อภินันทนาการ

สำนักหอสมุด

รายงานฉบับสมบูรณ์

(Final Report)

เรื่อง

การศึกษาเปรียบเทียบประสิทธิภาพของวิธีจีเนติกและวิธีพาร์ทิเคิลสวอม เพื่อจัดตารางการผลิตในอุตสาหกรรม

สำนักหอสมุด มหาวิทยาลัยนเรศวร
วันลงทะเบียน 12 ต.ค. 2554
เลขทะเบียน 1576147
เลขเรียกหนังสือ ว TA 5 ภ165ร 2551

โดย

ผู้ช่วยศาสตราจารย์ ดร. ภูพงษ์ พงษ์เจริญ

ภาควิชาวิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยนเรศวร

ทุนอุดหนุนการวิจัยงบรายได้คณะวิศวกรรมศาสตร์

มหาวิทยาลัยนเรศวร

ประจำปี 2551

ประกาศคุณูปการ

(Acknowledgements)

ผู้วิจัยขอขอบคุณคณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร ที่ให้การสนับสนุนทุนอุดหนุนการวิจัยจากงบรายได้ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร ประจำปี 2551 และขอขอบคุณ ภาควิชาวิศวกรรม อุตสาหการ คณะวิศวกรรมศาสตร์ ที่ได้ให้ความสะดวกด้านสถานที่ เครื่องมือและอุปกรณ์ในการทำวิจัย อีกทั้งขอขอบคุณหน่วยงานต่างๆ เช่นหน่วยวิจัย ของคณะวิศวกรรมศาสตร์ และหน่วยงานอื่นๆ ที่เกี่ยวข้อง ที่ให้ความสะดวกในด้านข้อมูล เอกสารแบบฟอร์มต่างๆ ที่ต้องใช้ประกอบในการดำเนินการเบิกจ่ายเงินวิจัย

ผศ.ดร.ภูพงษ์ พงษ์เจริญ
ผู้วิจัย
วันที่ 10 กันยายน 2551

บทคัดย่อ

ปัญหาการจัดตารางการผลิตในอุตสาหกรรมเป็นปัญหาประเภท NP-hard กล่าวคือ การหาจำนวนทางเลือกหรือจำนวนคำตอบที่เป็นไปได้มีเป็นจำนวนมาก ที่ต้องใช้เวลาในการหาคำตอบยาวนาน และจำนวนของค่าคำตอบที่เป็นไปได้จะเพิ่มขึ้นอย่างรวดเร็วเมื่อขนาดของปัญหาใหญ่ขึ้น การแก้ปัญหาด้วยวิธีทางคณิตศาสตร์จึงเป็นไปได้ยากมาก อีกทั้งยังใช้เวลาในการคำนวณนานมากเทคนิคใหม่ๆ อย่างวิธี metaheuristics ที่ประสบความสำเร็จ ซึ่งมีวิธีการค้นหาคำตอบแบบสุ่ม stochastic search methods ได้แก่ วิธีจีเนติกอัลกอริทึม (Genetic Algorithms: GA) และวิธีพาร์ทิเคิลสวอมออฟติไมเซชัน (Particle Swarm Optimisation: PSO) จึงเป็นอีกแนวทางหนึ่งที่เหมาะสมและได้รับความนิยมในการนำมาใช้แก้ปัญหาการจัดตารางการผลิต

โครงงานวิจัยนี้จึงได้สร้างโปรแกรมต้นแบบที่ศึกษาเปรียบเทียบประสิทธิภาพการทำงานของวิธีจีเนติกอัลกอริทึม และวิธีพาร์ทิเคิลสวอมออฟติไมเซชัน เพื่อจัดตารางการผลิตในอุตสาหกรรม โดยโปรแกรมดังกล่าวสามารถคำนวณและแสดงผลการจัดตารางการผลิตตามที่ผู้ใช้จะกำหนดปัญหาผ่านระบบต่อเชื่อมแบบรูปภาพ (Graphical User Interface) อีกทั้งยังสามารถเปลี่ยนแปลงค่าปัจจัยของทั้ง 2 วิธีการได้เพื่อความสะดวกในการทำการทดลอง

ทั้งนี้ ในระหว่างการดำเนินโครงการวิจัย ผลงานวิจัยบางส่วนโดยเฉพาะประเด็นผลการวิจัยเชิงลึกในด้านทฤษฎีต่างๆ (เช่น ค่าปัจจัยและกลไกที่มีผลกระทบต่อประสิทธิภาพการทำงานของวิธีจีเนติกอัลกอริทึม หรือการเปรียบเทียบประสิทธิภาพการหาคำตอบของวิธีพาร์ทิเคิลสวอมออฟติไมเซชัน) ได้ถูกเขียนเป็นบทความวิจัยและนำเสนอแบบบรรยายในที่ประชุมวิชาการระดับนานาชาติจำนวน 1 บทความ ดังรายละเอียดในภาคผนวกของรายงานฉบับนี้แล้ว

ABSTRACT

Production scheduling problem is mostly classified as NP hard problem, which means that the amount of computation required to find an optimal solution increases exponentially with problem size. This makes it time consuming and difficult to manually schedule that satisfy the objectives and constraints. Stochastic search methods such as Genetic Algorithm and Particle Swarm Optimisation have been recently adopted for solving these kinds of problem.

In this research project, a prototype of computer aided scheduling tool has been developed. Both Genetic Algorithms and Particle Swarm Optimisation have been embedded in the tool for finding the optimum solutions that minimise the total penalty cost related to earliness and tardiness of production. The graphic user interfaces provided within the tool allow user to choose the problem and the parameters associated with the algorithms. The interface is very helpful for conducting a comprehensive computational experiment.

During this research project, parts of this research have been written as research articles and published in the international journals (see more detail in the appendix).

ข้อสรุปโครงการ
(Executive Summary)

---

1. ข้อมูลโครงการ

ชื่อโครงการ (ภาษาไทย)	การศึกษาเปรียบเทียบประสิทธิภาพของวิธีจีเนติกและวิธีพาร์ทิเคิลสวอมเพื่อจัดตารางการผลิตในอุตสาหกรรม
(ภาษาอังกฤษ)	A comparative study on the performance of Genetic Algorithms and Particle Swarm Optimisation for production scheduling
ชื่อหัวหน้าโครงการ	นายภูพงษ์ พงษ์เจริญ
ที่อยู่	คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อำเภอเมือง จังหวัดพิษณุโลก 65000
โทรศัพท์	0-5526-1000 ต่อ 4256, 4201
โทรสาร	0-5526-1000 ต่อ 4254
E-mail	Pupongp@nu.ac.th or Pupongp@yahoo.com
สาขาวิชาที่ทำการวิจัย	สาขาวิชาวิศวกรรมอุตสาหการ
งบประมาณทั้งโครงการ	100,000 บาท
ระยะเวลาดำเนินงาน	1 ปี (เริ่มวันที่ 1 ตุลาคม 2550 ถึง 30 กันยายน 2551)

2. ที่มาและปัญหางานวิจัย

ปัญหาการจัดตารางการผลิตเป็นปัญหาที่อยู่ในกลุ่มปัญหาประเภท NP-hard (Garey and Johnson, 1979 and Evan et al., 1976) กล่าวคือ การหาจำนวนทางเลือกหรือจำนวนคำตอบที่เป็นไปได้มีเป็นจำนวนมากที่ต้องใช้เวลาในการหาคำตอบยาวนาน และจำนวนของค่าคำตอบที่เป็นไปได้จะเพิ่มขึ้นอย่างรวดเร็วเมื่อขนาดของปัญหาใหญ่ขึ้น การแก้ปัญหาด้วยวิธีทางคณิตศาสตร์จึงเป็นไปได้ยากมาก อีกทั้งยังใช้เวลาในการคำนวณนานมาก และอาจจะพบปัญหาเรื่องหน่วยความจำได้ (Nagar et al., 1995) เทคนิคใหม่ๆ อย่างวิธี meta-heuristics ที่ประสบความสำเร็จ (Osman and Laporte, 1996) ซึ่งมีวิธีการค้นหาคำตอบแบบสุ่ม stochastic search methods ได้แก่ Simulated Annealing (SA), Tabu Search (TS), Genetic Algorithms (GA), Neural Network (NN), Ant Colony Optimisation (ACO), Particle Swarm Optimisation (PSO) จึงเป็นแนวทางที่เหมาะสมในการนำมาใช้แก้ปัญหาการจัดตารางการผลิต

