รายงานวิจัยฉบับสมบูรณ์

การพัฒนาโปรแกรมช่วยแก้ปัญหาการจัดเรียงกล่องผลิตภัณฑ์ลงในตู้สินค้าด้วยวิธีการระบบภูมิคุ้มกันเสมือน

โดย

ผศ.ดร. ภูพงษ์ พงษ์เจริญ

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

สนับสนุนโดยกองทุนวิจัยมหาวิทยาลัยนเรศวร

บทคัดย่อ

โลจิสติกส์ทางทะเลถือว่ามีความสำคัญและมีผลกระทบทางตรงจากปริมาณการค้าขายระหว่างประเทศที่เพิ่มมากขึ้นในปัจจุบัน กล่องบรรจุภัณฑ์ของสินค้าแต่ละบริษัทมักจะมีขนาดแตกต่างกัน ซึ่งจะถูกจัดเรียงเข้าไปในตู้ขนส่งสินค้า (Shipping container) ก่อนจะถูกขนย้ายขึ้นเรือบรรทุกสินค้าเพื่อจัดส่งผ่านท่าเรือไปยังประเทศปลายทางต่อไป บริษัทที่ให้บริการด้านการขนส่งทางทะเลต้องบริหารจัดการตู้ขนส่งสินค้าอย่างมีประสิทธิภาพ กล่าวคือต้องพยายามให้มีการใช้พื้นที่ที่มีอยู่อย่างจำกัดภายในตู้สินค้าแต่ละขนาดให้เต็มประสิทธิภาพ ผ่านการจัดเรียงกล่องบรรจุภัณฑ์เข้าไปในตู้สินค้าอย่างเหมาะสม ซึ่งปัญหาการจัดเรียงกล่องผลิตภัณฑ์เข้าไปในตู้สินค้า (Container packing problem) นั้นจัดว่าเป็นปัญหาที่ซับซ้อนและยากต่อการค้นหาคำตอบที่เหมาะสมที่สุด (Optimum solution) จึงมีความจำเป็นที่จะต้องมีโปรแกรมช่วยในการแก้ปัญหาการจัดเรียงกล่องผลิตภัณฑ์เข้าไปในตู้สินค้า (Container packing problem)

โครงการวิจัยนี้จึงได้พัฒนาโปรแกรมที่การประยุกต์ใช้วิธีการระบบภูมิคุ้มกันเสมือน (Artificial Immune System) วิธีพาร์ทิเคิลสวอมออฟติไมเซชั่น (Particle Swarm Optimisation) และวิธีจีเนติกอัลกอริทึม (Genetic algorithms) เพื่อช่วยแก้ปัญหาการจัดเรียงกล่องผลิตภัณฑ์เข้าไปในตู้สินค้า (Container packing problem) โดยโปรแกรมดังกล่าวถูกสร้างขึ้นจากภาษาคอมพิวเตอร์ Visual basic และถูกใช้ทดสอบแก้ปัญหาการจัดเรียงกล่อง ตั้งแต่จำนวน 100 ถึง 5,000 กล่องบรรจุภัณฑ์ที่มีขนาดแตกต่างกันทั้งหมด เพื่อทำการศึกษาผลกระทบจากการกำหนดค่าของปัจจัยที่เกี่ยวข้องกับการใช้วิธีการในการค้นหาคำตอบที่กล่าวไว้ข้างต้น โดยทำการวิเคราะห์ผลกระทบด้วยวิธีการทางสถิติวิเคราะห์ ด้วยช่วงความเชื่อมั่นที่ 95% นอกจากนี้ยังได้ทำการทดลองเปรียบเทียบเชิงประสิทธิภาพของการค้นหาคำตอบของการจัดเรียงกล่องในตู้สินค้า จากการใช้วิธีการทั้ง 3 วิธีที่กล่าวไว้ข้างต้นด้วย ทั้งนี้พบว่าคำตอบที่ได้จากวิธีการระบบภูมิคุ้มกันเสมือน ให้ค่าคำตอบที่ดีกว่าคำตอบที่ได้จากการใช้วิธีพาร์ทิเคิลสวอมออฟติไมเซชั่น และวิธีจีเนติกอัลกอริทึม อย่างไรก็ตาม จากการทดลองพบว่าการใช้วิธีการระบบภูมิคุ้มกันเสมือน ต้องการเวลาในการคำนวณเพื่อค้นหาคำตอบนานกว่าวิธีพาร์ทิเคิลสวอมออฟติไมเซชั่น และวิธีจีเนติกอัลกอริทึม

ทั้งนี้ในระหว่างการดำเนินโครงการวิจัย ผลงานวิจัยบางส่วนโดยเฉพาะประเด็นผลการวิจัยเชิงลึกในด้านทฤษฎีต่างๆ (เช่น แนวการประยุกต์ใช้แก้ปัญา ค่าปัจจัยและกลไกที่มีผลกระทบต่อประสิทธิภาพการทำงานของวิธีการต่างๆ ดังที่กล่าวไว้ข้างต้น) ได้ถูกเขียนเป็นบทความวิจัย และได้เผยแพร่ในวารสารระดับนานาชาติจำนวน 1 บทความ ทั้งนี้ได้แนบบทความดังกล่าว ในภาคผนวกของรายงานฉบับนี้แล้ว

ABSTRACT

Marine logistics has become increasingly important as the amount of global trade has increased. Products are usually packed in various sizes of boxes, which are then arranged into containers before shipping. Shipping companies aim to optimise the use of space when packing heterogeneous boxes into containers. The container packing problem (CPP) aims to optimise the packing of a number of rectangular boxes into a set of containers. The problems may be classified as being homogeneous (identical boxes); weakly heterogeneous (a few different sizes); or strongly heterogeneous (many different boxes). The CPP is categorised as an NP hard problem, which means that the amount of computation required to find solutions increases exponentially with problem size.

This research project was aimed to develop a programming tool that applied three metaheuristics called Artificial Immune System (AIS), Particle Swarm Optimisation (PSO) and Genetic Algorithm (GA) for solving multiple container packing problems (MCPP). The stochastic optimisation tool was written in Microsoft Visual basic. A sequential series of experiments was designed to identify the best parameter settings and configuration of the algorithms for solving MCPP. The work optimised the packing of a standard marine container for a strongly heterogeneous problem. The experimental results were analysed using the general linear model form of analysis of variance to identify appropriate algorithm configuration and parameter settings. It was found that each algorithm's parameters were statistically significant with a 95% confidence interval. The best configurations were then used in a sequential experiment that compared the performance of the AIS, PSO and GA algorithms for solving twenty-one heterogeneous MCPP. It was found that the average best-so-far solutions obtained from the AIS were marginally better than those produced by the GA and PSO for all problem sizes but the AIS required longer computational time than GA.

During the research project, parts of this research have been written as a research article, which has been published in the International Journal of Production Economics (Impact factor in 2010 = 1.988) (see more details of the article in the appendix).

ประกาศคุณูปการ
(Acknowledgements)

ผู้วิจัยขอขอบคุณ ทุนอุดหนุนการวิจัยจากงบประมาณแผ่นดิน มหาวิทยาลัยนเรศวร ที่ให้การสนับสนุนทุนในการดำเนินโครงการวิจัยนี้ และขอขอบคุณนิสิตช่วยงานวิจัยทุกคน ที่มีส่วนช่วยให้การดำเนินงานวิจัยในโครงการนี้ อีกทั้งขอขอบคุณ ภาควิชาวิศวกรรมอุตสาหการ และคณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร ที่ได้ให้ความสะดวกด้านสถานที่ และห้องปฏิบัติการคอมพิวเตอร์เพื่อการทำวิจัยและการทดลองที่เกี่ยวข้องกับการประมวลผลด้านการคำนวณด้วยคอมพิวเตอร์ความเร็วสูง จนทำให้การปฏิบัติงานในขั้นตอนต่างๆ ของโครงการวิจัยนี้สำเร็จลุล่วงไปได้ด้วยดี

สุดท้ายนี้ขอขอบคุณ เจ้าหน้าที่ของ กองบริหารการวิจัย (DRA) มหาวิทยาลัยนเรศวร ทุกคน ที่ให้ความสะดวกในด้านข้อมูล เอกสารแบบฟอร์มต่างๆ ที่ต้องใช้ประกอบในการดำเนินโครงการวิจัย จนทำให้งานเอกสารที่เกี่ยวข้องกับงานวิจัยนี้ดำเนินการไปได้ด้วยดี

ผู้ช่วยศาสตราจารย์ ดร. ภูพงษ์ พงษ์เจริญ
วันที่ 30 พฤศจิกายน พ.ศ. 2555

สารบัญ

(List of Contents)

หน้า

ข้อสรุปโครงการ
(Executive Summary)

1 ข้อมูลโครงการ

ชื่อโครงการ (ภาษาไทย)	การพัฒนาโปรแกรมช่วยแก้ปัญหาการจัดเรียงกล่องผลิตภัณฑ์ลงในตู้สินค้าด้วยวิธีการระบบภูมิคุ้มกันเสมือน
(ภาษาอังกฤษ)	Software Development for Solving Container Packing Problem Using Artificial Immune Systems

ชื่อหัวหน้าโครงการ	นายภูพงษ์ พงษ์เจริญ
ที่อยู่	คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อำเภอเมือง จังหวัดพิษณุโลก 65000
โทรศัพท์	0-5596-4201
โทรสาร	0-5596-4003
E-mail	Pupongp@nu.ac.th or Pupongp@gmail.com

สาขาวิชาที่ทำการวิจัย	สาขาวิชาวิศวกรรมอุตสาหการ
งบประมาณทั้งโครงการ	134,000 บาท
ระยะเวลาดำเนินงาน	2 ปี 0 เดือน (ตั้งแต่วันที่ 1 ธันวาคม 2553 ถึงวันที่ 30 มกราคม 2555)

2 เนื้อหางานวิจัย

จากแนวโน้มการค้าระหว่างประเทศที่เพิ่มมากขึ้นอันเนื่องมาจากกระแสโลกาภิวัตน์ สินค้าหรือผลิตภัณฑ์ที่มักจะผลิตในประเทศที่มีค่าจ้างแรงงานต่ำๆ เพื่อประโยชน์ทางการค้า แล้วจึงจัดส่งผลิตภัณฑ์สำเร็จรูปนั้นไปจำหน่ายในอีกหลายประเทศ โดยรูปแบบการขนส่งนั้น อาจจะพบได้ทั้งในทางอากาศ (เช่น เครื่องบินเป็นต้น) ทางบก (เช่นรถบรรทุก หรือรถไฟ) หรือทางน้ำ (ทางเรือ) การขนส่งสินค้าระหว่างประเทศส่วนใหญ่ พบว่านิยมขนส่งกล่องผลิตภัณฑ์ลงในตู้สินค้าหรือตู้คอนเทนเนอร์ (Container) บรรทุกบนเรือเดินสมุทร เนื่องจาก หลายประเทศเป็นเกาะ การขนส่งไม่สามารถกระทำได้ทางบก อีกทั้งยังสามารถขนส่งได้ในปริมาณมาก ต้นทุนค่าขนส่งต่ำ แต่อาจจะใช้เวลานานกว่าแบบอื่น ดังนั้นในหลายประเทศรวมทั้งประเทศไทย จึงมีท่าเรือมากกว่า 1 แห่งเพื่อรองรับต่อการเจริญเติบโตและการขยายตัวทางเศรษฐกิจการค้าระหว่างประเทศ จากสถิติปริมาณการขนส่งตู้สินค้าในปี 2547 แจ้งไว้บนเว็บไซต์ของการท่าเรือกรุงเทพฯ (http://www.bkp.port.co.th) พบว่าปริมาณตู้สินค้าที่ขนส่งเข้าและออก ผ่านท่าเรือกรุงเทพฯแห่งเดียว มีจำนวนรวมทั้งสิ้น 1,318,403 ตู้ ดังนั้นการใช้พื้นที่ภายในตู้คอนเทนเนอร์อย่างมีประสิทธิผล ย่อมส่งผลโดยตรงต่อประสิทธิภาพในการขนส่งโดยรวม เนื่องจากการใช้ตู้สินค้าลดน้อยลงแต่ยังคงความสามารถในการขนถ่ายกล่องผลิตภัณฑ์ได้ปริมาณเดิม จะมีส่วนช่วยในการลดค่าเช่า ค่าระวาง และค่าใช้จ่ายอื่นๆ อันเกี่ยวเนื่องกับตู้คอนเทนเนอร์ได้อีกด้วย

