รายงานวิจัยฉบับสมบูรณ์

โครงการ การลดต้นทุนการดำเนินงานในมหาวิทยาลัยให้น้อยที่สุด ด้วยวิธีการหาค่าเหมาะสมที่สุดแบบได้รับแรงดลใจมาจากสัตว์

โดย ดร.ธัชชัย เทพกรณ์ และคณะ

กันยายน 2562

สัญญาเลขที่ MRG6080066

รายงานวิจัยฉบับสมบูรณ์

โครงการ การลดต้นทุนการดำเนินงานในมหาวิทยาลัยให้น้อยที่สุดด้วยวิธีการหาค่าเหมาะสมที่สุดแบบได้รับแรงดลใจมาจากสัตว์

คณะผู้วิจัย **สังกัด**

1. ดร.ธัชชัย เทพกรณ์ คณะเทคโนโลยีอุตสาหกรรม มหาวิทยาลัยราชภัฏพิบูลสงคราม

2. ผู้ช่วยศาสตราจารย์ ดร.ภูพงษ์ พงษ์เจริญ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

สนับสนุนโดยสำนักงานคณะกรรมการการอุดมศึกษา และสำนักงานกองทุนสนับสนุนการวิจัย

บทคัดย่อ

รหัสโครงการ : MRG6080066

ชื่อโครงการ : การลดต้นทุนการดำเนินงานในมหาวิทยาลัยให้น้อยที่สุดด้วยวิธีการหาค่าเหมาะสมที่สุดแบบได้รับแรงดลใจมาจากสัตว์

ชื่อนักวิจัย : 1. ดร.ธัชชัย เทพกรณ์ คณะเทคโนโลยีอุตสาหกรรม มหาวิทยาลัยราชภัฏพิบูลสงคราม
2. ผู้ช่วยศาสตราจารย์ ดร.ภูพงษ์ พงษ์เจริญ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

E-mail Address: thatchai.t@psru.ac.th

ระยะเวลาโครงการ: 24 เดือน

โปรแกรมช่วยในการจัดตารางเรียนตารางสอนโดยประยุกต์ใช้วิธีการคุกคูเสิร์ช (CST) ได้ถูกพัฒนาขึ้นมาเพื่อแก้ปัญหาการจัดตารางสอนในระดับอุดมศึกษาโดยพิจารณาการจัดตารางเรียนตารางสอนแบบคำนึงถึงต้นทุนการดำเนินการรวมของมหาวิทยาลัยที่น้อยที่สุด ซึ่งประกอบไปด้วย ต้นทุนการใช้ห้องเรียนแต่ละห้อง ต้นทุนการจ้างอาจารย์ และต้นทุนการทำความสะอาดหรือจัดเตรียมห้อง โปรแกรม CST จะประกอบไปด้วยกลยุทธ์การเดินสุ่ม 2 รูปแบบสำหรับวิธีการคุกคูเสิร์ช (CS) คือ แบบ Lévy Flights (CSLF) และแบบ Gaussian Random Walks (CSGRW) นอกจากนี้วิธีการค้นหาแบบเฉพาะพื้นที่ (Local Search: LS) ประกอบด้วย วิธีการ Insertion Operator (IO) และวิธีการ Exchange Operator (EO) ได้ถูกนำมาผสมผสาน (Hybridisation) กับวิธีการ CS อีกด้วย ปัญหาการจัดตารางสอนจากข้อมูลจริงของมหาวิทยาลัยนเรศวรได้ถูกนำมาออกแบบเป็น 11 โจทย์ปัญหาสำหรับใช้ในการทดสอบประสิทธิภาพของโปรแกรม CST เครื่องมือทางสถิติขั้นสูงสำหรับการออกแบบการทดลองและการวิเคราะห์ผลถูกนำมาใช้ในการค้นหาและวิเคราะห์อิทธิพลค่าพารามิเตอร์ของวิธีการ CS ก่อนทำการสรุปค่าพารามิเตอร์ที่เหมาะสมในทุกโจทย์ปัญหา ผลการทดลองพบว่า วิธีการ CSLF มีประสิทธิภาพดีกว่าวิธีการ CSGRW ในโจทย์ปัญหาส่วนใหญ่อย่างไม่มีนัยสำคัญทางสถิติ ยิ่งไปกว่านั้น กลยุทธ์ LS ทั้งสองวิธีสามารถปรับปรุงประสิทธิภาพของวิธีการ CSLF ได้ทั้งในด้านคุณภาพของคำตอบ ความเร็วในการลู่เข้าสู่คำตอบที่ดี รวมถึงเวลาที่ใช้ในการประมวลผล