ดังนั้นงานวิจัยนี้จึงมีแนวทางที่จะศึกษาเปรียบเทียบในด้านประสิทธิภาพการทำงานของวิธีจีเนติกอัลกอริทึม (Genetic Algorithms) และวิธีพาร์ทิเคิลสวอมออฟติไมเซชัน (Particle Swarm Optimisation) เพื่อจัดตารางการผลิตในอุตสาหกรรม โดยจะศึกษาเปรียบเทียบทั้งในส่วนของคุณภาพของผลลัพธ์ที่ได้จากทั้งสองวิธี และเวลาที่ใช้ในการค้นหาคำตอบ โดยจะต้องทำการศึกษาการกำหนดค่าของปัจจัยที่เหมาะสมของทั้งสองวิธีก่อนที่จะทำการศึกษาเปรียบเทียบ

3. วัตถุประสงค์ของโครงการวิจัย

เพื่อศึกษาโครงสร้างการทำงานที่เหมาะสมของวิธีจีเนติกอัลกอริทึม และวิธีพาร์ทิเคิลสวอมออฟติไมเซชัน และศึกษาเปรียบเทียบด้านประสิทธิภาพการทำงานทั้งในส่วนของคุณภาพของผลลัพธ์ที่ได้จากทั้งสองวิธีและเวลาที่ใช้ในการค้นหาคำตอบ เพื่อจัดตารางการผลิตในอุตสาหกรรม

4. ประโยชน์ที่คาดว่าจะได้รับจากโครงการ

ได้ผลการวิจัยที่เป็นองค์ความรู้ใหม่ในเรื่องของการกำหนดค่าให้กับตัวแปรและกลไกพร้อมเปรียบเทียบประสิทธิภาพของทั้งสองวิธี โดยผลการวิจัยจะสามารถตีพิมพ์ในวารสารระดับนานาชาติหรือระดับชาติที่เป็นที่ยอมรับได้อย่างน้อย 1 เรื่อง อีกทั้งผู้วิจัยยังได้โปรแกรมช่วยในการจัดตารางการผลิต ได้ความรู้ ประสบการณ์ในด้านการวิจัยมากขึ้น เพื่อประโยชน์ต่อการเรียนการสอนและการวิจัยอย่างต่อเนื่องต่อไป

5. ขอบเขตของการวิจัย

ในส่วนของโปรแกรมที่จะพัฒนา ผู้วิจัยจะใช้ภาษาคอมพิวเตอร์ที่ชื่อว่า ทีซีแอล (TCL) (Ousterhout, 1994) ซึ่งเป็นภาษาใหม่ที่รองรับต่อการเขียนโปรแกรมทุกชนิดเช่นเดียวกับ C หรือ Pascal เป็นต้น แต่ TCL สามารถทำงานได้บนหลากหลายระบบปฏิบัติการ ง่ายต่อการเรียนรู้และใช้งานโดยเฉพาะอย่างยิ่งในส่วนของ Graphic User Interface มีเสถียรภาพสูง สามารถทำงานร่วมกับโปรแกรมอื่นได้ อีกทั้งยังเป็นฟรีโปรแกรม สามารถหาได้จาก Internet ไม่ต้องเสียค่าใช้จ่าย

โปรแกรมที่จะถูกพัฒนาขึ้นมานั้น จะถูกตรวจสอบการทำงานและการคำนวณในทุกขั้นตอน แล้วจึงนำไปทดสอบแก้ปัญหาการจัดตารางการผลิตโดยใช้ข้อมูลจริงจากโรงงานอุตสาหกรรมซึ่งผลิตสินค้าหรือสร้างสินค้าตามสั่ง ซึ่งผลิตภัณฑ์ดังกล่าวมีโครงสร้างของผลิตภัณฑ์ค่อนข้างซับซ้อน ดังนั้นปัญหาที่ทำการศึกษานั้นเป็นปัญหาการจัดตารางการทำงานของแต่ละขั้นตอนและชิ้นส่วนของผลิตภัณฑ์ ภายใต้สภาวการณ์ที่ต้องผลิตสินค้าหลายๆ ประเภทพร้อมๆ กัน แต่ละผลิตภัณฑ์มีโครงสร้างซับซ้อน การทำงานในแต่ละขั้นตอนมีเงื่อนไขก่อน-หลังเนื่องจากเหตุผลด้านการประกอบ โดยการผลิตในแต่ละขั้นตอนมีความต้องการใช้เครื่องจักรร่วมกัน จึงต้องมีการจัดสรรเวลาในการใช้เครื่องจักรที่จำกัด โดยโปรแกรมที่จะพัฒนาขึ้นมานี้ จะสามารถระบุและให้คำตอบได้ว่า ชิ้นส่วนหมายเลขใด ใช้เวลาผลิตเท่าใด บนเครื่องจักรใด ทำงานก่อนหน้าและตามหลังงานใด เพื่อใช้ประกอบในผลิตภัณฑ์ไหน ในช่วงเวลาใด เริ่มทำงานและเสร็จงานเมื่อใด โดยแสดงผลเป็นรูป Gantt Chart

โปรแกรมที่จะพัฒนาขึ้นมานี้จะรองรับระบบ Graphic User Interface และผู้ใช้สามารถที่จะกำหนดปัจจัยหรือตัวแปรที่เกี่ยวข้องต่อการจัดตารางการผลิตได้ และสามารถที่จะเลือกปัญหาการจัดตารางผลิตปัญหาใดก็ได้ขึ้นมาหาคำตอบ และบันทึกผลคำตอบที่ดีที่สุดลงในไฟล์เพื่อที่จะนำไฟล์ผลลัพธ์กลับมาใช้ใหม่ได้ โดยทดสอบแก้ปัญหาการจัดตารางการผลิตหลายๆ ขนาด

6. ระเบียบวิธีวิจัย

ขั้นตอนการวิจัยสามารถแบ่งเป็นขั้นตอนต่างๆ ดังนี้

1. ศึกษาโครงสร้างและขั้นตอนการทำงานอย่างละเอียดของวิธีจีเนติกอัลกอริทึมและวิธีพาร์ทิเคิลสวอมออฟติไมเซชัน พร้อมรวบรวมและออกแบบข้อมูลที่เกี่ยวข้องในการจัดตารางการผลิต
2. ออกแบบโครงสร้างของโครโมโซม โดยมีการเข้ารหัส เพื่อเก็บข้อมูลจำนวนผลิตภัณฑ์ ส่วนประกอบ และชิ้นส่วน เพื่อเป็น Primary key ในการเรียกใช้ข้อมูลที่เกี่ยวข้องในการผลิต
3. พัฒนาและทดสอบการทำงานของโปรแกรมจัดตารางการผลิตโดยจะต้องเลือกใช้ชุดข้อมูลในการจัดตารางได้ ซึ่งโปรแกรมดังกล่าวจะสามารถเลือกกำหนดค่าตัวแปรหรือปัจจัยต่างๆ ของทั้งสองวิธีได้

4. ออกแบบการทดลองและดำเนินการทดลอง พร้อมเก็บผลการทดลองเพื่อทำการวิเคราะห์ผลกระทบของการกำหนดค่าให้ตัวแปรหรือปัจจัยต่อประสิทธิภาพการทำงานของของทั้งสองวิธี
5. ออกแบบการทดลองและดำเนินการทดลอง พร้อมเก็บผลการทดลองเพื่อทำการเปรียบเทียบประสิทธิภาพการทำงานของของทั้งสองวิธี โดยใช้ผลการศึกษาในข้อที่ 4 มาใช้กำหนดปัจจัย
6. จัดทำรายงานสรุปผลการทำงานพร้อมเขียนบทความตีพิมพ์เผยแพร่