การใช้พื้นที่ภายในตู้คอนเทนเนอร์อย่างมีประสิทธิภาพนั้น เป็นความต้องการที่ผู้ประกอบการให้บริการการขนส่ง (Service Provider) ให้ความสำคัญมาก แต่ปัญหาการจัดเรียงกล่องผลิตภัณฑ์ลงในตู้สินค้า (Container Packing Problem: CPP) ซึ่งสามารถกระทำได้โดยพยายามลดพื้นที่ว่างเปล่าและเหลือใช้ภายในตู้สินค้าให้น้อยที่สุดนั้นไม่ใช่เรื่องง่าย โดยเฉพาะอย่างยิ่งเมื่อมีจำนวนกล่องผลิตภัณฑ์มากๆ เนื่องจากจำนวนคำตอบ (รูปแบบการจัดเรียง) ทั้งหมดที่สามารถเป็นไปได้ ขึ้นอยู่กับจำนวนของกล่องที่จะต้องทำการจัดเรียง กล่าวคือเช่น ถ้ามีกล่องสินค้าที่จะทำการขนส่งจำนวน 10 กล่อง วิธีในการจัดเรียงกล่องลงในตู้คอนเทนเนอร์โดยมีลำดับการจัดเรียงกล่องใดเข้าไปก่อน แล้วกล่องใดเข้าไปที่หลังจะมีจำนวนวิธีในการจัดเรียงที่เป็นไปได้ถึง 10! หรือ 3,628,800 คำตอบ ซึ่งจำนวนคำตอบดังกล่าว อยู่บนสมมติฐานว่า กล่องที่จะนำมาจัดเรียงนั้น ไม่สามารถพลิกหรือหมุนได้ แต่ถ้าไม่มีสมมติฐานดังกล่าว กล่องแต่ละใบสามารถที่จะพลิกหมุนได้ ซึ่งสามารถทำได้ 6 แบบ (Bortfeldt and Gehring, 2001) ดังนั้นวิธีการจัดเรียงกล่อง 10 กล่องที่เป็นไปได้จะเพิ่มขึ้นเป็น $6n \times n!$ หรือ 219.4 ล้านล้านคำตอบ ฉะนั้นปัญหานี้จึงเป็นปัญหาที่ท้าทายเป็นอย่างยิ่งและจัดเป็นปัญหาที่มีความซับซ้อน อยากต่อการแก้ปัญหาด้วยมือ (Manual) จำเป็นต้องใช้เครื่องคอมพิวเตอร์เข้ามาช่วยในการคำนวณเพื่อความรวดเร็วและความแม่นยำถูกต้อง

ปัญหาการจัดเรียงกล่องผลิตภัณฑ์ลงในตู้สินค้า (Container Packing Problem: CPP) เป็นปัญหาประเภท NP-hard (Garey and Johnson, 1979 and Evan et al., 1976) กล่าวคือ การหาจำนวนทางเลือกหรือจำนวนคำตอบที่เป็นไปได้มีทางเลือก (Candidate Solutions) เป็นจำนวนมาก โดยเฉพาะปัญหาขนาดใหญ่ ดังนั้นการแก้ปัญหาด้วยวิธีทางคณิตศาสตร์จึงเป็นไปได้ยากมาก อีกทั้งยังใช้เวลาในการคำนวณนานมาก และอาจจะพบปัญหาเรื่องหน่วยความจำได้ (Nagar et al., 1995) เทคนิคใหม่ๆ อย่างวิธีระบบภูมิคุ้มกันเสมือน (Artificial Immune Systems) ซึ่งมีวิธีการค้นหาคำตอบแบบสุ่ม (Stochastic Search Methods) โดยอาศัยหลักการระบบภูมิคุ้มกันของสิ่งมีชีวิต (Biological Immune Systems) จึงเป็นอีกแนวทางหนึ่งที่ท้าทายต่อการนำมาใช้แก้ปัญหาการจัดเรียงกล่องผลิตภัณฑ์ลงในตู้สินค้า

ดังนั้นงานวิจัยนี้จึงมีแนวทางที่จะพัฒนาโปรแกรมช่วยในการจัดเรียงกล่องผลิตภัณฑ์ลงในตู้สินค้า โดยโปรแกรมดังกล่าวผู้ใช้งานสามารถกำหนดวิธีการที่จะใช้ในการค้นหาคำตอบปัญหาการจัดเรียงกล่องผลิตภัณฑ์ลงในตู้สินค้า ซึ่งวิธีการที่นำมาประยุกต์เพื่อแก้ปัญหาการจัดเรียงคือ วิธีระบบภูมิคุ้มกันเสมือน (Artificial Immune Systems: AIS) อีกทั้งจะได้เปรียบเทียบประสิทธิภาพการค้นหาคำตอบเมื่อเทียบกับวิธีการที่ได้รับความนิยมสูงสุดอีกวิธีหนึ่งคือ วิธีจีเนติกอัลกอริทึม (Genetic Algorithms: GA)

3 วัตถุประสงค์

พัฒนาโปรแกรมช่วยในการจัดเรียงกล่องผลิตภัณฑ์ลงในตู้สินค้า เพื่อศึกษาเปรียบเทียบประสิทธิภาพการทำงานของวิธีจีเนติกอัลกอริทึม (GA) และวิธีการระบบภูมิคุ้มกันเสมือน (AIS) ในการจัดเรียงกล่องผลิตภัณฑ์ลงในตู้สินค้า โดยจะศึกษาเปรียบเทียบทั้งในส่วนของคุณภาพของผลลัพธ์ที่ได้จากทั้งสองวิธีและเวลาที่ใช้ในการค้นหาคำตอบ โดยจะต้องทำการศึกษาการกำหนดค่าของปัจจัยที่เหมาะสมของทั้งสองวิธีก่อน

4 ขอบเขตของโครงการวิจัย

เพื่อสร้างโปรแกรมต้นแบบช่วยในการจัดเรียงกล่องสินค้าหรือผลิตภัณฑ์ลงในตู้คอนเทนเนอร์ โดยการหาวิธีการแบบต่างๆ เพื่อจัดเรียงกล่องผลิตภัณฑ์ขนาดต่างๆ กัน ลงในตู้สินค้า โดยคำนึงถึงการใช้พื้นที่ในตู้คอนเทน

เนอร์เพื่อเป็นการลดจำนวนความต้องการใช้ตู้สินค้าให้น้อยที่สุด โดยการใช้โปรแกรมดังกล่าวมีดัชนีชี้วัดในด้านความสามารถของโปรแกรมเพื่อเป็นเกณฑ์ความสำเร็จของโครงการดังต่อไปนี้

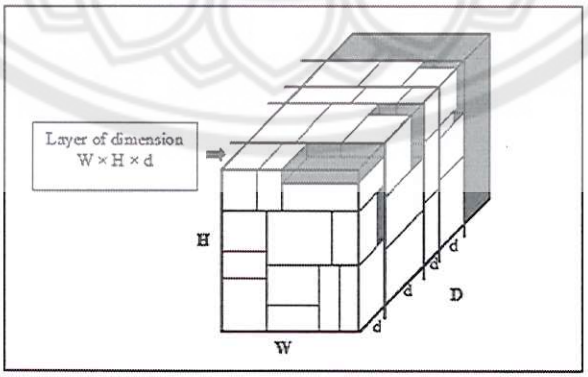
- สามารถเลือกขนาดของตู้คอนเทนเนอร์โดยขนาดของตู้จะใช้ข้อมูลจริงซึ่งมีการใช้อยู่หลายขนาด
- สามารถเลือกจำนวนกล่องและรูปแบบการจัดเรียงกล่อง (Heuristic for Arrangement: HA) 2 รูปแบบ คือแบบ Guillotine-cutting Approach กับแบบ Wall-building Approach
- สามารถกำหนดให้กล่อง (กรณีหลายกล่อง) ที่มีเจ้าของเดียวกันให้อยู่ในตู้คอนเทนเนอร์เดียวกันได้

5 การทบทวนวรรณกรรม/สารสนเทศ (Information) ที่เกี่ยวข้อง

ปัญหาการจัดเรียงกล่องลงในคอนเทนเนอร์ (CPP) จัดเป็นประเภทหนึ่งของปัญหาการตัดและการบรรจุ (Cutting and packing problem) (Dyckhoff, 1990) ซึ่งปัญหาการตัดและการบรรจุนี้มีมากมายหลายแบบจนทำให้เกิดความสับสนในลักษณะโครงสร้างและรูปแบบของปัญหา ดังนั้น Dyckhoff (1990) จึงได้พัฒนาแนวทางในการแบ่งประเภทของปัญหาชนิดนี้ด้วยการสร้างรูปแบบมาตรฐานของปัญหาการตัดและการบรรจุให้แบ่งเป็นประเภทที่ชัดเจน โดยใช้สัญลักษณ์ลำดับอักษร 4 ตัวคือ (α / β / γ / δ) ซึ่ง α หมายถึง ขนาดของมิติ, β หมายถึง การกำหนดชนิด, γ หมายถึง ลักษณะของวัตถุขนาดใหญ่ และ δ หมายถึง ลักษณะของวัตถุย่อย Pisinger (2002) ได้อธิบายการประยุกต์ใช้วิธีการฮิวริสติกต่างๆ เข้ามาช่วยแก้ปัญหา CPP โดยฮิวริสติกส์ที่ใช้ในการจัดเรียงกล่องสามารถที่จะแยกได้หลายประเภท เช่น Stack building approach, Wall-building approach, Guillotine cutting approach และ Cuboid arrangement approach โดยสามารถอธิบายวิธีการฮิวริสติกส์แบบต่างๆ ได้ดังนี้

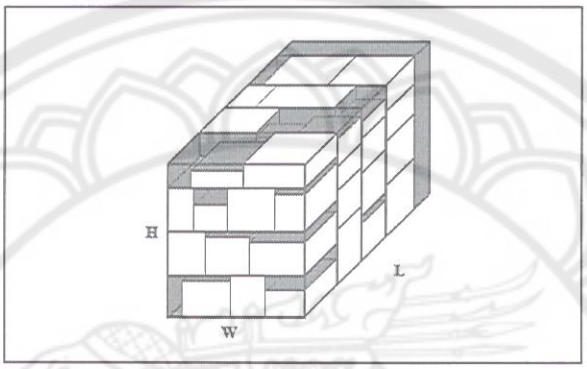
Stack building approach ถูกนำเสนอโดย Gilmore และ Gomory (1965) วิธีการจะคล้ายกับ Wall-building approach แต่จะต่างกันตรงที่การบรรจุกล่องแต่ละใบจะทำการจัดเรียงไปตามพื้นของคอนเทนเนอร์จากนั้นจะบรรจุแต่ละชั้น (Stack) ให้ขนานไปกับพื้นของคอนเทนเนอร์

Wall-building approach ถูกนำเสนอโดย George และ Robinson (1980) และถูกนำไปใช้โดยนักวิจัยหลายท่านได้แก่ Bischoff and Marriott (1990), Gehring, et al., (1990) และ Hemminki (1994) วิธีการทำ Wall-building approach คือ นำกล่องแต่ละใบมาจัดเรียงตามแบบผนังกำแพงซึ่ง 1 กำแพงคือ 1 ชั้น จากนั้นบรรจุแต่ละชั้น (Layer) ลงในคอนเทนเนอร์ตามแนวขวางของด้านลึก (Depth: D) ของคอนเทนเนอร์ ดังแสดงในภาพที่ 4



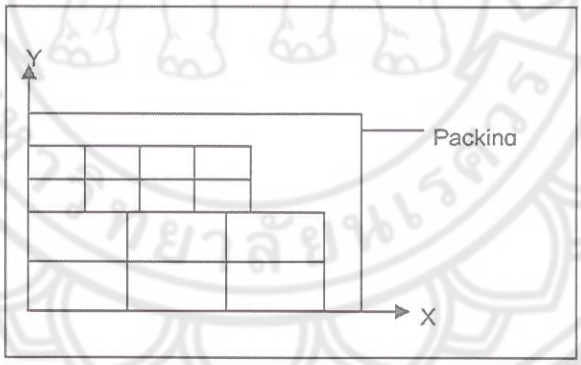
ภาพ 4 แสดงการบรรจุแบบ Wall-building approach

Guillotine cutting approach ถูกนำเสนอโดย Morabito และ Arenales (1994) การบรรจุแบบ Guillotine คือการบรรจุโดยที่ปริมาตรของคอนเทนเนอร์สามารถแบ่งออกเป็นชั้นๆ โดยใช้เส้นตัดขาดระหว่างสองชั้นที่ติดกันของกล่อง ซึ่งเส้นที่ตัดขาดนี้จะไม่ตัดผ่านกลางกล่องที่บรรจุอยู่ในคอนเทนเนอร์ และความสูง (Height) หรือความยาว (Length) ของแต่ละชั้นจะถูกกำหนดจากความสูง หรือความยาวของกล่องที่สูงหรือยาวที่สุด ที่ถูกจัดเรียงเข้าไปในชั้นนั้น ดังแสดงในภาพที่ 5



ภาพ 5 แสดงการบรรจุแบบ Guillotine cutting approach

Cuboid arrangement approach ถูกนำเสนอโดย Bortfeldt และ Gehring (1997) จะเป็นการบรรจุกล่องลงในคอนเทนเนอร์โดยพยายามจัดเรียงกล่องที่มีลักษณะคล้ายคลึงกันให้เป็นรูปลูกบาศก์ ดังแสดงในภาพที่ 6



ภาพ 6 แสดงการบรรจุแบบ Cuboid arrangement approach

สมการคณิตศาสตร์ที่สามารถนำมาใช้การคำนวณหาค่าประสิทธิภาพของการใช้พื้นที่ภายในสำหรับปัญหาการจัดเรียงกล่องลงในตู้คอนเทนเนอร์สามารถแสดงได้ดังนี้

Maximise $$V = \sum_{j=1}^{C} \sum_{i=1}^{n} v_i x_{ij} \qquad (1)$$

Subject to $$\sum_{j=1}^{C} x_{ij} \leq 1 \quad (i = 1, \ldots, n) \qquad (2)$$

$$\sum_{i=1}^{n} s_i x_{ij} \leq S_{\max} \ (j = 1, \ldots, C) \qquad (3)$$

$$s_i, v_i, S_{max} > 0 \qquad (4)$$

เมื่อ i คือ กล่องที่ i^{th} จากจำนวนกล่องสินค้าทั้งหมด n กล่อง

j คือ ตู้คอนเทนเนอร์ที่ j^{th} จากจำนวนตู้สินค้าทั้งหมด C ตู้

s_i คือ ขนาดของกล่องที่ i^{th}

v_i คือ ปริมาตรของกล่องที่ i^{th}

S_{max} คือ ขนาดของตู้คอนเทนเนอร์

เอกสารอ้างอิงของโครงการวิจัย

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.