คำหลัก: การจัดตารางสอน, เมต้าฮิวริสติกส์, วิธีการคุกคูเสิร์ช, การออกแบบการทดลอง

Abstract

Project Code: MRG6080066

Project Title: Minimising University Operating Costs Using Animal-inspired Optimisation Algorithms

Investigator: 1. Dr.Thatchai Thepphakorn,
Faculty of Industrial Technology, Pibulsongkram Rajabhat University.
2. Assistant Professor Dr.Pupong Pongcharoen,
Faculty of Engineering, Naresuan University

E-mail Address: thatchai.t@psru.ac.th

Project Period: 24 months

A new cuckoo search based timetabling (CST) tool has been developed for university courses scheduling based on minimising the total operation costs: (i) lecturing costs; (ii) overhead costs; and (iii) setup and cleaning costs. The CST program was consisted of two random walks strategies for the cuckoo search (CS) algorithm including the CS via Lévy Flights (CSLF) and the CS via Gaussian Random Walks (CSGRW). Two local search (LS) strategies including Insertion Operator (IO) and Exchange Operator (EO) were hybridised with the CS algorithm. This work, real-world course timetabling data obtained from the Naresuan University were designed for eleven instances and tested by the CST tool. Advance statistical tools for experimental design and analysis were used to investigate and analyse the factor influence of this system and conclude the appropriate parameter setting of the proposed method for all problems. The CSLF insignificantly outperformed the CSGRW for most cases. Moreover, both LS strategies dramatically improved the CSLF performances in terms of solution quality, convergence speeds, and execution times.

Keywords: course timetabling, metaheuristics, cuckoo search, experimental design

Executive Summary

Course timetabling problem generally arises every academic semester in educational institutions and it usually have a lot of data. For example, the current real-world timetabling data for Naresuan University (NU) consisted of 16 faculties, 5 colleges, 37 departments, 22,200 students, and 1,400 lecturers, approximately. This problem can be solved either by: (i) academic staff; (ii) semi-automatically timetabling tool; and (iii) automatically timetabling tool. Solving large course timetabling problems in colleges or universities using the first two methods is extremely difficult and may requires a group of people to work for several weeks because of high numbers of courses, students, lecturers, classrooms, and constraints involved.

A university consists of many sub course timetabling problems generated from the central/university timetabling level, faculty timetabling level, and department timetabling level. Generally, the central timetabling level is the most important and must be operated first. Because that level relates with many mandatory courses (such as English subject, Thai subject, and etc.) enrolled by lots of students across many faculties within a university. After that, the remaining timetabling levels are allowed to schedule the remaining courses (most of major courses for curricula) enrolled by senior students within a faculty or department responded. It can be seen that there are many timetabling steps and staff required for constructing course timetables due to high number of faculties and departments within a university. All timetabling levels cannot schedules concurrently that affect long waiting times from central timetabling to faculty and department timetabling. Moreover, the current timetabling tool for the NU is semi-automatic program, in which it cannot construct the optimal course timetable according to all user preferences.

The general constraints in course timetabling can be classified into two types: hard constraints (HC) and soft constraints (SC). HC are the most important and must be satisfied to have a feasible timetable. During course scheduling, infeasible or impracticable timetables are often generated because of variety and large number of HCs. Those timetables must be reschedule until receiving the feasible timetables. Moreover, course structures and special requirements found in real-world course timetabling problems can increase the difficulty and complexity to find a practicable or feasible timetables. For examples, a course required multiple sections and taught by the same multiple teachers must be taught only one section at the same time, all lecturers and students must be available for a lecture to be scheduled. Unfortunately, these special constraints have been regularly found in almost colleges and universities in Thailand. Therefore, investigating a feasible or practicable timetable, subject to all HC constraints, for large course timetabling data (including high numbers of courses, students, lecturers, classrooms) without using automatically course timetabling tools is very difficult or impossible works within the acceptable times.