7. แผนการดำเนินงานตลอดโครงการวิจัย

แผนงานการดำเนินงาน	เดือนที่											
	1	2	3	4	5	6	7	8	9	10	11	12
1. ศึกษาขั้นตอนการทำงานอย่างละเอียดของวิธีจีเนติกและวิธีพาร์ทิเคิลสวอมออฟติไมเซชัน												
2. ออกแบบโครงสร้างของโปรแกรมพร้อมรวบรวมและบันทึกข้อมูล												
3. ออกแบบโครงสร้างของชุดคำตอบ และเข้ารหัสเพื่อเก็บข้อมูลจำนวนผลิตภัณฑ์												
4. พัฒนาและทดสอบการทำงานของโปรแกรม												
5. ทดลอง พร้อมวิเคราะห์ผลทดลองของการศึกษาเปรียบเทียบประสิทธิภาพ												
6. สรุปผลพร้อมเขียนบทความตีพิมพ์เผยแพร่												
7. พิมพ์รายงานฉบับสมบูรณ์ พร้อมถ่ายเอกสารเข้าปกจัดทำเป็นรูปเล่ม												

8. แผนการถ่ายทอดเทคโนโลยี หรือผลการวิจัยสู่กลุ่มเป้าหมาย

ผลการวิจัยจะสามารถเขียนเป็นบทความวิชาการเพื่อตีพิมพ์ในวารสารหรือที่ประชุมวิชาการระดับนานาชาติที่เป็นที่ยอมรับได้อย่างน้อย 1 เรื่อง

9. รายละเอียดเพิ่มเติมเกี่ยวกับโครงการที่ได้ดำเนินการไปแล้ว โดยให้แนบบทความผลงาน ความก้าวหน้าทางวิชาการของโครงการระหว่างดำเนินการที่เคยพิมพ์หรือเผยแพร่ในวารสารหรือนำเสนอในการประชุมทางวิชาการ (ถ้ามี)

ผลการดำเนินการของโครงการได้ถูกเขียนเป็นบทความวิจัยและได้เผยแพร่แบบบรรยาย (oral presentation) เรื่อง "Comparison of metaheuristics for solving multi-product multi-stage multi-machine scheduling problems" ในการประชุมทางวิชาการระดับนานาชาติ the 15th International Working Seminar on Production Economics ณ เมือง Innsbruck ประเทศ ออสเตรีย ในระหว่างวันที่ 3-7 มีนาคม พ.ศ. 2551 ซึ่งบทความดังกล่าวได้ถูกแนบไว้แล้วในภาคผนวก

10. คำชี้แจงเกี่ยวกับปัญหาและอุปสรรค (ถ้ามี)

-

11. ประวัติผู้วิจัย

ชื่อ (ภาษาไทย) นายภูพงษ์ พงษ์เจริญ

(ภาษาอังกฤษ) Mr. Pupong Pongcharoen

ตำแหน่งปัจจุบัน อาจารย์ระดับ 8

หน่วยงานที่อยู่ที่ติดต่อได้พร้อมโทรศัพท์และโทรสาร

ภาควิชาวิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ถนนพิษณุโลก-นครสวรรค์ อำเภอเมือง จังหวัดพิษณุโลก 65000

โทรศัพท์ (055) 261000-4 ต่อ 4006, 4201

โทรสาร (055) 261062

Email: Pupongp@yahoo.com และ Pupongp@nu.ac.th

ประวัติการศึกษา

จบปี พ.ศ.	วุฒิ	สถานศึกษา
2537	วิศวกรรมศาสตร์บัณฑิต (อุตสาหการ)	มหาวิทยาลัยเชียงใหม่
2539	Master of Engineering (ISE)	Asian Institute of Technology
2544	Doctor of Philosophy (IE)	University of Newcastle upon Tyne, UK

สาขาวิชาการที่มีความชำนาญพิเศษ (ระบุสาขาวิชาการ)

Operation Research, Computer Simulation, Manufacturing Planning, Project Management

ประสบการณ์ที่เกี่ยวข้องกับการบริหารงานวิจัยทั้งภายในและภายนอกประเทศ

ผู้อำนวยการแผนงานวิจัย: ชื่อแผนงานวิจัย -

หัวหน้าโครงการวิจัย: ชื่อโครงการวิจัย -

งานวิจัยที่ทำเสร็จแล้ว: ชื่อโครงการวิจัย ปีที่พิมพ์ การเผยแพร่ และสถานภาพในการทำวิจัย

งานวิจัยที่กำลังทำ : ชื่อแผนงานวิจัย -

ภาคผนวก

luct Roadmapping
'onnection - 7th

ıap Process,

Comparison of Metaheuristics for Solving Multi-product Multi-stage Multi-machine Scheduling Problems

Pupong Pongcharoen[1,*], Adirek Chainual[1], Aphirak Khadwilard[2], Chayanit Kaweesirikon[3] and Chris Hicks[4]

[1] Dept of Industrial Engineering, Faculty of Engineering, Naresuan University, Pitsanulok, Thailand 65000
[2] Faculty of Engineering, Rajamangala University of Technology Lanna, Tak Campus, Tak, Thailand 63000
[3] Faculty of Science, Naresuan University, Pitsanulok, Thailand 65000
[4] Business School, University of Newcastle upon Tyne, Newcastle upon Tyne, UK. NE1 7RU
* Corresponding author: pupongp@nu.ac.th and pupongp@yahoo.com

Abstract

Capital goods companies manufacture complex products with deep and complex product structures that give rise to many stages of assembly relationships. They schedule multiple products in a multi-stage, multi-machine system (MMMS). Feasible schedules must correctly sequence the operations required for manufacturing components and also satisfy assembly precedence relationships. This paper presents a comparative study of the application of Genetic Algorithms, Particle Swarm Optimisation and the Ant Colony System for solving MMMS production scheduling problems. The algorithms were designed to minimise the combination of earliness and tardiness penalties. Four different sized scheduling problems were obtained from a collaborating company that manufactures complex capital goods. Simulation experiments were designed to test and evaluate the performance of the metaheuristics for solving these problems. The performance of the algorithms was compared in terms of the minimum penalty costs obtained and the computational time required. The experimental results indicated that the efficiency and effectiveness of the algorithms was close particularly for small, medium and large problems. However, for the extra large problem, the Ant Colony System produced the lowest penalty cost but it required more computational time than the other algorithms.

Keywords: Product Scheduling, Metaheuristics, Genetic Algorithms, Particle Swarm, Ant Colony System.

1. Introduction

Capital goods companies produce capital products such as steam turbine generators, power station boilers and transformers, which have deep and complex product structures. Components often have complex routings with a large number of operations on many different machines. There are many dependency relationships due to product structure relationships. There are many resource constraints and the performance objectives can vary for the different product families. Scheduling is important because companies seek to deliver products on time and utilise their resources efficiently. Production scheduling in capital goods companies is difficult because: i) there is high product variety; ii) the products and processes are complex; and iii) the demand fluctuates considerably.

There is a comprehensive literature related to classical flow and job shop scheduling problems [1, 2, 3, 4]. These problems generally involve the scheduling of a set of independent tasks, which is known as single stage scheduling [5], which means that there is a single processing stage without assembly precedence constraints [6].

The literature relating to the scheduling of multiple products with deep and complex product structures produced by many resources with finite capacity constraints is limited. Reeja and

Rajendran [7] highlighted the lack of production scheduling research that has taken into account assembly relationships and operation precedence constraints. Fry et al. [8] recognized that there is a strong relationship between product structure and sequencing rule performance.

The use of metaheuristics to solve combinatorial optimisation problems has been reported by many researchers [9, 10, 11, 12]. However, most research has focused on using either a single method or a hybrid algorithm to solve classical flow or job shop scheduling problems. There has been no research that has systematically compared the performance of several different metaheuristic methods for solving MMMS / capital goods scheduling problems.