Bischoff, E.E. & Marriott, M.D. (1990). A comparative evaluation of heuristics for container loading. European Journal of Operational Research, 44, 267-276.

Blazewicz, J., Domschke, W. & Pesh, E. (1996a). The job shop scheduling problem: Conventional and new solution techniques. European Journal of Operational Research, 93, 1-33.

Blazewicz, J., Domschke, W. & Pesh, E. (1996b). Scheduling Computer and Manufacturing Processes. Berlin: Springer.

Bortfeldt, A. & Gehring, H. (1997). Applying tabu search to container loading problems. Operations Research Proceedings, 533-538.

Bortfeldt, A. & Gehring, H. (2001). A hybrid genetic algorithm for the container loading problem. European Journal of Operational Research, 131, 143-161.

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.

Dasgupta, D. (2006). Advances in artificial immune systems. IEEE computational intelligence magazine. November, 40-49.

Dowsland, K.A. (1987). An exact algorithm for the pallet loading problem. European Journal of Operational Research, 31, 78-84.

Dyckhoff, H. (1990). A typology of cutting and packing problems. European Journal of Operational Research, 44, 145-159.

Garey, M.R. & Johnson, D.S. (1979). Computers and Intractability: A Guide to the Theory of NP-completeness. New York: Freeman.

Gehring, M., Menscher, K. & Meyer, M. (1990). A computer-based heuristic for packing pooled shipment containers. European Journal of Operational Research, 44, 277-288.

Gen, M. & Cheng, R. (1997). Genetic Algorithms and Engineering Design. New York: Wiley.

George, J.A. & Robinson, D.F. (1980). A heuristic for packing boxes into a container. Computers and Operations Research, 7, 147-156.

Gilmore, P.C. & Gomory, R.E. (1965). Multistage cutting stock problems of two and more dimensions. Operation Research, 13, 94-120.

Goldberg, D.E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Massachusetts: Addison-Wesley.

Hart, E.A. and Timmis, J. (2008). Application areas of AIS: The past, the present and the future. Applied Soft Computing, 8, 191-201.

Hemminki, J. (1994). Container loading with variable strategies in each layer. In ESI-X, EURO Summer Institute. Jouy-en-Josas, France.

Lu, Y. (1991). Solving combinatorial optimization problems by simulated annealing, genetic algorithms, and neural networks. Master thesis, The University of Minnesota, Minnesota.

Morabito, R. & Arenales, M. (1994). An AND/OR graph approach to the container loading problem. International Transactions in Operational Research, 1, 59-73.

Nagar, A., Haddock, J. & Heragu, S. (1995). Multiple and bicriteria scheduling: A literature survey. European Journal of Operational Research, 81, 88-104.

Osman, I.H. & Laporte, G. (1996). Metaheuristics: A bibliography. Annals of Operations Research, 63, 513-623.

Pisinger, D. (2002). Heuristics for the container loading problem. European Journal of Operational Research, 141, 382-392.

6 ระเบียบวิธีวิจัย

1. ศึกษาโครงสร้างและขั้นตอนการทำงานอย่างละเอียดของวิธีการระบบภูมิคุ้มกันเสมือน (Artificial Immune System) วิธีจีเนติกอัลกอริทึม (Genetic algorithms) และวิธีพาร์ทิเคิลสวอมออฟติไมเซชัน (Particle Swarm Optimisation)
2. ศึกษาปัญหาการหาค่าที่ดีที่สุดแบบเรียงสับเปลี่ยน (Combinatorial optimisation problem) เพื่อเลือกนำมาใช้ในการประยุกต์แก้ปัญหาด้วยวิธีเมต้าฮิวริสติกส์แบบต่างๆที่กล่าวไว้ข้างต้น พร้อมกับรวบรวม วิเคราะห์ ออกแบบและบันทึกข้อมูลที่เกี่ยวข้องต่อไป
3. ออกแบบโครงสร้างของตัวเก็บค่าคำตอบ (Candidate solutions) โดยตัวเก็บค่าคำตอบดังกล่าวอาจจะต้องมีการเข้ารหัส (Encoding) เพื่อนำไปผ่านกระบวนการหรือขั้นตอนการทำงานของแต่ละวิธีต่อไป และเป็น Primary key ในการเรียกใช้ข้อมูลที่เกี่ยวข้องด้วย
4. วิเคราะห์และออกแบบโครงสร้างการทำงานของโปรแกรมจำลองทางคอมพิวเตอร์ (Computer simulation program) และรองรับต่อการทำงานกับผู้ใช้งานเชิงรูปภาพ (Graphic user interface) ที่สามารถเลือกใช้วิธีเมต้าฮิวริสติกส์แบบใดก็ได้ เพื่อการศึกษา ทดลองและเปรียบเทียบเชิงประสิทธิภาพของวิธีดังกล่าว
5. พัฒนาโปรแกรมแบบจำลองทางคอมพิวเตอร์ โดยจะต้องสามารถเลือกใช้ชุดข้อมูลป้อนเข้าชุดใดๆ ได้ ซึ่งโปรแกรมดังกล่าวจะต้องสามารถเลือกกำหนดค่าตัวแปรหรือปัจจัยต่างๆ (Parameters and mechanisms)

ของแต่ละวิธีได้ และโปรแกรมดังกล่าวจะต้องสามารถรายงานค่าคำตอบที่ดีที่สุดจากการค้นหาแต่ละวิธี พร้อมเวลาที่ใช้ในการคำนวณด้วย

6. ทดสอบ (Validation and verification) การทำงานของโปรแกรมในทุกวิธี และทุกๆ ขั้นตอน เพื่อยืนยันการทำงานของโปรแกรมที่ถูกต้อง หากพบข้อผิดพลาดจะต้องทำการแก้ไข (Debugging) ต่อไป
7. ออกแบบ และดำเนินการทดลอง พร้อมเก็บผลการทดลองเพื่อทำการวิเคราะห์ผลกระทบของการกำหนดค่าให้ตัวแปรหรือปัจจัยต่อประสิทธิภาพการทำงานของแต่ละวิธี ทั้งนี้จะต้องนำทฤษฎีว่าด้วยการออกแบบและการวิเคราะห์การทดลอง (Experimental design and analysis) ที่เกี่ยวข้องมาประยุกต์เพื่อผลการวิจัยที่ถูกต้องและน่าเชื่อถือตามหลักสถิติ ก่อนที่จะเขียนบทความเผยแพร่ผลงานวิจัยเบื้องต้นต่อไป
8. ออกแบบ และดำเนินการทดลอง พร้อมเก็บผลการทดลองเพื่อทำการศึกษาเปรียบเทียบเชิงประสิทธิภาพการทำงานของของแต่ละวิธี โดยใช้ผลการศึกษาในข้อที่ 4 มาใช้ในการกำหนดปัจจัยของแต่ละวิธี ทั้งนี้จะต้องนำทฤษฎีว่าด้วยการออกแบบและการวิเคราะห์การทดลอง (Experimental design and analysis) ที่เกี่ยวข้องมาประยุกต์เพื่อผลการวิจัยที่ถูกต้องและน่าเชื่อถือตามหลักสถิติ ก่อนที่จะเขียนบทความเผยแพร่สรุปผลงานวิจัยตลอดโครงการ
9. จัดทำรายงานฉบับสมบูรณ์เสนอผู้ให้ทุน เพื่ออธิบายความสำเร็จของโครงการวิจัย

7 ผลงานตีพิมพ์เผยแพร่ (Output) ที่ได้จากโครงการ

ได้ผลงานที่เขียนบทความวิจัยเพื่อตีพิมพ์ในวารสารวิชาการระดับนานาชาติ จำนวน 1 ฉบับ (ดูรายละเอียดของบทความที่แนบไว้ในภาคผนวก) คือ

1) Development of a stochastic optimisation tool for solving the multiple container packing problems, International Journal of Production Economics, 140, 737-748, Impact Factor 2010 = 1.988.

8 ประวัติผู้วิจัย

1. ชื่อ-สกุล (ภาษาไทย) นายภูพงษ์ พงษ์เจริญ
   (ภาษาอังกฤษ) Pupong Pongcharoen
2. รหัสประจำตัวนักวิจัยแห่งชาติ 49-05-0014
3. ตำแหน่งปัจจุบัน ผู้ช่วยศาสตราจารย์ ระดับ 8
4. หน่วยงานที่อยู่ที่สามารถติดต่อได้สะดวก พร้อมหมายเลขโทรศัพท์ โทรสาร และ E-mail

ภาควิชาวิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์
มหาวิทยาลัยนเรศวร ตำบลท่าโพธิ์ อำเภอเมือง จังหวัดพิษณุโลก
โทรศัพท์ 055-964201 โทรสาร 055-964003
E-mail: pupongp@yahoo.com และ pupongp@nu.ac.th

5. ประวัติการศึกษา

จบปี พ.ศ.	วุฒิการศึกษา	สถานศึกษา
2537	วิศวกรรมศาสตร์บัณฑิต	มหาวิทยาลัยเชียงใหม่
2539	M.Eng. (Industrial Engineering)	Asian Institute of Technology (AIT)
2544	Ph.D. (Manufacturing Engineering)	University of Newcastle upon Tyne, UK.

6. สาขาวิชาการที่มีความชำนาญพิเศษ

Production and operation management; Supply chain and logistics; Applied statistics and operation research; Computer simulation; System modelling; Intelligent optimisation techniques (meta-heuristics); Sequencing and scheduling.

7. ประสบการณ์ที่เกี่ยวข้องกับการบริหารงานวิจัยและงานวิจัยทั้งภายในและภายนอกประเทศ

7.1 ผู้อำนวยการแผนงานวิจัย: ชื่อแผนงานวิจัย -

7.2 หัวหน้าโครงการวิจัย: ชื่อโครงการวิจัย -

7.3 งานวิจัยที่ทำเสร็จแล้ว: ชื่อข้อเสนอการวิจัย ปีที่พิมพ์ และการเผยแพร่

Hicks C, **Pongcharoen P.** An evaluation of various control strategies for companies that produce complex products with stochastic processing times, International Journal of Technology Management 2009; 48(2): 202-218. ISSN 0267-5730.

Pongcharoen P, Promtet W, Yenradee P, Hicks C. Stochastic optimisation timetabling tool for university course scheduling, International Journal of Production Economics 2007; 112(2): 903-918. ISSN 0925-5273.

Pongcharoen P, Khadwilard A, Hicks C. A genetic algorithm with a new repair process for solving multi-stage, multi-machine, multi-product scheduling problems, Industrial Engineering and Management Systems 2008; 7(3): 204-213. ISSN 1598-7248.

Pongcharoen P, Chainate W, Thapatsuwan P. Exploration of genetic parameters and operators through travelling salesman problem, ScienceAsia 2007; 33(2): 215-222. ISSN 1513-1874.

Chainual A, Lutuksin T, **Pongcharoen P.** Computer based Scheduling Tool for Multi-product Scheduling Problems, International Journal of the Computer, the Internet and Management 2007; 15(4): 26.1-6. ISSN 0858-7027.

Hicks C, **Pongcharoen P.** Dispatching rules for production scheduling in the capital goods industry, International Journal of Production Economics 2006; 104(1): 154-163. ISSN 0925-5273.

Pongcharoen P, Hicks C, Braiden PM. The development of genetic algorithms for the finite capacity scheduling of complex products, with multiple levels of products structure, European Journal of Operational Research 2004; 152: 215-225. ISSN 0377-2217.

Pongcharoen P, Hicks C, Braiden, PM, Stewardson DJ. Determining optimum genetic algorithm parameters for scheduling the manufacturing and assembly of complex products, International Journal of Production Economics 2002; 78(3): 311-322. ISSN 0925-5273.

Pongcharoen P, Stewardson DJ, Hicks C, Braiden PM. Applying designed experiments to optimise the performance of genetic algorithms used for scheduling complex products in the capital goods industry, Journal of Applied Statistics 2001; 28(3&4): 441-455. ISSN 0266-4763.

Pongcharoen P, Promtet W, Yenradee P, Hicks C. The development of a genetic algorithm tool for university course timetabling, Pre-prints of the 14th International Working Seminar on Production Economics 2006; 3: 259-270, February, 20th-24th, Innsbruck, Austria.

