ThisClass
  ,[{Cc2,Lc2},{Cc3,Lc3}]=AllData--[{Cc1,Lc1}] % other classes
  ,ThisConf=length(Lc1)*100/(length(Lc1)+length(Lc2)+length(Lc3))
  ,ThisConf.

subList(L1,L)->length(L--L1)==length(L)-length(L1).
splitClass([],_)->[];
splitClass([H|T],L)->[lists:filter(fun(X)->lists:last(X)==H end,L)|splitClass(T,L)] .
findSup1(L1,LL)->lists:filter(fun(L)->subList(L1,L) end,LL).
findSupOf(Lfind)-> %search Lfind in all Classes
  FileNs=fileNames()
  ,format("\nFile 1.~p 2.~p ",FileNs)
  ,{_,FNo}=io:read(" Start NewJob FileName> ")
  ,FileName=lists:nth(FNo,FileNs),
  {ok, Binary} = file:read_file(FileName),
  Lines = string:tokens(erlang:binary_to_list(Binary), "\n\r"),
  L=lists:map(fun(X) -> string:tokens(X," ") end,Lines)
  ,AD=[addCol(EachL,1) || EachL <-L]
  ,S1=splitClass(["none","exon/intron","intron/exon"],AD)
  ,[{_,CL1},{_,CL2},{_,CL3}] = [extract(my_no_value,LL1) || LL1 <-S1]
  ,OLen=[length(CL1),length(CL2),length(CL3)]
  ,Re=findSup1(Lfind,AD) % may be []
  ,S=splitClass(allClass(),Re)
  ,[LC1,LC2,LC3]=S,[C1,C2,C3]=allClass()
  ,AllData=[extract(C1,LC1), extract(C2,LC2),extract(C3,LC3)].

allClass()-> ["none","exon/intron","intron/exon"].
extract(C,[_])->{C,[_]};
extract(_,[H|T])->{last(H),extract2([H|T])}.
extract2(LL)->[Rec--[last(Rec)]||Rec<-LL].
myprint([],_)->endOfPrint;
myprint([OH|OT],[{C,LL}|T])->
  format("\nClass:~p has ~w = ~w percentOf ~w",[C,length(LL),length(LL)/OH*100,OH])
  ,myprint(OT,T).
c(Line,Re)->format("\nin~pCheck=~p~n",[Line,Re]).
myToSet(L)->[from_list(X)||X<-L].
myToList(SL)->[to_list(S)||S<-SL].
mySort(L)->lists:sort(fun({A,F1,K1},{B,F2,K2})->F1=<F2,K1=<K2 end,L).
% s()=fun({A,K1,_},{B,K2,_})->K1=<K2.

% c(assoDNA,[export_all]).
% assoDNA:main1().
%
% c(assoDNA,[export_all]) , assoDNA:main1().
% erlang:spawn_opt(assoDNA,main2,[1,2],[{min_heap_size,466}]).

% ===== END OF PARALLEL ASSOCIATION MINING PROGRAM ===== %

```


ประวัติผู้วิจัย

รองศาสตราจารย์ ดร.นิตยา เกิดประสพ สำเร็จการศึกษาในระดับปริญญาเอกสาขา Computer Science จาก Nova Southeastern University เมือง Fort Lauderdale รัฐฟลอริดา สหรัฐอเมริกา เมื่อปีพุทธศักราช 2542 (ค.ศ. 1999) ด้วยทุนการศึกษาของกระทรวงวิทยาศาสตร์และเทคโนโลยี โดยทำวิทยานิพนธ์ระดับปริญญาเอกในหัวข้อเรื่อง "The application of inductive logic programming to support semantic query optimization" หลังสำเร็จการศึกษาได้ปฏิบัติราชการในตำแหน่งอาจารย์ ประจำสาขาคอมพิวเตอร์ ภาควิชาคณิตศาสตร์ คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ต่อมาในปีพุทธศักราช 2543 ได้มาปฏิบัติงานในตำแหน่งอาจารย์ประจำ สาขาวิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเทคโนโลยีสุรนารี จนถึงปัจจุบัน งานวิจัยที่ทำในขณะนี้ คือการประยุกต์เทคโนโลยีเหมืองข้อมูลกับงานด้านการแพทย์ การสาธารณสุขและสิ่งแวดล้อม รวมถึงการพัฒนาเทคนิคเพื่อเพิ่มความสามารถในการจัดการความรู้ของระบบเหมืองข้อมูล

รองศาสตราจารย์ ดร.กิตติศักดิ์ เกิดประสพ สำเร็จการศึกษาในระดับปริญญาเอกสาขา Computer Science จาก Nova Southeastern University เมือง Fort Lauderdale รัฐฟลอริดา สหรัฐอเมริกา เมื่อปีพุทธศักราช 2542 (ค.ศ. 1999) ด้วยทุนการศึกษาของทบวงมหาวิทยาลัย (หรือสำนักงานคณะกรรมการอุดมศึกษาในปัจจุบัน) โดยทำวิทยานิพนธ์ระดับปริญญาเอกในหัวข้อเรื่อง "Active database rule set reduction by knowledge discovery" หลังสำเร็จการศึกษาได้ปฏิบัติงานในตำแหน่งอาจารย์ ประจำสาขาวิชาวิศวกรรมคอมพิวเตอร์ สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี ปัจจุบันดำรงตำแหน่งหัวหน้าหน่วยวิจัยวิศวกรรมความรู้ เน้นการวิจัยเกี่ยวกับการพัฒนาระบบเหมืองข้อมูลประสิทธิภาพสูง การประยุกต์เหมืองข้อมูลกับงานวิศวกรรม และการวิเคราะห์ข้อมูลเชิงสถิติ รวมถึงการวิจัยพื้นฐานเกี่ยวกับเทคนิคการวิเคราะห์ข้อมูลโดยวิธีอัตโนมัติ โดยมีผลงานวิจัยตีพิมพ์ในวารสารวิชาการและเอกสารการประชุมวิชาการจำนวนมากกว่า 290 เรื่องในด้านฐานข้อมูล การวิเคราะห์ข้อมูล การทำเหมืองข้อมูลและการค้นหาความรู้