The objectives of this paper are to:

- propose a mathematical model that represents the multi-product multi-stage multi-machine scheduling (MMMS) problem that is constrained by part and operation precedence;
- describe three metaheuristics (Genetic Algorithm, Particle Swarm and the Ant Colony System) that have been developed to solve the MMMS problem;
- present a simulation experiment that tested and compared these methods using four representative problems obtained from a collaborating capital goods company.

The remaining sections of this paper are structured as follows. The next section of this paper proposes a mathematical model for multi-product multi-stage multi-machine scheduling (MMMS) problem. Section 3 describes the three metaheuristics that have been developed to solve the MMMS problem. These include Genetic Algorithms (GA), Particle Swarm Optimisation (PSO) and the Ant Colony System (ACS). Section 4 presents four representative problems that were used to test and evaluate the methods. The experiments are described and the results are presented and analysed. The conclusions are in section 5.

2. The Multi-product Multi-stage Multi-machine Scheduling (MMMS) Problem

Sequencing determines the order of tasks taking into account operation and assembly precedence constraints. Scheduling is defined as "the allocation of resources over time to perform a collection of tasks" [13]. A schedule specifies sequence and timing, normally expressed as a set of start and due times. Scheduling is a combinatorial optimisation problem that is classified as an NP hard problem [14], which means that the amount of computation required to find solutions increases exponentially with problem size.

Multi-product multi-stage multi-machine scheduling (MMMS) requires the allocation of resources over time to perform the operations necessary to produce components which need to be synchronised with subassembly and assembly processes. The process routings determine the operational precedence constraints, whilst the product structures determine the assembly constraints. Figure 1 Ashows an example of a typical product structure of a product produced by a collaborating capital goods company.

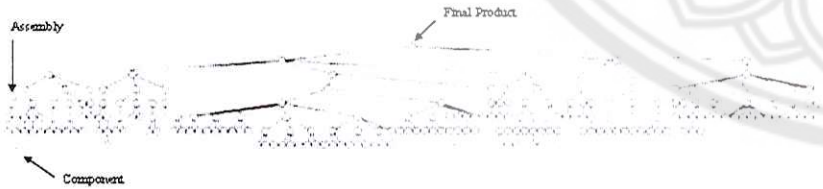

Figure 1 A typical product structure from a collaborating company [15].

The root node is the final product (P) which requires assemblies, subassemblies and components, which are the leaf nodes. Each component (C) requires a sequence of operations (O) to be performed on multiple machines (M).

Notation:

i	Operation i ($i = 1, \ldots, O$).
j	Component j ($j = 1, \ldots, C$).
k	Final product k ($k = 1, \ldots, P$).
m	Machine m ($m = 1, \ldots, M$).
R_m	Ready time for machine m (minutes).
C_k	Completion time of product k (minutes).
D_k	Due date of product k (minutes).
C_{jk}	Completion time of component j in product k (minutes).
D_{jk}	Due date of component j in product k (minutes).
E_k	Earliness of product k (minutes).
E_{jk}	Earliness of component j in product k (minutes).
T_k	Tardiness of product k (minutes).
SU_{ijkm}	Setup time for operation i on component j for product k on machine m (minutes).
ST_{ijkm}	Start time for operation i for component j in product k on machine m (minute).
PT_{ijkm}	Processing time for operation i for component j in product k on machine m (minutes).
FT_{ijkm}	Finishing time of operation i for component j in product k on machine m (minute).
TT_{ijkm}	Transfer time for operation i for component j in product k on machine m (minutes).
$X_{ijkabcm}$	1 if operation i for component j in product k precedes operation a for component b in product c on machine m; and 0 otherwise.
Pe	Earliness penalty (currency units per day).
Pt	Tardiness penalty (currency units per day).
$S(x)$	Set of child items for item x.
Sh	Length of working shifts (minutes).

The early completion of components, assemblies and products leads to stock holding costs, whilst the late delivery of products may incur a financial penalty that is specified in the contract with the customer. A common scheduling objective is to minimise these costs. The problem can be formulated mathematically as follows:

$$\text{Penalty cost} = \sum_{j=1}^{C}\sum_{k=1}^{P} Pe(E_{jk}) + \sum_{k=1}^{P} Pe(E_k) + \sum_{k=1}^{P} Pt(T_k) \quad (1)$$

Subject to:

$$ST_{ijkm} \geq R_m \qquad \forall i,j,k,m \qquad (2)$$

$$FT_{ijkm} = ST_{ijkm} + SU_{ijkm} + PT_{ijkm} + TT_{ijkm} \qquad \forall i,j,k,m \qquad (3)$$

$$C_{jk} \geq FT_{ijkm} \qquad \forall i,j,k,m \qquad (4)$$

$$E_{jk} = (D_{jk} - C_{jk})/Sh \text{ (when } D_{jk} > C_{jk} \text{ otherwise 0)} \qquad \forall j,k \qquad (5)$$

$$E_k = (D_k - C_k)/Sh \text{ (when } D_k > C_k \text{ otherwise 0)} \qquad \forall k \qquad (6)$$

$$T_k = (C_k - D_k)/Sh \text{ (when } C_k > D_k \text{, otherwise 0)} \quad \forall k \quad (7)$$

$$ST_{ixkm} - ST_{ijkm} \geq SU_{ijkm} + PT_{ijkm} + TT_{ijkm} \quad \forall i,k,m,j \in S(x) \quad (8)$$

$$ST_{gjkm} - ST_{ijkm} \geq SU_{ijkm} + PT_{ijkm} + TT_{ijkm} \quad \forall j,k,m,g=i+1 \quad (9)$$

$$X_{ijkabcm} + X_{abcijkm} = 1 \quad \forall a,b,c,i,j,k,m \quad (10)$$

$$X_{ijkabcm} \in (0, 1) \quad \forall a,b,c,i,j,k,m \quad (11)$$

$$E_{jk}, E_k, T_k \geq 0 \quad \forall j,k \quad (12)$$

$$ST_{ijkm}, R_m \geq 0 \quad \forall i,j,k,m \quad (13)$$

$$FT_{ijkm}, ST_{ijkm}, SU_{ijkm}, PT_{ijkm}, TT_{ijkm} \geq 0 \quad \forall i,j,k,m \quad (14)$$

The objective function (1) contains three parts: i) an earliness penalty cost for components; ii) an earliness penalty cost for final products; and iii) a tardiness penalty cost for products. The constraints in (2) ensure that the start time of any operations should be after the machine is ready. (3) specifies the finishing times for each operation in terms of the start, setup, machining and transfer times. (4) states that a component can't be completed until all the operations on the component are completed. (5), (6) and (7) calculate earliness and tardiness for components and products. (8) states that assemblies can't be started until all the items required are finished. For example, if part j is child item of part x, then the start time of the parent part (ST_{ixkm}) minus the start time of the child item (ST_{ijkm}) should be greater or equal to the sum of setup, processing and transfer times for the child item. Constraints (9) make sure that the operation precedence constraint within components is satisfied. For example, if a successor operation (g) is performed after its predecessor operation (i) is finished, the start time of the successor operation (ST_{gjkm}) minus the start time of the predecessor operation (ST_{ijkm}) should be greater or equal to the sum of setup, processing and transfer times of the predecessor operation. (10) ensures that only one operation can be performed on a machine at a time using the decision variables defined in constraints (11). Constraints (12)-(14) guarantee non-negative values for those defined variables.

Pongcharoen et al. [16] developed a Genetic Algorithms for solving the MMMS problem and reported that the schedules produced significantly outperformed the collaborating company's schedules. Pongcharoen et al. [17] investigated the significance of a repair process within the Genetic Algorithms and developed a systematic process for selecting Genetic Algorithms parameters that used a design of experiments approach and regression analysis. A limitation of this work was that it did not consider the systematic selection of genetic operators. This previous research related to the MMMS problem has not provided a mathematical formulation. Chen and Ji [18] applied the Mixed Integer Programming method to solve a single MMMS problem. They considered a small problem with two final products, with a total of 39 operations on 21 items (excluding final products). Each item required only one operation. Operation precedence constraints within the process routing for each item were therefore ignored. Setup and transfer times were also ignored in their mathematical model. Their approach was constrained by the limitations of the integer programming methods, which becomes prohibitively computationally expensive for large problems because it requires full enumerative search.