๕ - ส.ค. ๒๕๕๖

Hicks C, Hossen F, **Pongcharoen P**. Laissez-faire or full control? An evaluation of various control strategies for companies that produce complex products with stochastic processing times, Pre-prints of the 14th International Working Seminar on Production Economics 2006; 4: 67-80, February, 20th-24th, Innsbruck, Austria.

Pongcharoen, P, Khadwilard, A. and Klakankhai, A. Optimising logistics chain network using genetic algorithms, Proceedings of the 18th International Conference on Production Research 2005, July 31st-August 4th, Salerno, Italy.

Hicks C, **Pongcharoen P**, Braiden PM. Stochastic simulation studies of dispatching rules for production scheduling in the capital goods industry, Proceedings of the 13th International Working Seminar on Production Economics 2004, Innsbruck, Austria, February, 16th-20th.

Pongcharoen P, Promtet W. Exploring and determining genetic algorithms parameters through experimental design and analysis, Proceedings of the 33rd International Conference on Computer and Industrial Engineering 2004, Jeju, Korea, March 24th-27th.

Stewardson DJ, Hicks C, **Pongcharoen P**, Coleman SY, Braiden PM. Overcoming complexity via statistical thinking: optimising genetic algorithms for use in complex scheduling problems via designed experiments, Tackling Industrial Complexity: the Ideas That Make a Difference 2002, Downing College, Cambridge, London, April 9th-10th, p275-290, ISBN 1-902546-24-5.

Stewardson DJ, Hicks C, **Pongcharoen P**, Braiden PM. Sparse experimental design: an effective and efficient way of discovering better genetic algorithm structures, Proceedings of the 2nd European Conference on Intelligent Management Systems in Operations 2001, University of Salford, Manchester, UK, July 3rd-4th.

Stewardson DJ, Hicks C, **Pongcharoen P**, Braiden PM. A demonstration of the utility of fractional experimental design for finding optimal genetic algorithm parameter settings, Proceedings of 2nd European Conference on Intelligent Management Systems in Operations 2001, University of Salford, Manchester, UK, July 3rd-4th.

Pongcharoen P, Hicks C, Braiden PM, Metcalfe AV, Stewardson DJ. Using genetic algorithms for scheduling the production of capital goods, Proceedings of the 11th International Working Seminar on Production Economics 2000, February 21st-25th, Igls, Austria, p429-458.

Pongcharoen P, Stewardson DJ, Hicks C, Braiden PM. Applying designed experiments to optimise the performance of genetic algorithms used for scheduling complex products in the capital goods industry, Proceedings of International Conference on the Industrial Statistics in Action 2000, University of Newcastle upon Tyne, UK, September 8th-10th, p98-112.

Pongcharoen P, Hicks C, Braiden PM, Stewardson DJ. Scheduling complex products using genetic algorithms with alternative fitness functions, Proceedings of the International Conference on Complexity and Complex Systems in Industry 2000, University of Warwick, UK, September 19th-20th, p653-663, ISBN 0-902683-50-0.

ภาคผนวก

Int. J. Production Economics 140 (2012) 737–748

Contents lists available at ScienceDirect

Int. J. Production Economics

journal homepage: www.elsevier.com/locate/ijpe

Development of a stochastic optimisation tool for solving the multiple container packing problems

Peeraya Thapatsuwan [a], Pupong Pongcharoen [a,*], Chris Hicks [b], Warattapop Chainate [c]

[a] Industrial Engineering Department, Faculty of Engineering, Naresuan University, Pitsanulok 65000, Thailand
[b] Business School, University of Newcastle upon Tyne, Newcastle upon Tyne, NE1 7RU, UK
[c] Faculty of Liberal Arts and Science, Kasetsart University Kamphaeng Saen Campus, Nakhonpathom 73140, Thailand

ARTICLE INFO

Article history:
Received 5 April 2010
Accepted 11 May 2011
Available online 18 May 2011

Keywords:
Container packing
Artificial Immune System
Genetic Algorithm
Particle Swarm Optimisation
Parameter setting
Experimental design and analysis

ABSTRACT

Marine logistics has become increasingly important as the amount of global trade has increased. Products are usually packed in various sizes of boxes, which are then arranged into containers before shipping. Shipping companies aim to optimise the use of space when packing heterogeneous boxes into containers. The container packing problem (CPP) aims to optimise the packing of a number of rectangular boxes into a set of containers. The problems may be classified as being homogeneous (identical boxes), weakly heterogeneous (a few different sizes) or strongly heterogeneous (many different boxes). The CPP is categorised as an NP hard problem, which means that the amount of computation required to find solutions increases exponentially with problem size.

This work describes the development and application of an Artificial Immune System (AIS), Particle Swarm Optimisation (PSO) and a Genetic Algorithm (GA) for solving multiple container packing problems (MCPP). The stochastic optimisation tool was written in Microsoft Visual Basic. A sequential series of experiments was designed to identify the best parameter settings and configuration of the algorithms for solving MCPP. The work optimised the packing of a standard marine container for a strongly heterogeneous problem.

The experimental results were analysed using the general linear model form of analysis of variance to identify appropriate algorithm configuration and parameter settings. It was found that each algorithm's parameters were statistically significant with a 95% confidence interval. The best configurations were then used in a sequential experiment that compared the performance of the AIS, PSO and GA algorithms for solving 21 heterogeneous MCPP. It was found that the average best-so-far solutions obtained from AIS were marginally better than those produced by GA and PSO for all problem sizes but AIS required longer computational time than GA.

1. Introduction

The increased globalisation of trade has led to a large increase in the volume of shipping. Most international cargo is transported in containers through major seaports. The efficiency of container packing is very important for service providers and can have a large impact on profitability.

The container packing problem (CPP) involves arranging a set of boxes into a set of containers with fixed dimensions. The objective is to minimise the amount of wasted space. The quality of solutions is usually measured in terms of space (volume) utilisation. The total number of sequences for arranging n boxes is ($n!$), which is further increased as there are six ways of turning or flipping each box (6^n); so the number of possible solutions for arranging 10 boxes can be up to ($10! \times 6^{10}$) or 219 billion possible sequences. In terms of computational complexity the CPP is an NP hard problem (Soak et al., 2008), which means that the amount of computation required increases exponentially with problem size.

Metaheuristics, such as Genetic Algorithms (GA), Particle Swarm Optimisation (PSO) and more recently Artificial Immune Systems (AIS), have been successfully applied to solve large and complex combinatorial optimisation problems (Farmer et al., 1986; Goldberg, 1989). These stochastic search methods are capable of finding near optimal solutions within an acceptable amount of computational time. AIS has been successfully used to solve combinatorial optimisation problems such as flow shop scheduling (Engin and Doyen, 2004), job-shop scheduling (Tsai et al., 2007), project scheduling (Agarwal et al., 2007) and the travelling salesman problem (Pongcharoen et al., 2008). However, no research has been reported that has used AIS to solve CPP.

The objectives of this paper are to:

- describe the development of a computer aided packing (CAP) programme that includes AIS, PSO and GA for solving a wide

* Corresponding author.
E-mail addresses: pupongp@yahoo.com, pupongp@nu.ac.th (P. Pongcharoen).

0925-5273/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.ijpe.2011.05.012

range of multiple container packing problems (to enable the performance of AIS to be compared with those of GA and PSO, which have been shown to be effective methods for solving CPP);
- perform a series of experiments based upon a design of experiments approach to find the best settings for AIS parameters for various types of problems;
- benchmark the performance of the AIS, PSO and GA methods in terms of the quality of solutions obtained and the computational time required.

The remaining sections in this paper are organised as follows. Section 2 reviews the literature relating to container packing problems. Section 3 describes the formulation of the problem and presents a mathematical model. Section 4 briefly describes the process and pseudo-code of the proposed algorithms. Descriptions of AIS, PSO and GA procedures for solving CPP are detailed in Section 5. Section 6 presents the experimental design and analyses results. Finally, Section 7 summarises the conclusions of the research and suggests possible further work.

2. Container packing problems

There are many cutting and packing problems with different names that have the same logical structure. These include cutting stock and trim loss problems; bin packing, dual bin packing, strip packing, vendor packing and knapsack problems; vehicle loading, pallet loading, container loading and car loading problems; and layout, nesting and the partition problem (Dyckhoff, 1990). Dyckhoff developed a comprehensive typology of these problems that was further developed by Wascher et al. (2007). The common logical structure is that there are data that define the geometry of items with fixed shapes in one or more dimensions using real numbers. There is one group of large objects and a group of small objects. The cutting or packing process produces geometric combinations of small objects that are assigned to the large objects. Strip packing problems have a container with fixed height and width, but unspecified depth; the objective is to pack all the boxes so that the depth can be minimised. In the knapsack problem, each box has an associated profit; the objective is to load the boxes into a single container so that profit can be maximised. The bin packing problem has containers with fixed sizes; the objective is to pack the boxes into the specified number of containers (Pisinger, 2002). The container packing problem aims to optimise the volume utilisation in container(s) (Ngoi et al., 1994). The container loading problem may additionally take into account weight and weight distribution (Davies and Bischoff, 1999).

Bischoff and Marriott (1990) pointed out that the above problem types cover many situations with significantly different characteristics. First, there is a distinction between situations where a specified cargo is to be shipped in the best combination of containers and others where the amount of cargo assigned to a single container is maximised. Second, some problems aim to maximise the volume utilisation whereas others focus on the value of items packed. The constraints and assumptions also vary; for example some problems take into account the weight and weight distributions, whereas others do not. Cargo fragility and material handling aspects are further considerations that can be very significant in practice. They point out that there are many other factors that could be considered; the problem constraints and objectives are highly variable and may be difficult to define precisely. Thus the there are many types of CPP that involve different (and sometime multiple) objectives and constraints.

The CPP can be classified according to various criteria, including packing schemes (wall building/guillotine cutting); homo/heterogeneous objects; rectangular/nonrectangular packing; and n-dimensional shapes and single/multiple container(s). Common objective functions for three-dimensional packing include the minimisation of the length/number of container required for a specified cargo; maximising the volume of the cargo packed in a given container; and space (volume) utilisation.

Various assumptions have been made in order to simplify, formulate and solve CPP. The most common assumptions can be summarised as follows: (i) boxes are of rectangular shape; (ii) boxes must be arranged within the whole container and must be parallel to its side walls; (iii) boxes cannot overlap each other; (iv) boxes can/cannot be rotated; (v) boxes are stabilised by filling the empty space with foam rubber (Pisinger, 2002); (vi) boxes are to be packed into a single container/multiple containers; (vii) boxes are supported by those underneath and (viii) unlike container loading, weight limitations and weight distribution may be ignored in CPP (Davies and Bischoff, 1999).

Container packing problems are NP hard. Therefore there are no general algorithms that can guarantee an optimal solution, so the methods adopted for its solution are based upon heuristic approaches (Bischoff and Marriott, 1990; Bortfeldt et al., 2003). It has been a popular area of research and there are many articles that provide good reviews of the relevant literature (see for example, Bischoff and Marriott, 1990; Bortfeldt and Gehring, 2001; Chen et al., 1995; Egeblad and Pisinger, 2009; Ngoi et al., 1994; Pisinger, 2002). The wall building (George and Robinson, 1980) and guillotine cutting approaches (Morabito and Arenales, 1994) are the most commonly used arrangement heuristics.

George and Robinson (1980) developed a wall building approach to container packing, which has been very widely applied and has formed the basis of many variants (Bischoff and Marriott, 1990). It fills the container in layers across its width. The depth of each layer is determined by the size of the first box packed into the layer. The procedure starts from one end of the container and attempts to keep an even workface over the cross section of the container. The filling scheme chooses a box type and then completes as many columns as possible (widthwise). A ranking scheme is used to select boxes. The size of the smallest dimension is the first ranking criterion; the box with the largest value has the highest ranking because it is difficult to pack large boxes towards the end of the packing process. The second criterion ranks according to the quantity of boxes of each type, because the layer is more likely to be filled throughout the layer if there are many similar boxes. The length of the largest dimension is the third criterion, which means awkwardly long boxes are packed early. The types of boxes to be loaded are considered to be 'opened', if boxes of the same type have already been loaded, or 'unopened' if not. The heuristic gives preference to boxes of an 'open' type when starting a new layer. If there are no 'open' boxes, the first box in a layer (the layer determining box) is chosen using either the first or the second ranking criterion. It determines the depth of the cross section considered at the current stage of the packing process. The current layer is filled with boxes of the same type, with other boxes used to fill voids.

Morabito and Arenales (1994) developed a guillotine cutting approach. An initial cut is made within the container's length L, which produces two boxes B and C that are then cut independently; each box can be further cut. The boxes may be represented as nodes in an orientated graph. The root node represents the container. A cut on a box is represented as an arc in the orientated graph pointing to two successor nodes that represent the new boxes created by the cut. The leaf nodes represent the final boxes produced by the cutting sequence. The set of all nodes and arcs is called an AND/OR graph. A search strategy that used

back-tracking and hill-climbing was developed to traverse the AND/OR graph and enumerated the nodes.