3. Metaheuristics for Multi-product Multi-stage Multi-machine Scheduling Problems

Optimisation algorithms can be categorised as being conventional or approximation optimisation algorithms [6, 19]. Conventional optimisation algorithms are usually based upon mathematical models such as Integer Linear Programming [18], Branch and Bound [20] or Dynamic Programming [21]. Approximation optimisation algorithms are based upon

constructive approaches (e.g. dispatching rules) and/or stochastic search techniques (so called metaheuristics). Metaheuristics including Genetic Algorithms (GA) [16], Particle Swarm Optimisation (PSO) [22] and Ant Colony System (ACS) [23] have been proposed as alternative ways for solving large-scale combinatorial optimisation problems. One of the most important issues for applying the metaheuristics is the balance between intensification (exploitation) of the search within the promising regions and diversification (exploration) which investigates other parts of the solution space.

3.1 The Metaheuristic Optimisation Scheduling Tool (MOST)

In this work, Genetic Algorithm, Particle Swarm Optimisation and an Ant Colony System were developed for solving the multi-stage multi-machine multi-product scheduling (MMMS) problem. The Metaheuristics optimisation scheduling tool (MOST) program includes a graphic user interface and together with the algorithms consists of more than 10,000 lines of code. It was written in a modular style using the Tcl/Tk programming language [24]. The architectural design of the MOST program (see Figure 2) includes three phases: i) *input*, in which product information and manufacturing data is entered. This includes part codes, product structures, part instance identifiers, resource information (e.g. machine number) and operational data (such as setup, machining and transfer times); ii) *scheduling*, where the embedded algorithms generate and evaluate schedules in accordance with precedence relationships and finite capacity constraints; and iii) *output*, which represents the best schedule as a Gantt chart and provides information on the schedule including the penalty cost. The graphic user interface (GUI) allows users to manipulate data, set parameters and specify the required outputs.

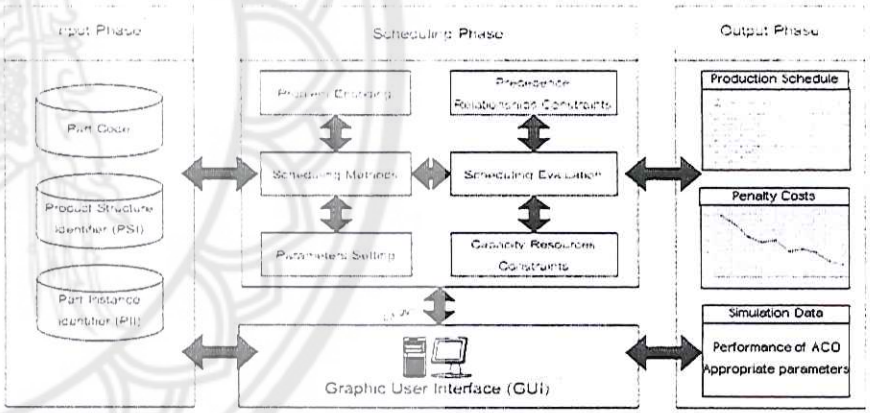

Figure 2 An architectural design for metaheuristics based scheduling program.

Table 1 Compares the processes and feature associated with the different metaheuristics.

Term or process	Particle Swarm Optimisation	Genetic Algorithms	Ant Colony System
Candidate solution	Particle	Chromosome	Ant
Repetitive search	Iterative repositioning	Reproductive generation	Iterative walking
Intensification process (exploitation process)	Repositioning based on social experience	Crossover operation	Global pheromone update
Diversification process (exploration process)	Repositioning based on it own experience	Mutation operation	Local pheromone update
Selection criteria	position	fitness	Pheromone

The next subsections briefly review the methods in more detail.

3.1.1 Particle Swarm Optimisation (PSO)

The Particle Swarm Optimisation (PSO) algorithm is an evolutionary computational technique that was inspired by the social behaviour of insects. The algorithm was originally designed and developed by Kennedy and Eberhart [25]. The evolutionary strategy within the PSO allows a particle (individual) to adjust its flight according to its flying experience and its companions' flying experience. The pseudo code of PSO is provided in Figure 3.

```
Pseudocode for a PSO procedure
For each particle
    Initialize particle
End For
Do
For each particle
Calculate fitness value
If the fitness value is better than the best fitness value (pBest) in history
Set current value as the new pBest
End
Choose the particle with the best fitness value of all the particles as the gBest
For each particle
    Calculate particle velocity according equation (a)
    Update particle position according equation (b)
End
    While maximum iterations or minimum error criteria is not attained
```

Figure 3 Pseudo code of Particle Swarm Optimisation [26].

The PSO method starts with a random initialisation of a group (swarm) of N individuals (particles). Each group is a candidate solution within the search space [26]. Each particle is treated as a point in d-dimensional problem space. The i^{th} particle is represented as $X_i = (x_{1d}, x_{2d}, \ldots, x_{Nd})$ where $i = 1, 2, 3, \ldots, N$. The position of each particle in the solution space is measured by using an objective function. The best position so far (the position giving the best fitness value) of the i^{th} particle is recorded and represented as $P_i = (p_{1d}, p_{2d}, \ldots, p_{Nd})$. Particles work iteratively on the basis of social behaviour within their group. They aim to find the global optimum by dynamically adjusting their position and velocity based upon their own experience (pbest) and the best experience within the swarm (gbest). The rate of the position change (velocity) for particle i^{th} is represented as v_{id}. The particles are manipulated according to equations (2) and (3).

$$v_{id} = w * v_{id} + c_1 * rand() * (p_{id} - x_{id}) + c_2 * Rand() * (p_{gd} - x_{id}) \quad (2)$$

$$x_{id} = x_{id} + v_{id} \quad (3)$$

The inertia weight, w, is introduced to alter the balance between a particle's own history and the group's experiences for repositioning. A high value of inertia weight facilitates global exploration (searching new areas) whilst a low inertia weight tends to assist local exploitation (fine tuning within the current search area). The cognition learning rate (c_1) and the social learning rate (c_2) are constant. *Rand()* and *rand()* are two random functions in the range [0,1] [26]. The whole process is repeated iteratively until a termination criterion is satisfied.

In this work the general form of the simple particle swarm optimisation was modified to make it suitable for representing the MMMSP. The particle swarm optimisation process includes particle encoding and initialisation, a repair process, performance evaluation, updating the best positions and particle repositioning.

Particle Encoding

Each operation in the problem was encoded as an alphanumeric string which had three parts: a product structure identifier; a product instance identifier; and an operation number. Figure 4 illustrates the product structure identifiers. The root node refers to the product, which in this case had a part number of 1. There were three subassemblies with part numbers 2, 3 and 4 and product structure identifiers 1:2, 1:3 and 1:4. Part number 2 was an assembly with two components (5 and 6) with the product structure identifiers 1:2:5 and 1:2:6. All the leaf nodes are components. This coding system uniquely defines the location of each type of part/assembly within the product structure.

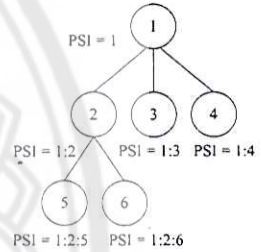

Figure 4 Product structure identifier (PSI).

However, in the general case it is possible for an assembly to include more than one instance of an item of the same type. Figure 5 represents the situation where the product with part code 1 contains two identical assemblies with the part code 2. The product structure identifiers are the same as in Figure 4. However, it is necessary to also consider the product instance identifiers that distinguish the different instances of identical parts.

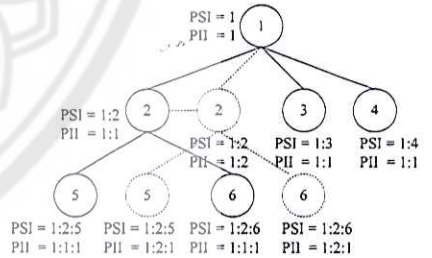

Figure 5 Coding scheme of product instance identifier (PII).

Particle Representation and Initialisation

Each particle consists of all the operations required for all the items within the scheduling problem. This includes all the operations on all the products, assemblies, subassemblies and components. Initially all the operations are randomly sequenced to initialise a particle (see

Figure 6 for an example). The initialisation process is repeated to generate a swarm (X_I) of particles of the desired size (N). The i^{th} particle in d-dimensional problem (x_{id}) represents a candidate solution (schedule) where $i = 1, 2, 3, \ldots, N$.

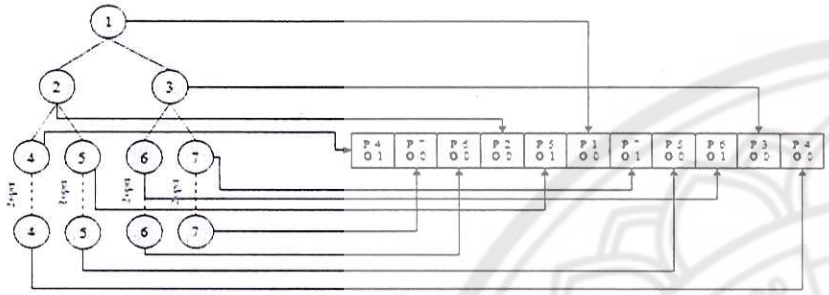

Figure 6 Particle representation.

Repair Process

The initialisation process within the PSO may produce infeasible particles (solutions). There are three ways to deal with infeasible solutions: i) discard them; ii) apply a high penalty in the objective function so that they are unlikely to survive; or iii) repair them [27]. Discarding infeasible solutions or applying a high penalty is only an option when a large proportion of the population is feasible. Repair processes are problem specific. In this work, a two stage repair process was used that included part and operation precedence adjustments.

Part Precedence Adjustment

In this stage, a particle may be rectified by considering part assembly constraints. For example, if there is a product with the product structure shown in Figure 6, there are four assembly routes from components to the final product (4-2-1, 5-2-1, 6-3-1 and 7-3-1). The process of checking and reordering parts considers each route in turn and take into account the assembly requirements as illustrated in Figure 7. It starts by copying parts (Part number 4, 2 and 1) belonging to the first assembly route to a temporary sequence. The operations required to make these parts are then copied to a temporary sequence (two operations on part 4 and one operation on parts 1 and 2). The copied parts are then reordered based upon their level within the product structure and then replaced back into the particle according to the new sequence. This process of part precedence adjustment is then repeated for the three remaining assembly routes within the product structure.

Operation Precedence Adjustment

The process of operation precedence adjustment considers only components that require more than one operation. Consider the same example with four components (part numbers 4, 5, 6 and 7), each of which requires two operations (see Figure 6). The process of checking and reordering operations is considered part by part. As shown in Figure 8, two operations (Operation number 0 and 1) for part number 4 are copied and the sequence of operations is checked. In this case the sequence of operations is in the wrong order in terms of the required process routing; so the sequence of operations is reordered by swapping the operations. The operations are then replaced back into the particle with the corrected sequence. This process is then repeated for those three remaining parts.

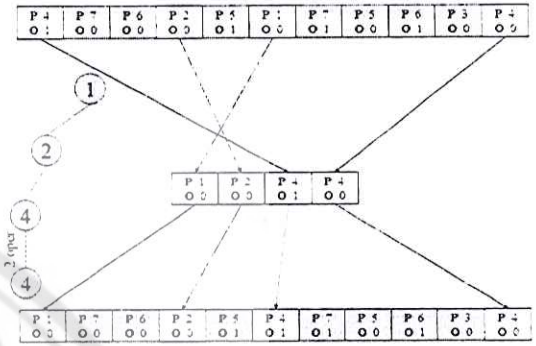

Figure 7 Checking and reordering based on part precedence constraints.

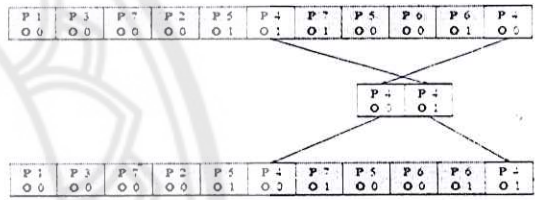

Figure 8 Checking and reordering based on operation precedence constraints.

Position evaluation

The position of each particle (x_{id}) in the d-dimensional solution space is measured by the objective (fitness) function given in Equation (1). Each schedule (particle) is then evaluated in terms of the penalty cost due to earliness and tardiness. A tardiness penalty rate per day (P_t) that is twice the earliness penalty rate (P_e) was used in this work.

Updating Best Positions

The best position so far (the position giving the best so far of fitness value) of the i^{th} particle (x_{id}) called "pbest" is recorded and represented as p_{id}. The best particle is called "*gbest*" and is recorded and represented as p_{gd}. Particles are processed iteratively on the basis of social behaviour within the group (swarm) that aims to find the global optimum by dynamically adjusting the particle's position and velocity based on its own experience (*pbest*) and the best experience of particle in the swarm (*gbest*).

Particle repositioning

The evolutionary strategy within the particle swarm optimisation allowed a particle (individual) to adjust its flying according to its own flying experience and its companions' flying experience. The particles were manipulated according to the equation (2) and (3) described in section 3. From equation (2), it can be seen that a particle repositioned itself from its current position (x_{id}) towards its own best experience (p_{id} - x_{id}) and the group's best experience (p_{gd} - x_{id}). For a continuous problem with a numerical solution space, the repositioning process is quite straightforward. However, with discrete problems, such as scheduling, the position of the particle represents the sequence of operations to be performed. For this reason, moving from one schedule (position) to another requires a swapping sequence. To overcome this difficulty, the swapping method introduced by Wang et al. [28] was adopted.

Suppose there is a sequencing problem with five operations (operation A, B, C, D and E). If a sequence solution, S = (D,E,C,A,B), has a swap operator applied between position i and j, $SO(i,j)$, the new solution, $S' = S + SO(i,j)$. The plus sign "+" can be explained as follows. If the solution (S) has a swap operation, $SO(1,2)$, then S' = (D,E,C,A,B) + $SO(1,2)$ = (E,D,C,A,B). A sequence may require a series of swap operators (SO) called a swap sequence, $SS = (SO_1, SO_2, SO_3, \ldots, SO_n)$. A new solution after the swap sequence would be $S' = S + SS$. In other words, the swap sequence transforms the current solution (S) to the new solution (S') or SS = S'-S. This swap sequence is adopted for repositioning from x_{id} to p_{gd} denoted as (p_{gd} - x_{id}). The learning rates (c_1 and c_2) and random numbers [*rand()* and *Rand()*] specify the probability of performing the swap sequence (SS). The particles with new positions are then evaluated and then *pbest* and *gbest* are updated. The whole process is repeated iteratively until a termination criterion is satisfied.

3.1.2 Genetic Algorithms (GA)

Genetic algorithms (GA) are based upon an analogy with biological evolution, in which the fitness of an individual determines its ability to survive and reproduce [29, 30]. GA have several advantages. GA deal with a coded problem instead of decision variables [31]. It does not require the problem to be expressed mathematically. It only requires an objective function for evaluating the fitness of chromosomes after they have undergone genetic operations. Genetic Algorithms use stochastic transition rules to guide the search [30]. GA perform multiple directional search using a set of candidate solutions, whereas most conventional methods conduct single directional search [29]. GA conduct both exploitation and exploration search whilst most conventional methods apply one or the other [29]. The pseudo code of GA is illustrated in Figure 9.

```
Pseuducode for a GA procedure
Begin:
    Generate random population of P solutions (Chromosomes):
        For each individual i ∈ P: calculate fitness (i):
        For i = 1 to number of generations:
            Randomly select an operation (crossover or mutation):
            If crossover:
                Select two parents at random i_a and i_b:
                Generate on offspring i_c = crossover (i_a and i_b):
            Else if mutation:
                Select one chromosome i at random:
                Generate on offspring i_c = mutation (i):
            End if:
            Calculate the fitness of the offspring i_c:
            If i_c is better than the worst chromosome then
            replace the worst chromosome by i_c:
        Next i:
    Check if termination = true:
End:
```

Figure 9 Pseudo code of Genetic Algorithms.