A wide range of optimisation methods have been used to find solutions to CPP. Methods based upon full enumerative search have been used, including Linear Programming (Beasley, 1985) and Branch and Bound (Pisinger, 2002), but these are suitable only for relatively small problems because CPP is NP hard. A range of metaheuristic methods have also been used, including Tabu Search (Bortfeldt et al., 2003; Gendreau et al., 2006), Genetic Algorithms (Bortfeldt and Gehring, 2001; Thapatsuwan et al., 2007), Ant Colony Optimisation (Lee et al., 2005) and Particle Swarm Optimisation (Thapatsuwan et al., 2009). These metaheuristics have included heuristics such as wall building or guillotine cutting for generating the detailed packing sequences. Bortfeldt and Gehring (2001) found that the Genetic Algorithm was particularly suitable for strongly homogeneous box sets and also commented that the heuristic approach for packing had a larger impact on the results than the search strategy.

3. Problem formulation

A general mathematical model for maximising the efficiency of volume usage (V) for CPP has been developed by previous research (Chen et al., 1995; Christensen and Rousoe, 2009). The formulation of the problem is as follows:

Notation

- B total number of boxes
- C total number of containers
- l_i, w_i, h_i parameters indicating, respectively, the length, width and height of box i
- L_j, W_j, H_j parameters indicating, respectively, the length, width and height of container j
- x_i, y_i, z_i continuous variables indicating the coordinates of back-left-bottom corner of box i that specifies the placement of box i
- l_i^x, l_i^y, l_i^z binary variables that indicate whether the length of box i is parallel to the X-, Y- or Z-axis; for example, $l_i^x=1$ if the length of box i is parallel to the X-axis; otherwise it is equal to 0
- w_i^x, w_i^y, w_i^z binary variables indicating whether the width of box i is parallel to the X-, Y- or Z-axis; for example, $w_i^x=1$ if the width of box i is parallel to the X-axis; otherwise it is equal to 0
- h_i^x, h_i^y, h_i^z binary variables indicating whether the height of box i is parallel to the X-, Y- or Z-axis; for example, $h_i^x=1$ if the height of box i is parallel to the X-axis; otherwise it is equal to 0
- le_{ik} a binary variable indicating if box i is placed on the left side of box k
- ri_{ik} a binary variable indicating if box i is placed on the right side of box k
- be_{ik} a binary variable indicating if box i is placed behind box k
- fr_{ik} a binary variable indicating if box i is placed in front of box k
- ab_{ik} a binary variable indicating if box i is placed above box k
- un_{ik} a binary variable indicating if box i is placed underneath box k
- p_{ij} a binary variable; $p_{ij}=1$ if ith box is placed in jth container; otherwise it is equal to 0
- c_j a binary variable; if $c_j=1$, container j is used; otherwise it is equal to 0
- M an arbitrarily large number used in Big-M constraints

The variables le_{ik}, ri_{ik}, be_{ik}, fr_{ik}, ab_{ik} and un_{ik} are defined only for $i<k$. The container is placed in a coordinate system with its origin at the back-left-bottom corner. The length L of the container is placed along the X-axis, the width W along the Y-axis and the height H along the Z-axis. The objective of MCPP following the linear mixed integer programming model can be formulated as

$$\text{minimize} \quad \sum_{j=1}^{C} L_j W_j H_j c_j - \sum_{i=1}^{B} l_i w_i h_i \tag{1}$$

subject to

$$x_i+(l_i l_i^x)+(w_i w_i^x)+(h_i h_i^x) \le x_k+(1-be_{ik})M \quad \forall i,k,\ i<k \tag{2}$$

$$x_k+(l_k l_k^x)+(w_k w_k^x)+(h_k h_k^x) \le x_i+(1-fr_{ik})M \quad \forall i,k,\ i<k \tag{3}$$

$$y_i+(l_i l_i^y)+(w_i w_i^y)+(h_i h_i^y) \le y_k+(1-le_{ik})M \quad \forall i,k,\ i<k \tag{4}$$

$$y_k+(l_k l_k^y)+(w_k w_k^y)+(h_k h_k^y) \le y_i+(1-ri_{ik})M \quad \forall i,k,\ i<k \tag{5}$$

$$z_i+(l_i l_i^z)+(w_i w_i^z)+(h_i h_i^z) \le z_k+(1-un_{ik})M \quad \forall i,k,\ i<k \tag{6}$$

$$z_k+(l_k l_k^z)+(w_k w_k^z)+(h_k h_k^z) \le z_i+(1-ab_{ik})M \quad \forall i,k,\ i<k \tag{7}$$

$$le_{ik}+ri_{ik}+be_{ik}+fr_{ik}+ab_{ik}+un_{ik} \ge p_{ij}+p_{kj}-1 \quad \forall i,k,\ i<k \tag{8}$$

$$\sum_{j=1}^{C} p_{ij} = 1 \quad \forall i \tag{9}$$

$$\sum_{i=1}^{B} p_{ij} \le M c_j \quad \forall j \tag{10}$$

$$x_i+(l_i l_i^x)+(w_i w_i^x)+(h_i h_i^x) \le L_j+(1-p_{ij})M \quad \forall i,j \tag{11}$$

$$y_i+(l_i l_i^y)+(w_i w_i^y)+(h_i h_i^y) \le W_j+(1-p_{ij})M \quad \forall i,j \tag{12}$$

$$z_i+(l_i l_i^z)+(w_i w_i^z)+(h_i h_i^z) \le H_j+(1-p_{ij})M \quad \forall i,j \tag{13}$$

$$l_i^x,\ l_i^y,\ l_i^z,\ w_i^x,\ w_i^y,\ w_i^z,\ h_i^x,\ h_i^y,\ h_i^z,\ le_{ik},\ ri_{ik},\ be_{ik},\ fr_{ik},\ ab_{ik},\ un_{ik},\ p_{ij},\ c_j \in [0,1] \quad \forall i,k,\ i<k \tag{14}$$

$$x_i, y_i, z_i \ge 0 \quad \forall i \tag{15}$$

Constraints (2)–(7) ensure that the loaded boxes do not overlap each other. Constraint (8) checks for overlap, which is not allowed, and forces at least one of the six variables le_{ik}, ri_{ik}, be_{ik}, fr_{ik}, ab_{ik}, un_{ik} to one. Constraint (9) guarantees that each box will be packed into a single container only. Constraint (10) indicates that a container is considered to be used when any box has been assigned to it. Constraints (11)–(13) make sure that all the boxes packed into a container fit within its physical dimensions. Finally constraints (14) and (15) specify the range of the variables.

4. Approximation optimisation algorithms

Approximation optimisation algorithms, the so called metaheuristics, have received considerable attention over the last few decades. This is because the stochastic search process helps find practical and near optimal solutions, within an acceptable amount of computational time. The methods are particularly popular for solving very large-scale and complex combinatorial optimisation problems because full enumerative search is impractical for these problems. This paper relates to research that applied the Artificial Immune System, Particle Swarm Optimisation and a Genetic Algorithm to solve multiple container packing problems. GA was used as a benchmark because it had been found

to be particularly effective at solving CPP (Bortfeldt and Gehring, 2001).

4.1. Artificial Immune System (AIS)

An immune system is a system of biological structures and processes within an organism that protects against disease by identifying and killing pathogens and tumour cells. It detects a wide variety of agents, from viruses to parasitic worms, and needs to distinguish them from the organism's own healthy cells and tissues in order to function properly. Detection is complicated as pathogens can evolve rapidly, producing adaptations that avoid the immune system and allow the pathogens to successfully infect their hosts (Abbas et al., 2000; Delves et al., 2006). The function of the immune system is to detect and recognise foreign bodies and molecules that enter the body (e.g. viral infections, bacteria or transplanted tissues). It also recognises abnormal or mutated cells such as cancerous cells.

Murphy et al. (2007) described how the immune system functions can be classified as innate and adaptive immune responses. The innate immune response functions through innate immune cells such as phagocytes, natural killer cells and dendritic cells. The adaptive immune response is conferred by 'B' cells (humoral) and 'T' cells (cell mediated). Phagocytes, dendritic cells as well as 'B' cells can specifically present antigens to 'T' cells. The antigens are derived from foreign bodies that those cells have internalised and processed previously. 'T' cells recognise the presented antigens using their specific 'T' cell receptors on the 'T' cell membranes. The specifically stimulated 'T' cell then responds by proliferating, which gives rise to antigen-specific 'T' cell clones. Each clone has 'T' cells with the same specificity to the stimulating antigen. The 'B' cells also specifically bind antigens using membrane receptors called 'B' cell receptors. The antigen-bound 'B' cells are then activated to proliferate and also to become 'antibody-producing' plasma cells. The proliferating 'B' cell clones and the antibodies secreted from plasma cells all have the same specificity to the stimulating antigen. The proliferation rate of a 'B' cell is directly proportional to the degree to which it recognises the antigen. The 'B' cell response learns by raising its population size and affinity (the degree of the cell recognition with the antigen; Murphy et al., 2007).

The Artificial Immune System (AIS) was initially proposed in the mid 1980s by Farmer et al. (1986). AIS is one of several biology-inspired optimisation algorithms, which is a branch of computational intelligence (Dasgupta, 2006). There are variants of AIS algorithms, including immune networks (Farmer et al., 1986), negative selection (Forrest et al., 1994), danger theory (Matzinger, 2002) and clonal selection (de Castro and Von Zuben, 2000). Clonal selection focuses on how 'B' cells and 'T' cells can adapt their self to match and even kill the invaders (Burnet, 1959). It is based on two main principles (Garrett, 2005): clonal selection and affinity maturation by hypermutation principles. With clonal selection each antibody (candidate solution) has an affinity (fitness) value determined by the affinity (objective) function. Affinity maturation consists of two main processes: mutation and receptor editing. Mutation mechanisms such as inverse mutation and/or pairwise interchange mutation can be used to generate a clone from an antibody (Engin and Doyen, 2004). The number of clones is determined by its affinity value and the size of antibody population. After cloning, sorting and deleting the repetition, the receptor editing process is conducted by eliminating antibodies from the population based on the desired percentage of antibody elimination (%B). The whole process is repeated until the termination criterion is satisfied. The pseudo-code of AIS is outlined in Fig. 1.

4.2. Genetic Algorithms (GA)

Genetic Algorithms (GA), initially introduced by Holland (1975), have become one of the best known biology-inspired metaheuristics. The simple GA mechanism starts by encoding the problem to produce a list of genes. The genes are randomly combined to produce a population of chromosomes, each of which represents a possible solution. The population size (P) and the number of generations (G) are important parameters that need to be specified. The combination of P and G determines the number of chromosomes generated, which relates to the amount of search and the computational time required. The next step is to perform genetic operations (crossover and/or mutation) on chromosomes, which are randomly selected from the population as parents, for producing offspring. The fitness function is used to measure the chromosomes' fitness value, from which the probability of survival is determined. After performing the fitness evaluation process, a chromosome selection mechanism such as the Roulette Wheel (Goldberg, 1989) is then used to stochastically choose the same number of chromosomes to the next generation. The GA process is repeated until a termination condition is satisfied. The mechanism of GA is demonstrated as the pseudo-code in Fig. 2.

4.3. Particle Swarm Optimisation (PSO)

Swarm Intelligence is based upon a behavioural simulation of social insects such as wasps, termites and bees. Particle Swarm

```
Initialise the value of AIS parameters [antibody size (P), iterations (I_max), and percentage of antibody
elimination (%B)].
Generate a population of P antibodies
For each antibody (i ε P), calculate affinity (i)
Set current iteration (I) = 1
Do
    For each antibody (i)
        Calculate the number of clones (N_c) and clone antibody (i)
        For each clone, apply inverse mutation to create a new antibody
            Calculate the affinity of the new antibody
            If affinity (new antibody) is better than the clone then clone = new antibody
            Else Perform pairwise interchange mutation to create a new antibody
                Calculate the affinity of the new antibody
                If affinity (new antibody) is better than the clone then clone = new antibody
        antibody (i) = clone
    Eliminate the worst antibodies from the population based on %B
    Create new antibodies to replace the eliminated antibodies
    I = I + 1
While I ≤ I_max
```

Fig. 1. Pseudo-code of the AIS procedure.

Initialise the value of GA parameters [population size (P), number of generations (G), and probabilities of crossover (P_c) and mutation (P_m)].
Generate a population of P chromosomes
For each chromosome ($i \in P$), calculate fitness (i)
Set current generation (g) = 1
Do
  Based on P_c, randomly select two parent chromosomes for crossover operation
  Based on P_m, randomly select a parent chromosome for mutation operation
  Calculate the fitness of the offspring
  If offspring is better than the parent, replace the parent
  Randomly select the survived chromosome for next generation using roulette wheel
  $g = g + 1$
While $g \leq G$

Fig. 2. Pseudo-code of the GA procedure.

Initialise swarm size (N), number of iterations (I_{max}), inertia weight (ω), self (c_1) and social (c_2) learning rates.
Generate a swarm of N particles
For each particle ($i \in N$), calculate fitness (i)
  Set current iteration (I) = 1
  Do
    For each particle, update the best fitness of particle (i) as *pBest*
    Update the best fitness of all particles as *gBest*
    For each particle
      Calculate particle velocity based on the *pBest* and *gBest*
      Update particle position based on the new velocity
    End
    $I = I + 1$
While $I \leq I_{max}$

Fig. 3. Pseudo-code of the PSO procedure.