There have been a number of research articles related to GA and their applications in the fields of production and operations management (POM). Aytug et al. [10] provided a comprehensive review of 110 GA research articles published between 1996-2002. They

suggested that the setting of GA parameters and selection of operators in previous research had mainly used an ad hoc approach. A survey conducted by Chaudhry and Luo [11] stated that 103 of 178 GA related research articles had applied GA to solve scheduling or facility layout problems. However, most of the scheduling research articles related to classical job or flow shop problems that ignored assembly relationship and operation precedence [7].

The GA process within the MOST is almost similar to the PSO process except the evolution process is different. In other words, the processes for solution encoding, representation, initialisation and the repair of infeasible solutions (schedules) for PSO mentioned above were also applied in the GA method. The major dissimilarity is that the repositioning process was used to create new solutions for the PSO, whilst the GA applies crossover and mutation operations. The appropriate settings of the genetic parameters (including population size, number of generations and probabilities of crossover and mutation) and operators (crossover and mutation operations) have been thoroughly evaluated [32].

3.1.3 Ant Colony System (ACS)

The Ant Colony System (ACS) was initially proposed by Dorigo et al. [33] who was inspired by the foraging behaviour ants. They have poor vision and communication skills, which leads to a low probability of longevity. However, a large group of ants can systematically perform complex tasks with effectively by using pheromone (a chemical substance deposited by ants as they travel). With the ACS a colony of collaborative artificial ants is imitated with the objective of finding the shortest path between the nest and the source of food [34, 35]. The pseudo code of ACS is provided in Figure 10. The has been applied successfully to solve numerous combinatorial optimisation problems including the travelling salesman problem [33], the quadratic assignment problem [36], timetabling [37], job shop scheduling [38] and flow shop scheduling [39].

```
Pseudocode for an ACO procedure
Begin:
    Initialize the pheromone trails and parameters:
    Generate population of m solutions(ants):
    For each individual ant k ∈ m: calculate fitness (k):
    For each ant determine its best position:
    Determine the best global ant:
    Update the pheromone trail:
    Check if termination = true:
End :
```

Figure 10 Pseudo code of Ant Colony System.

The first algorithm based on the Ant Colony Optimisation (ACO) concept was called an Ant System (AS). It was developed to solve the travelling salesman problem [40]. The AS has been developed to include a number of extensions (e.g. elitist AS [40]; rank-based AS [41]; Ant Colony System [33] and Max-Min AS [42]) that aim to improve its performance. All the main procedures within the AS and its extensions are generally similar (see Dorigo et al. [35] for more details). The main differences in the methods relate to the details of pheromone updating and the management of pheromone trials. In this work, the ACO called Ant Colony System (ACS) was applied to solve scheduling the MMMS problem.

4. Experimental Design and Analysis

Four different industrial scheduling problems of varying size were considered (see Table 2). Data including production schedules, product structure relationships, process plans and resource loading information were obtained from a collaborating company engaged in the capital goods industry. The small problem involved two different products (245 and 451), with a combined requirement of twenty-five machining operations on eight resources with nine assembly operations. The details for the other problems are shown in Table 2.

Table 2 Industrial scheduling problems.

Problem size	Part number	Characteristics of Scheduling Problems				
		Number of Products	Number of Components	Machining / Assembly Operations	Number of Resources	Levels of product structure
Small	245 & 451	2	6	25/9	8	4
Medium	229 & 451	2	8	57/10	7	4
Large	4 & 228	2	12	118/17	17	4
Extra large	227	1	46	497/39	24	6

The metaheuristics' parameters considered in this work are summarised in Table 3. The values chosen were informed by previous research [43, 44, 45] which had systematically selected the values using a design of experiments approach [46]. A full factorial experimental design was developed to study the relative performance of the three metaheuristics. They were evaluated in terms of penalty cost and the required computational time. Each run was replicated five times. The results are summarised in Table 4. The penalty cost for earliness was £500 per day whilst the penalty cost for tardiness was £1,000/day.

Table 3 Parameter settings for each algorithm.

Algorithms	Parameters and its setting
Particle Swarm Optimisation (PSO)	Swarm size (N) = 50
	No. of Iterations (I) = 50
	Inertia weight (w) = 0.9
	Self learning rate (c_1) =0.9
	Social learning rate (c_2) =0.9
Genetic Algorithms (GA)	Population size = 50
	No. of generations = 50
	Probability of crossover = 0.9
	Probability of mutation = 0.1
Ant Colony System (ACS)	No. of ants (A) = 25
	No. of iterations (I) = 100
	Pheromone weight (α) = 0.01
	Heuristic information weight (β) = 2.5
	Pheromone evaporation weight (ρ) = 0.01

Table 4 shows that all the methods found the same result for the small problem. For the medium sized problem, both PSO and GA found the best minimum penalty cost at £8,000 but the mean penalty cost achieved by the GA was lower than PSO. For the large problem the GA performed best as it found the solution with the lowest penalty cost. It also had the lowest mean. However, the ACS outperformed both the GA and PSO for the extra large problem and it also had the lowest mean. This suggests that the performance of the proposed algorithms was broadly similar for the small, medium and large problems. However, for the extra large problem, the ACS was significantly better than GA and PSO by 26.74% and 27.13%, respectively. However, the ACS required at least three times more execution time than the GA or PSO on small sized problem. For extra large sized problem, the average execution time

taken by the ACS was thirty times longer than other algorithms. A personal computer with 2.0GHz Core 2 Duo processor and 1 GB RAM was used for all the experiments.

Table 4 Penalty costs and execution times related to schedules using the proposed algorithms

Algorithms		Problem Sizes			
		Small	Medium	Large	Extra Large
PSO	Average execution time (min.)	1.77	2.02	3.30	10.58
	Minimum penalty cost (£)	6,500	8,000	53,500	4,813,500
	Mean penalty cost (£)	6,500	8,600	55,800	5,060,000
	Standard deviation (£)	0.0	548	1,441	163,174
GA	Average execution time (min.)	0.23	0.53	1.29	5.41
	Minimum penalty cost (£)	6,500	8,000	51,000	4,788,000
	Mean penalty cost (£)	6,500	8,100	54,000	5,056,600
	Standard deviation (£)	0.0	224	1,658	181,976
ACS	Average execution time (min.)	1.70	5.50	19.67	355.55
	Minimum penalty cost (£)	6,500	9,000	59,500	3,507,500
	Mean penalty cost (£)	6,500	9,000	61,100	3,933,800
	Standard deviation (£)	0.0	0.0	1,517	349,634

5. Conclusions

Production scheduling is a problem faced by capital goods companies that manufacture complex products with deep and complex product structures that include many stages of assembly relationships. Feasible schedules must correctly sequence the operations required to produce components and assembly precedence constraints must also be satisfied.

This paper has presented a comparative study of the application of metaheuristics including Genetic Algorithms (GA), Particle Swarm Optimisation (PSO) and Ant Colony System (ACS) for solving multi-product, multi-stage, multi-machine scheduling problems. The algorithms aimed to minimise the combination of earliness and tardiness penalties. Small, medium, large and extra large scheduling problems were obtained from a collaborating company that manufactures complex capital goods. Simulation experiments were designed and conducted to investigate the performance of the algorithms on various problem sizes. It was found that the performances of the proposed algorithms were not dramatically different for the small, medium and large sized problems. For extra large problem, ACS was significantly better. The minimum penalty cost was 26.74% lower than the GA and 27.13% lower than the PSO. However, the average execution times taken by the ACS were much longer than GA and PSO especially for the extra large problem.

6. Acknowledgements

The first author would like to acknowledge the Faculty of Engineering, Naresuan University for financial support on the travel expenses and conference fee.