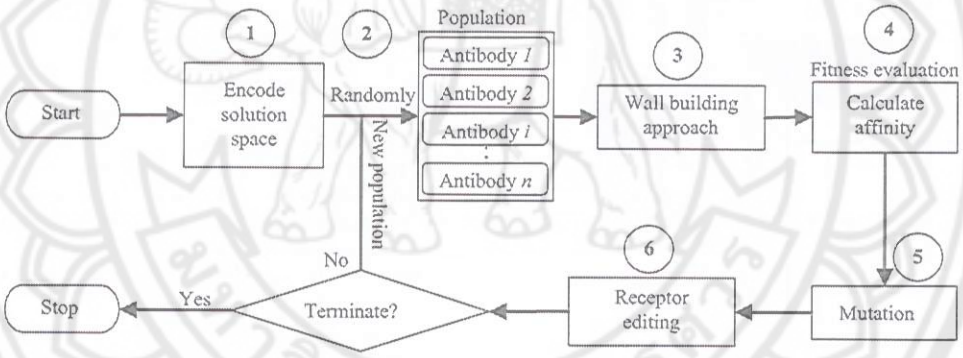


Fig. 4. AIS procedures for solving CPP.

Optimisation (PSO) is inspired by social behaviour of bird flocking or fish schooling. PSO was developed by Kennedy and Eberhart (1995). The pseudo-code of applied PSO is illustrated in Fig. 3. In the general concept of PSO, each particle moves to the next position in the search space by conducting with a velocity according to both its own best previous experience (*pBest*) and the best experience of all members (*gBest*; Kennedy et al., 2001). The formulas are

update velocity operation:

$$V_{id}(t) = \omega_{id} V_{id}(t-1) + c_1\, rand()(P_{id} - X_{id}(t-1)) + c_2\, Rand()(P_{gd} - X_{id}(t-1)) \quad (16)$$

update position operation:

$$X_{id} = X_{id}(t-1) + V_{id} \quad (17)$$

where d is the dimension of the search space and g is the index of the best particle compared with all the particles in the population (*gBest*). The best previous position (*pBest*) of the particle i is represented as $P_i = (P_{i1}, P_{i2}, \ldots, P_{in})$; c_1 and c_2 are positive constants, *rand*() and *Rand*() are random numbers ranging between 0 and 1, and ω is the inertia weight. The rate of position change for particle i is represented by $V_i = (V_{i1}, V_{i2}, \ldots, V_{in})$.

5. Development of the stochastic optimisation tool

In this work the container packing based optimisation tool (CPOT) was developed, which includes AIS, PSO and GA optimisation options. The CPOT programme has approximately 1000 written lines of code (including graphic user interface and the proposed metaheuristics) in modular style using Visual Basic programming.

5.1. AIS procedures for CPP

The process of AIS is illustrated in Fig. 4. The algorithm consists of six main processes: problem encoding, population initialisation, box arrangement using the wall building approach, antibody evaluation, mutation and receptor editing. The main processes are described in the following subsections.

5.1.1. Problem encoding

The design task to be optimised by CPP is to determine the sequence of heterogeneous boxes to be arranged into a container that achieves the best volume utilisation. Each box can be rotated

in 6 ways (see Fig. 5). The encoded sub-antibody contains two items of information: box identifier and type of box rotation. For example, the encoded sub-antibody of B15R2 refers to box number 15 with rotation type II.

5.1.2. Population initialisation

The encoded sub-antibodies are randomly sequenced to generate an antibody. The length of an antibody (candidate solution) is determined by the total number of boxes to be packed. Fig. 6 illustrates a typical antibody representation for an eight-box problem. The first sub-antibody in the given sequence indicates that box number 5 (B5) is the first box to be packed with rotation type III followed by box number 3 (B3) with rotation type I and so on. The process of antibody creation can be repeated to generate a population of the desired size. The population size (P) determines the number of candidate solutions in the solution space; increasing the population size increases the amount of search and the amount of memory and computation required.

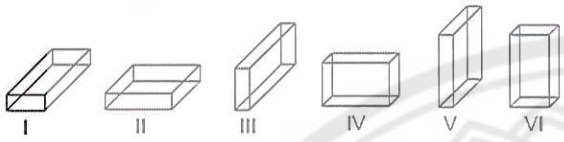


Fig. 5. Six types of box rotation.

B5R3	B3R1	B6R5	B1R2	B8R6	B4R4	B2R3	B7R4

Fig. 6. Antibody representation.

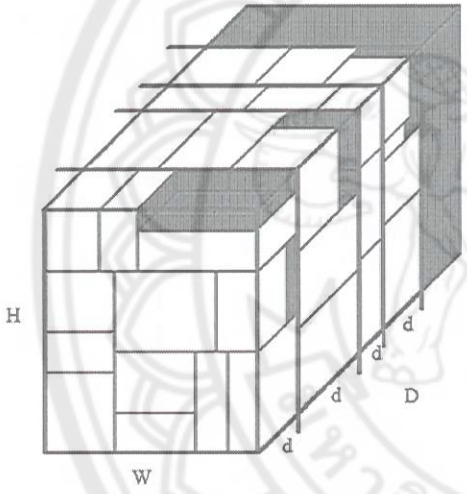


Fig. 7. Wall building approach.

5.1.3. Box arrangement using the wall building approach

After randomly generating an antibody, the boxes are sequentially packed into the container one by one in layers across its width (see Fig. 7). The depth of each layer is determined by the size of the first box packed into the layer. The procedure starts from one end of the container and attempts to keep an even workface over the cross section of the container.

5.1.4. Antibody evaluation

The next stage is to measure the affinity of the antibodies, which is determined by the amount of wasted space within the containers. The maximisation of volume utilisation is very important for service providers and can have a large impact on profitability. The calculation of volume utilisation can use Eq. (1) mentioned in Section 3.

5.1.5. Mutation process

Mutation is the crucial process for AIS algorithm. It determines the diversity of populations and the amount of exploration within the search space. In the mutation process (see Fig. 8), each antibody is cloned; the number of clones is determined by its affinity value (fitness) and the size of antibody population. Each clone is then mutated using the inverse mutation operator shown in Fig. 9. If a newly mutated antibody is better than the original clone, the clone is replaced by the new mutated antibody. Otherwise, the pairwise interchange mutation operator (see Fig. 10) is used to try and produce an antibody with a better affinity value. This process is repeated until all the antibodies are mutated.

5.1.6. Receptor editing

After finishing the mutation process, the cloned antibodies are sorted. The receptor editing process is then conducted by eliminating

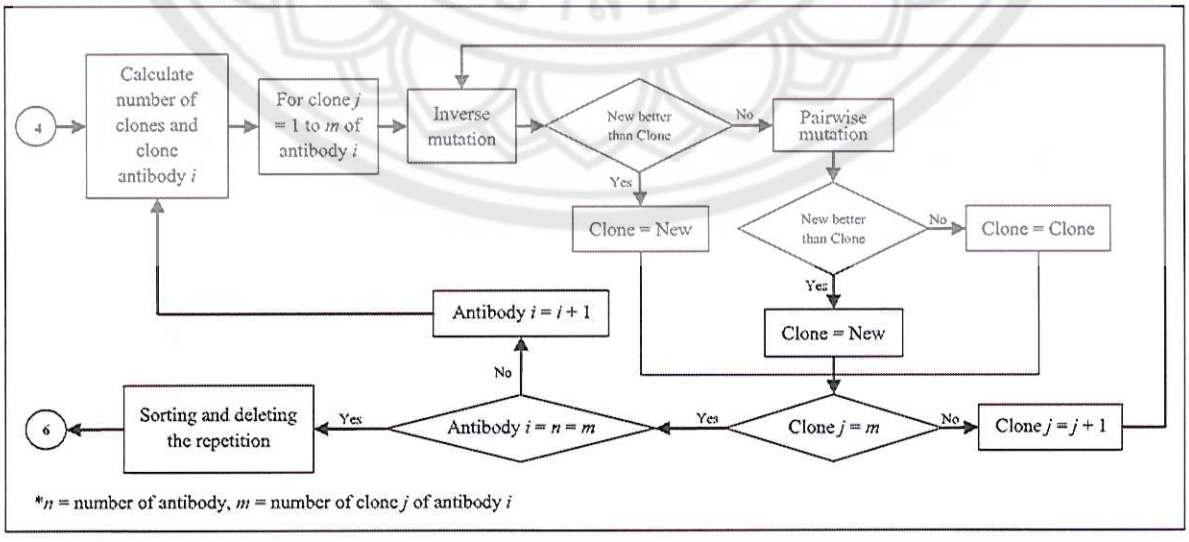


Fig. 8. Mutation mechanism.

a proportion of bad antibodies from the population based on the desired percentage of antibody elimination (%B). For example, %B set to 10 means that the worst 10% of antibodies will be deleted and replaced by the same number of new randomly generated antibodies. The antibody elimination percentage (%B) plays an important role in determining the degree of exploration in the solution space. The whole process is repeated until the iteration (I_{max}) criterion is satisfied. Higher values of the iteration criterion (I_{max}) increase the amount of search, which may lead to an improved solution. I_{max}, and also determine the number of loops in the programme, which influences the amount of computation required.

5.2. GA procedures for CPP

The process of GA for solving CPP is illustrated in Fig. 11. The algorithm consists of six main processes: problem encoding, population initialisation, box arrangement using the wall building approach, genetic operations, fitness evaluation and the Roulette Wheel Selection (RWS) scheme. Since some of the GA procedures are similar to the AIS procedures, only genetic operations (including crossover and mutation) and the RWS scheme are different. Those extra processes are described in the following subsections.

5.2.1. Genetic operations

After a population of chromosomes has been randomly generated, the next stage is to randomly select chromosomes that will be subjected to crossover and mutation operations. In crossover the characteristics of two parents are combined to produce an offspring, whilst mutation produces random change in one chromosome. There are many types of crossover and mutation operations reported in literature (Pongcharoen et al., 2001). The best crossover and mutation operations previously used for solving the CPP (Thapatsuwan et al., 2006) were the cycling crossover (see Fig. 12) and the Enhanced Two Operations Random Swap Mutation (see Fig. 13).

5.2.2. Roulette Wheel Selection (RWS) scheme

In this work, the well-known Roulette Wheel approach (Goldberg, 1989) was used to select chromosomes for the next generation. The number of segments within the wheel equals the number of chromosomes in the population. The size of the segment is determined by the fitness of the chromosome. Wheel spinning is simulated by a uniformly generated random number over the range 0–1. The spinning process is repeated until the

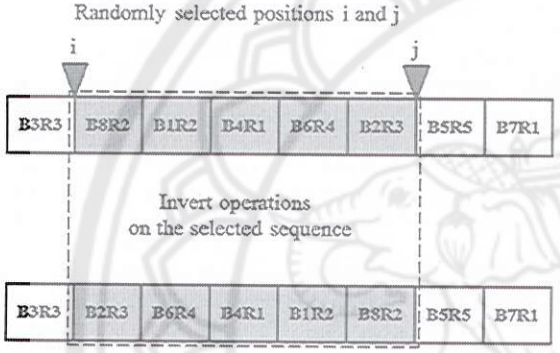


Fig. 9. Inverse mutation.

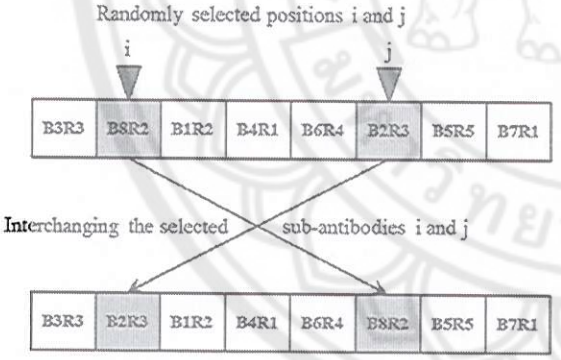


Fig. 10. Pairwise interchange mutation.

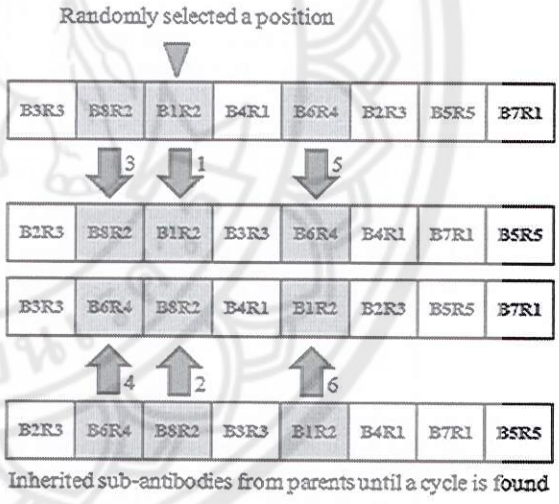


Fig. 12. Cycling crossover.

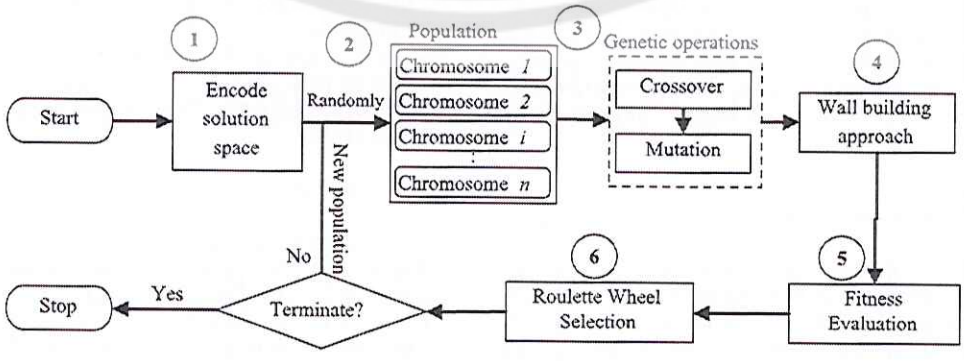


Fig. 11. GA procedures for solving CPP.

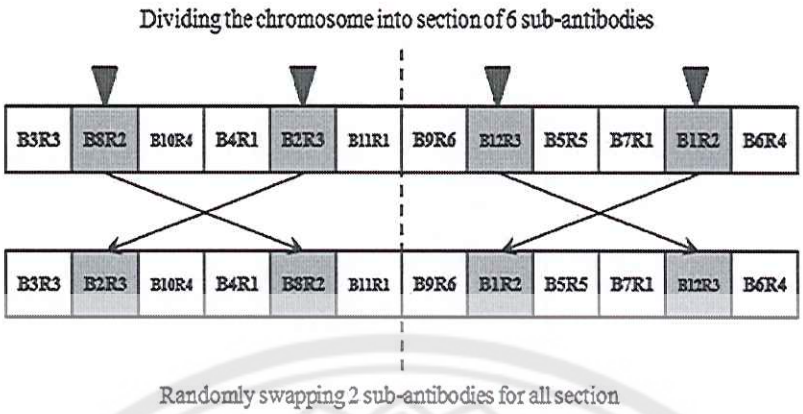


Fig. 13. Enhanced two operations random swap.

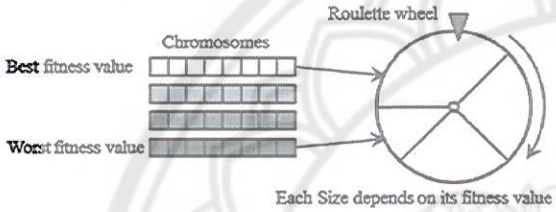


Fig. 14. Roulette Wheel Selection.

correct population size has been produced for the next generation. The probability of survival and number of replicates of a chromosome in the next generation are determined by its fitness. The GA process is repeated until the specified number of generations (G) is satisfied (Fig. 14).

5.3. *PSO procedures for CPP*

The PSO process is illustrated in Fig. 15. The algorithm consists of five main processes: problem encoding, population initialisation, box arrangement using the wall building approach, fitness evaluation and updating velocity and position. The PSO procedures are similar to the AIS procedures apart from updating velocity and position. Those extra processes are described in the following subsections.

5.3.1. *Updating velocity and position*

In each iteration every particle changes its current position and velocity, which are calculated from the previous values of *pBest* and *gBest* as shown in Eqs. (16) and (17), respectively. In combinatorial optimisation problems, approaches such as the swap operator (Wang et al., 2003) or adjustment operator (Cuiru et al., 2005) are required. The procedures for updating velocity and position using the swap operator are shown in Figs. 16 and 17, respectively. The difference between the swap operator and adjustment operators is that the swap operator (SO) exchanges nodes whilst the adjustment operator (AO) inserts nodes.

6. Experimental design and analysis

The experiments conducted in this work were based upon a two-step sequential experiment. Experiment A was designed to identify appropriate settings of the AIS parameters whilst Experiment B aimed to compare the performance of AIS with GA and PSO in terms of the quality of solutions obtained and the computational time required. All computational runs were conducted on a personal computer with Core2Quad 2.66 GHz CPU and 4 GB DDRIII RAM.

A standard marine container is 20 ft long, 8 ft wide and 8 ft high (often referred to as one Twenty-foot Equivalent Unit: TEU). This research considered the packing of standard size containers. The length, width and height of the boxes to be packed were randomly generated according to a uniform distribution in the ranges of 70–100 cm (length), 50–80 cm (width) and 30–60 cm (height). Therefore all the boxes considered in this work had different sizes, so the problems were strongly heterogeneous, the most difficult type of CPP. However the optimisation programme also allows users to specify the sizes of the boxes and containers to meet their requirements, which means that the tool can be used to solve specific or more straightforward problems.

6.1. *Experiment A*

The aim of this experiment was to investigate and identify appropriate settings of the AIS parameters (including the combination of the number of antibodies and the number of iterations (AI) and the percentage of eliminating antibodies ($\%B$)) for a CPP with 100 boxes and identically sized containers. The full factorial experimental design and the range of values considered for each of the factors are shown in Table 1. The computational runs were replicated 30 times with different random seed numbers. In practice, the computation time available is limited. Therefore the combination of the number of antibodies and iterations was fixed at 40,000; test runs identified that this was sufficient to achieve convergent results.

The results were analysed using the general linear model form of analysis of variance (ANOVA), which is shown in Table 2. It can be seen that the combinations of the number of antibodies and iterations (AI), and percentage of eliminated antibodies ($\%B$) were statistically significant with a 95% confidence level. In order to identify the appropriate setting of the factors considered, the main effect plots are therefore provided in Fig. 18. It can be seen that the best settings of AI and $\%B$ parameter were 100×400 and 25, respectively.

6.2. *Experiment B*

This experiment aimed to benchmark the performance of AIS using the best parameter settings that had been identified in experiment A. The GA and PSO parameters were based upon previous research (Thapatsuwan et al., 2006). The settings of GA parameters values used were the combination of population size and the number of generation (PG) $= 100 \times 400$; probability of

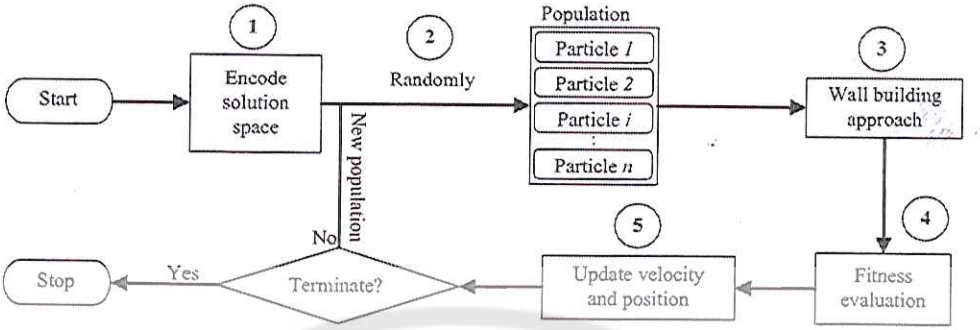


Fig. 15. PSO procedures for solving CPP.

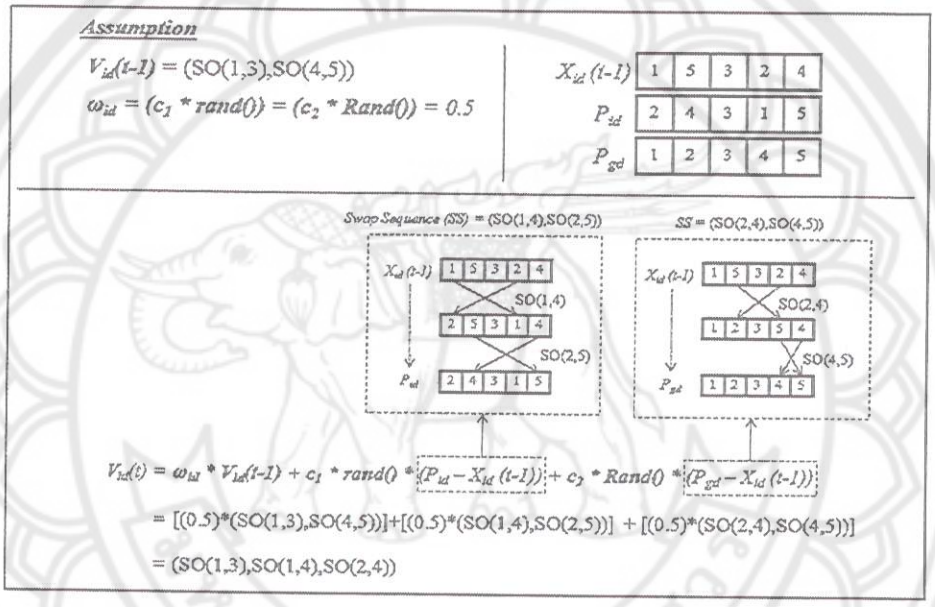


Fig. 16. Updating velocity using the swap operator.

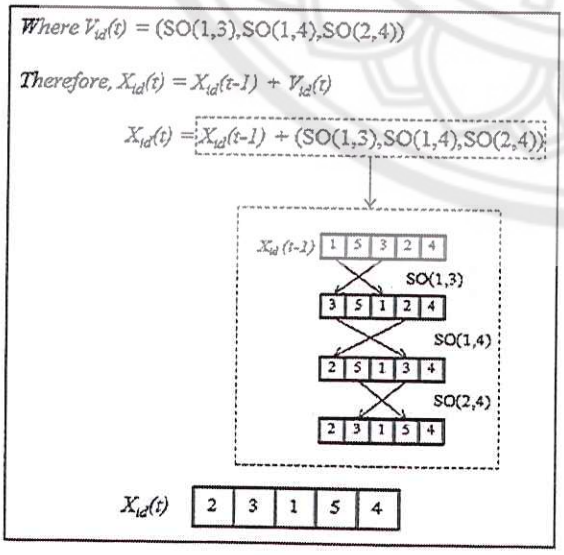


Fig. 17. Updating position using the swap operator.

Table 1
Experimental factors and levels considered.

Factors	Levels	Value
Amount of search (*AI*)	4	50 × 800, 100 × 400, 200 × 200, 400 × 100
Percentage of eliminated antibody (*%B*)	5	5, 10, 25, 50, 75

Table 2
Analysis of variance (ANOVA) on the experimental results.

Source	DF	Sum of squares	Mean square	F	ρ
AI	3	22.4881	7.4960	9.02	0.000
%B	4	22.1709	5.5427	6.67	0.000
Error	592	491.9230	0.8310		
Total	599	536.5820			

crossover (P_c)=0.5; probability of mutation (P_m)=0.15; crossover operator (*COP*)=cycling crossover (*CX*); and mutation operator (*MOP*)=enhanced two operations random swap mutation (*E2ORS*). The PSO parameters—the combination of the number

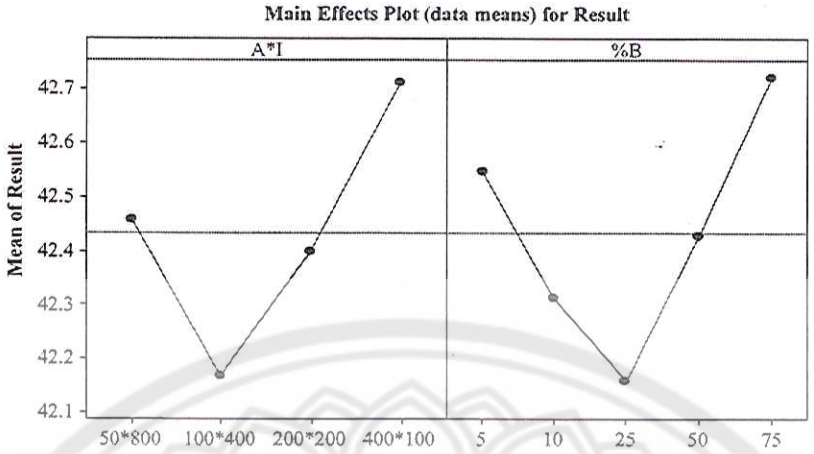


Fig. 18. Main effect plot.

Table 3
Experimental results of all problems and algorithms.