References

1. Garey, M., Johnson, D., Sethi, R., 1976. The complexity of flow shop and job shop scheduling. *Mathematics of Operations Research,* 1(2), 117-129.
2. Graham, R., Lawler, E., Lenstra, J., K.A., R., 1979. Optimisation and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics,* 5(287-326.
3. Reisman, A., Kumar, A., Motwani, J., 1994. Flowshop scheduling/sequencing research: A statistical review of the literature, 1952-1994. *IEEE Transactions on Engineering Management,* 44(3), 316-329.

สำนักหอสมุด 2 มี.ค. 2554

4. Ruiz, R., Maroto, C., 2005. A comprehensive review and evaluation of permutation flow shop heuristics. *European Journal of Operational Research,* 165, 479-494.
5. He, Y., Hui, C.W., 2007. Genetic algorithm based on heuristic rules for high-constrained large-size single-stage multi-product scheduling with parallel units. *Chemical Engineering and Processing,* 46(11), 1175-1191.
6. Blazewicz, J., Domschke, W., Pesch, E., 1996. The job shop scheduling problem: Conventional and new solution techniques. *European Journal of Operational Research,* 93(1), 1-33.
7. Reeja, M.K., Rajendran, C., 2000. Dispatching rules for scheduling in assembly job shops - part I. *International Journal of Production Research,* 38(9), 2051-2066.
8. Fry, T.D., Oliff, M.D., Minor, E.D., Leong, G.K., 1989. The effects of product structure and sequencing rule on assembly shop performance. *International Journal of Production Research,* 27, 671-686.
9. Alrashidi, M.R., El-Hawary, M.E., 2006. A survey of particle swarm optimization applications in power system operations. *Electric Power Components and Systems,* 34(12), 1349-1357.
10. Aytug, H., Knouja, M., Vergara, F.E., 2003. Use of genetic algorithms to solve production and operations management problems: A review. *International Journal of Production Research,* 41(17), 3955–4009.
11. Chaudhry, S.S., Luo, W., 2005. Application of genetic algorithms in production and operations management: A review. *International Journal of Production Research,* 43(19), 4083-4101.
12. Dorigo, M., Blum, C., 2005. Ant colony optimization theory: A survey. *Theoretical Computer Science,* 344(2-3), 243-278.
13. Baker, K.R., 1974. *Introduction to sequencing and scheduling.* New York: Wiley and Sons.
14. King, J.R., Spackis, A.S., 1980. Scheduling: Bibliography and review. *International Journal of Physical Distribution and Materials Management,* 10, 105-132.
15. Hicks, C., Pongcharoen, P., 2006. Dispatching rules for production scheduling in the capital goods industry. *International Journal of Production Economics,* 104(1), 154-163.
16. Pongcharoen, P., Hicks, C., Braiden, P.M., 2004. The development of genetic algorithms for the finite capacity scheduling of complex products, with multiple levels of product structure. *European Journal of Operational Research,* 152(1), 215-225.
17. Pongcharoen, P., Hicks, C., Braiden, P.M., Stewardson, D.J., 2002. Determining optimum genetic algorithm parameters for scheduling the manufacturing and assembly of complex products. *International Journal of Production Economics,* 78(3), 311-322.
18. Chen, K., Ji, P., 2007. A mixed integer programming model for advanced planning and scheduling (aps). *European Journal of Operational Research,* 181(1), 515-522.
19. Nagar, A., Haddock, J., Heragu, S., 1995. Multiple and bicriteria scheduling: A literature survey. *European Journal of Operational Research,* 81(1), 88-104.
20. Brucker, P., Knust, S., Schoo, A., Thiele, O., 1998. A branch and bound algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research,* 107(2), 272-288.
21. Choi, J., Realff, M.J., Lee, J.H., 2004. Dynamic programming in a heuristically confined state space: A stochastic resource-constrained project scheduling application. *Computers & Chemical Engineering,* 28(6-7), 1039-1058.
22. Zhang, H., Li, H., Tam, C.M., 2006. Particle swarm optimization for resource-constrained project scheduling. *International Journal of Project Management,* 24(1), 83-92.
23. Holthaus, O., Rajendran, C., 2005. A fast ant-colony algorithm for single-machine scheduling to minimize the sum of weighted tardiness of jobs. *Journal of the Operational Research Society,* 56(8), 947-953.

24. Ousterhout, J.K., 1994. *Tcl and the tk toolkit.* Massachusetts, USA: Addison-Wesley.
25. Kennedy, J., Eberhart, R.C., 1995. Particle swarm optimization, *the IEEE International Conference on Neural Networks,* IEEE Service Center, 1942-1948.
26. Kennedy, J., Eberhart, R.C., 2001. *Swarm intelligence.* San Francisco, CA: Morgan Kaufmann Publishers.
27. Blum, C., Roli, A., 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing surveys,* 35(3), 268-308.
28. Wang, A.P., Huang, L., Zhou, C.G., Pang, W., 2003. Particle swarm optimisation for travelling salesman problem, *the second international conference on machine learning and cybernetics,*
29. Gen, M., Cheng, R., 1997. *Genetic algorithms and engineering design.* New York: John Wiley and Sons.
30. Goldberg, D.E., 1989. *Genetic algorithms in search, optimisation and machine learning.* Massachusetts: Addison-Wesley.
31. Syarif, A., Yun, Y., Gen, M., 2002. Study on multi-stage logistic chain network: A spanning tree-based genetic algorithm approach. *Computers & Industrial Engineering,* 43(1-2), 299-314.
32. Pongcharoen, P., Stewardson, D.J., Hicks, C., Braiden, P.M., 2001. Applying designed experiments to optimize the performance of genetic algorithms used for scheduling complex products in the capital goods industry. *Journal of Applied Statistic,* 28(3-4), 441-455.
33. Dorigo, M., Gambardella, L.M., 1997. Ant colonies for the travelling salesman problem. *Biosystems,* 43(2), 73-81.
34. Dorigo, M., Maniezzo, V., Colorni, A., 1996. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B,* 26, 1-33.
35. Dorigo, M., Stutzle, T., 2004. *Ant colony optimization.* Massachusetts: Bradford Book.
36. Maniezzo, V., Colorni, A., 1999. The ant system applied to the quadratic assignment problem. *IEEE Transactions on Data and Knowledge Engineering,* 11(5), 769-778.
37. Socha, K., Knowles, J., Sampels, M., 2002. A max-min ant system for the university course timetabling problem. In: Dorigo, M., Dicaro, G., Sampels, M.(Eds). *A max-min ant system for the university course timetabling problem.* 2463, 1-13. Springer-Verlag.
38. Colorni, A., Dorigo, M., Maniezzo, V., Tubian, M., 1994. Ant system for job-shop scheduling. *Belgian Journal of Operations Research, Statistics and Computer Science,* 34(1), 39-53.
39. Stutzle, T., 1998. An ant approach to the flow shop problem, *Proceedings of the sixth European Congress on Intelligent Techniques and Soft Computing,* Verlag Mainz, Wissenschaftsverlag, 1560-1564.
40. Dorigo, M., Maniezzo, V., Colorni, A., 1991. *Positive feedback as a search strategy.* Technical Report 91-016, Politecnico di Milano, Milan.
41. Bullnheimer, B., Hartl, R.F., Strauss, C., 1997. *A new rank based version of the ant system - a computational study.* Technical report Institute of Management Science, University of Vienna, Austria.
42. Stutzle, T., Hoos, H.H., 2000. Max-min ant system. *Future Generation Computer Systems,* 16(8), 889-914.
43. Chainual, A., Lutuksin, T., Pongcharoen, P., 2007. Computer based scheduling tool for multi-product scheduling problems. *International Journal of the Computer, the Internet and Management,* Article in press.
44. Kaweesirikon, C., "Particle swarm optimisation for production scheduling," Master degree, Naresuan University, Pitsanulok, 2006.

45. Khadwilard, A., Pongcharoen, P., 2006. Investigation of genetic parameters on different sizes of production scheduling problem, *the 7th Asian Pacific Industrial Engineering and Management Systems*, CD-ROM format.
46. Montgomery, D.C., 2001. *Design and analysis of experiments*. NY: John Wiley and Sons.