Problem size (boxes)	Algorithms	Quality of solutions obtained (m^2)				Computational time (min)
		Average	Std. dev.	Minimum	Maximum	
100	AIS	42.57633	0.506976	41.791	43.47	1.26
	GA	44.11153	1.547972	41.348	46.753	0.3
	PSO	47.62433	0.772368	45.402	48.645	0.11
250	AIS	117.3667	1.666614	114.34	120.826	2.95
	GA	121.565	2.877986	115.691	126.237	0.63
	PSO	126.0867	0.951880	123.799	127.312	0.51
500	AIS	244.3247	2.133445	239.784	247.622	5.78
	GA	253.7833	2.847689	248.432	257.891	1.16
	PSO	259.083	2.289821	255.459	263.879	1.81
750	AIS	374.6155	2.452509	369.012	378.741	8.35
	GA	386.1964	5.914920	373.066	398.412	1.78
	PSO	397.4422	2.557095	391.443	402.119	2.8
1000	AIS	510.6101	2.896194	503.916	514.455	11.78
	GA	527.3841	7.927805	510.402	539.043	2.35
	PSO	536.6465	2.168516	533.456	539.519	6.53
1250	AIS	643.5694	4.573054	636.387	650.71	15.11
	GA	665.0334	6.414416	653.953	674.787	3.06
	PSO	673.4685	2.906056	666.679	677.759	8.33
1500	AIS	778.1182	3.438976	771.56	783.992	18.63
	GA	796.4086	6.232432	786.694	807.798	3.53
	PSO	811.6292	3.812494	804.015	816.716	15.33
1750	AIS	912.3596	6.539778	901.329	924.195	23.18
	GA	936.169	10.08861	909.166	948.106	4.2
	PSO	949.215	4.229002	939.728	954.627	18.95
2000	AIS	1055.614	6.467699	1040.556	1063.499	27.3
	GA	1080.734	10.38829	1064.339	1095.981	5.38
	PSO	1090.751	3.654231	1084.96	1095.441	21.1
2250	AIS	1194.653	7.932795	1175.729	1205.751	31.5
	GA	1216.282	9.361185	1194.847	1231.695	6.05
	PSO	1233.569	2.747422	1229.804	1237.641	27.7
2500	AIS	1332.626	4.262932	1322.889	1339.844	35.48
	GA	1363.61	11.47981	1342.276	1383.296	6.98
	PSO	1372.217	5.347195	1363.085	1384.395	35.3
2750	AIS	1469.038	6.878416	1458.861	1480.152	40.6
	GA	1500.733	8.700060	1485.827	1519.374	7.95
	PSO	1517.065	3.861985	1507.177	1522.606	42.33
3000	AIS	1609.318	5.508595	1599.416	1616.136	44.03
	GA	1638.646	6.178088	1630.189	1650.212	8.68
	PSO	1658.035	2.302658	1652.915	1661.022	49.68
3250	AIS	1747.702	5.369967	1739.149	1761.039	47.2
	GA	1782.168	9.204404	1762.66	1799.979	9.46
	PSO	1796.562	4.913563	1786.719	1803.763	56.16
3500	AIS	1883.763	5.058473	1876.484	1891.888	52.46
	GA	1920.577	14.97923	1890.537	1941.368	10.6
	PSO	1937.999	6.260651	1923.661	1944.341	64.7
3750	AIS	2024.503	7.275208	2008.415	2032.737	57.55
	GA	2057.469	10.67214	2041.245	2077.623	11.41
	PSO	2075.818	4.912674	2064.651	2081.677	74.83

Table 3 (*continued*)

Problem size (boxes)	Algorithms	Quality of solutions obtained (m^3)				Computational time (min)
		Average	Std. dev.	Minimum	Maximum	
4000	AIS	2156.155	9.956512	2132.238	2172.259	61.21
	GA	2200.024	15.64693	2172.482	2220.634	12.63
	PSO	2214.67	6.479133	2201.986	2224.687	87.1
4250	AIS	2302.487	6.676512	2291.792	2313.918	64.96
	GA	2338.150	12.19414	2312.837	2357.699	13.9
	PSO	2351.926	6.832332	2339.052	2360.942	95.8
4500	AIS	2442.967	6.251996	2434.851	2454.509	69.96
	GA	2474.944	14.21576	2450.443	2502.755	14.38
	PSO	2497.468	6.263813	2488.008	2506.95	108.48
4750	AIS	2578.646	6.316946	2571.152	2593.184	74.33
	GA	2617.120	13.49513	2587.508	2637.881	15.65
	PSO	2643.046	3.192388	2637.799	2647.892	119.33
5000	AIS	2722.148	10.62053	2698.43	2740.248	78.88
	GA	2768.881	11.02224	2750.518	2786.485	16.23
	PSO	2781.563	7.462582	2762.627	2789.458	133.38

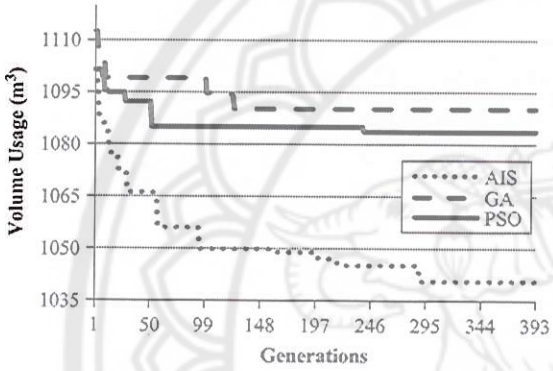


Fig. 19. Progress graphs of each algorithm from the large problem.

of particles and the number of iterations, inertia weight, self-cognition rate and social learning rates were set at 100×25, 0.5, 0.8 and 0.5, respectively. The adjustment operator was adopted for repositioning particles during iteration. For each algorithm, the computational runs were replicated 15 times with different random seeds. The experimental results for 21 different problems were analysed in terms of the mean, standard deviation (Std. dev.), minimum and maximum volume utilisation, and computational time as show in Table 3. It can be seen that the mean solutions obtained by AIS were better than those produced by GA and PSO for all problem sizes. Considering the terms of both minimum and maximum volume utilisations, AIS produced better results than GA for all problem sizes (except the smallest problem with 100 boxes). However, the average computational time taken by AIS was at least four times longer than by GA. The average computational time taken by PSO increased dramatically when the problem size was larger. This was caused by the box adjusting operation conducted in the particle repositioning process of PSO.

In order to study the progress of searching, the 'best so far' solutions obtained by the algorithms for a problem with 2000 heterogeneous rectangular boxes were investigated as shown in Fig. 19. AIS converged more quickly and produced better results than GA and PSO.

7. Conclusions

This paper has described successful development and application of a tool incorporating an Artificial Immune System (AIS), Particle Swarm Optimisation (PSO) and a Genetic Algorithm (GA) for solving the multiple container packing problems (MCPP). A two-step sequential experiment was designed and conducted to identify the best parameter configuration of the algorithms for solving 21 benchmark problems, all of which involved the packing of heterogeneous rectangular boxes into a set of standard marine containers. An analysis of variance found that the AIS parameters amount of search (*AI*) and the percentage of eliminating antibodies (*%B*) were statistically significant with a 95% confidence interval. A main effect plot found that values of 100×400 and 25% performed the best. In the sequential experiment, it was found that the average best-so-far solutions obtained from AIS were better than those produced by GA and PSO for all problem sizes. The convergence graph indicated that the best-so-far result obtained from AIS decreased quicker than those from GA and PSO. However, the average computational times taken by AIS were at least four times longer than byGA. Further work could be done that could consider a two-stage approach of applying a clustering method, e.g. *k*-mean for grouping the box sizes first before considering the packing sequence of boxes into containers.

Acknowledgements

The first author would like to thank the Office of the Higher Education Commission, Thailand, for granting a research fund under the Strategic Scholarships Program for Frontier Research Network. This work was also supported by the Naresuan University Research Fund under the grant number R2554B093.

References

Abbas, A.K., Lichtman, A.H., Pober, J.S., 2000. Cellular and Molecular Immunology 4th edition W.B. Saunders, Philadelphia.

Agarwal, R., Tiwari, M.K., Mukherjee, S.K., 2007. Artificial immune system based approach for solving resource constraint project scheduling problem. International Journal of Advanced Manufacturing Technology 34, 584–593.

Beasley, J.E., 1985. An exact two-dimensional non-guillotine cutting tree-search procedure. Operations Research 33, 49–64.

Bischoff, E.E., Marriott, M.D., 1990. A comparative evaluation of heuristics for container loading. European Journal of Operational Research 44, 267–276.

Bortfeldt, A., Gehring, H., 2001. A hybrid genetic algorithm for the container loading problem. European Journal of Operational Research 131, 143–161.

Bortfeldt, A., Gehring, H., Mack, D., 2003. A parallel tabu search algorithm for solving the container loading problem. Parallel Computing 29, 641–662.

Burnet, F.M., 1959. The Clonal Selection Theory of Acquired Immunity. Vanderbilt University Press, Nashville.

Chen, C.S., Lee, S.M., Shen, Q.S., 1995. An analytical model for the container loading problem. European Journal of Operational Research 80, 68–76.

Christensen, S.G., Rousoe, D.M., 2009. Container loading with multi-drop constraints. International Transactions in Operational Research 16, 727–743.

Cuiru, W., Jiangwei, Z., Jing, Y., Chaoju, H., Jun, L., 2005. A modified particle swarm optimization algorithm and its application for solving traveling salesman problem. In Proceedings of the International Conference on Neural Networks and Brain, 2005, ICNN&B '05, pp. 689–694.

Dasgupta, D., 2006. Advances in artificial immune systems. IEEE Computational Intelligence Magazine 1, 40–49.

Davies, A.P., Bischoff, E.E., 1999. Weight distribution considerations in container loading. European Journal of Operational Research 114, 509–527.

de Castro, L.N., Von Zuben, F.J., 2000. The Clonal Selection Algorithm with engineering applications. In: Proceedings of the Workshop on Artificial Immune Systems and Their Applications, Las Vegas, pp. 36–37.

Delves, P., Martin, S., Burton, D., Roitt, I., 2006. Roitt's Essential Immunology. Wiley-Blackwell, London.

Dyckhoff, H., 1990. A typology of cutting and packing problems. European Journal of Operational Research 44, 145–159.

Egeblad, J., Pisinger, D., 2009. Heuristic approaches for the two- and three-dimensional knapsack packing problem. Computers & Operations Research 36, 1026–1049.

Engin, O., Doyen, A., 2004. A new approach to solve hybrid flow shop scheduling problems by artificial immune system. Future Generation Computer Systems 20, 1083–1095.

Farmer, J.D., Packard, N.H., Perelson, A.S., 1986. The immune system, adaption, and machine learning. Physica D 22, 187–204.

Forrest, S., Perelson, A.S., Allen, L., Cherukuri, R., 1994. Self-nonself discrimination in a computer. In: Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, California, pp. 202–212.

Garrett, S.M., 2005. How do we evaluate Artificial Immune Systems? Evolutionary Computation 13, 145–178.

Gendreau, M., Iori, M., Laporte, G., Martello, S., 2006. A tabu search algorithm for a routing and container loading problem. Transportation Science 40, 342–350.

George, J.A., Robinson, D.F., 1980. A heuristic for packing boxes into a container. Computers & Operations Research 7, 147–156.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimisation and Machine Learning. Addison-Wesley, Reading, MA.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimisation. In: Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, pp. 1942–1948.

Kennedy, J., Eberhart, R.C., Shi, Y., 2001. Swarm Intelligence, San Francisco. Morgan Kaufmann.

Lee, Y.H., Kang, J., Ryu, K.R., Kim, K.H., 2005. Optimization of container load sequencing by a hybrid of ant colony optimization and tabu search. In: Advances in Natural Computation, Part 2. Proceedings 3611, pp. 1259–1268.

Matzinger, P., 2002. The danger model: a renewed sense of self. Science 296, 301–305.

Morabito, R., Arenales, M., 1994. An AND/OR-graph approach to the container loading problem. International Transactions in Operational Research 1, 59–73.

Murphy, K., Travers, P., Walport, M., 2007. Janeway's Immunobiology 7th edition Garland Science. London.

Ngoi, B.K.A., Tay, M.L., Chua, E.S., 1994. Applying spacial representation techniques to the container packing problem. International Journal of Production Research 32, 111–123.

Pisinger, D., 2002. Heuristics for the container loading problem. European Journal of Operational Research 141, 382–392.

Pongcharoen, P., Stewardson, D.J., Hicks, C., Braiden, P.M., 2001. Applying designed experiments to optimise the performance of genetic algorithms used for scheduling complex products in the capital goods industry. Journal of Applied Statistics 28, 441–455.

Pongcharoen, P., Chainate, W., Pongcharoen, S., 2008. Improving Artificial Immune System performance: inductive bias and alternative mutations. In: Artificial Immune Systems, Proceedings 5132, pp. 220–231.

Soak, S.M., Lee, S.W., Yeo, G.T., Jeon, M.G., 2008. An effective evolutionary algorithm for the multiple container packing problem. Progress in Natural Science 18, 337–344.

Thapatsuwan, P., Chainate, W., Pongcharoen, P., 2007. Improving packing efficiency for shipping container. In: Proceedings of the 24th South East Asia Regional Computer Conference, Bangkok, Thailand.

Thapatsuwan, P., Pongcharoen, P., Chainate, W., 2006. Investigation of Genetic Algorithm parameters and comparison of heuristic arrangements for container packing problem. KMITL Science Journal 6, 274–284.

Thapatsuwan, P., Sepsirisuk, J., Chainate, W., Pongcharoen, P., 2009. Modifying Particle Swarm Optimisation and Genetic Algorithm for solving Multiple Container Packing Problems. In: Proceedings of the 2009 International Conference on Computer and Automation Engineering, pp. 137–141.

Tsai, J.T., Ho, W.H., Liu, T.K., Chou, J.H., 2007. Improved immune algorithm for global numerical optimization and job-shop scheduling problems. Applied Mathematics and Computation 194, 406–424.

Wang, K.P., Huang, L., Zhou, C.G., Pang, W., 2003. Particle Swarm Optimization for Traveling Salesman Problem. In: Proceedings of the Second International Conference on Machine Learning and Cybernetics, Xian, pp. 1583–1585.

Wascher, G., HauBner, H., Schumann, H., 2007. An improved typology of cutting and packing problems. European Journal of Operational Research 183, 1109–1130.