Research works considering the objective function that related with SC in term of minimal total operating costs (such as lecturing costs, overhead or classroom leasing costs etc.) have been rarely

reported from literature. The overhead costs (classrooms leasing) at the Universidad de Chile was greatly improved by using a course scheduling system, called *eClasSkeduler* program. In addition, the lecturing costs at Universidad de La Sabana in Colombia was enhanced by using integer linear programming method. However, there is no any research work considering both overhead and lecturing costs simultaneously to solve course timetabling problems.

Nowadays, automatically course timetabling tools based on mathematical models and algorithms, especially for animal-inspired optimisation methods, are becoming increasingly effective at constructing timetables to the desired specification. There are commercially available timetabling program packages but most of them are expensive, inconvenienced training, and maintenance (such as Mimosa scheduling program prices for site license based on the number of students, 800 EUR - 8,000 EUR, http://www.mimosasoftware.com/prices.html). Due to the difference of cultures and traditions among countries, the foreign software packages may not be suitable to solve the university course timetabling problems in Thailand. The examples of special characteristics that are always found in the real-life course timetabling problem in Thailand such as multiple lecturers per a course, multiple sections per a course, video conference teaching, location (or building) and classroom bookings, day and period bookings, split and joint classes between lecture and laboratory within a course. Therefore, developing a tailor-made course timetabling tool is an appropriate choice for the specific problems.

The proposed automatically course timetabling program not only deals with manpower, human errors, scheduling times, outsourcing costs, impractical timetables, optimal course timetable, resource operating costs, and specific timetabling problems but can also be adapted and applied to other educational institutes in Thailand, 156 (public and private) institutes responded by the Office of the Higher Education Commission (OHEC) (http://www.mua.go.th/) and 910 (public and private) institutes responded by the Office of the Vocational Education Commission (OVEC) (http://www.vec.go.th/). The following outcomes are to: reduce a lot of government fund of Thailand to provide or buy the automatically course timetabling tools from other countries for all educational institutes; increase educational resource utilisation for each institute; and reduce many operating costs in colleges and universities. Besides saving lots of government fund allocated to all public colleges and universities in Thailand every year, the autonomous, public, and private universities can survive sustainably by themselves in the future. Educational standards, rankings, and competitive capability of colleges and universities in Thailand would be eventually increased due to effective course timetabling affects resource utilisation as well as staff and student satisfaction.

สารบัญ

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญของปัญหา

การจัดตารางการศึกษาเป็นกิจกรรมที่เกิดขึ้นเป็นประจำทุกภาคการศึกษาในทุกสถาบันการศึกษา โดยมีความเกี่ยวข้องกับการจัดสรรทรัพยากรทางการศึกษาที่มีอยู่อย่างจำกัดภายใต้ข้อบังคับต่างๆ ลงในช่วงเวลาที่มีความเหมาะสมและเป็นไปตามวัตถุประสงค์มากที่สุด [1] การสร้างตารางการศึกษา (เช่น ตารางเรียน ตารางสอน ตารางการใช้ห้องเรียน เป็นต้น) ที่มีคุณภาพดีย่อมทำให้การดำเนินการเรียนการสอนในสถาบันการศึกษาเกิดประสิทธิภาพที่ดีเช่นกันในแต่ละเทอม ดังนั้นตลอดระยะเวลา 50 กว่าปีที่ผ่านมาจนถึงปัจจุบัน การแก้ปัญหาการจัดตารางการศึกษาจึงได้รับความสนใจจากนักวิจัยจำนวนมากทั้งในกลุ่มของการวิจัยการดำเนินงาน (Operations research: OR) และในกลุ่มของความฉลาดประดิษฐ์ (Artificial intelligent: AI) และยังคงได้รับความสนใจเพิ่มมากขึ้นอย่างต่อเนื่องในช่วงหลายสิบปีหลัง [2]

การจัดตารางเรียนตารางสอนในสถาบันระดับอุดมศึกษา เช่น วิทยาลัย (College) มหาวิทยาลัย (University) เป็นต้น ปกติแล้วจะมีขนาดของปัญหาที่ใหญ่มาก มีจำนวนข้อบังคับและข้อจำกัดมากกว่าการจัดตารางการศึกษาในสถาบันการศึกษาขั้นพื้นฐาน เช่น โรงเรียนมัธยมศึกษา (Secondary school) เป็นต้น ซึ่งความซับซ้อนของปัญหาและความยุ่งยากในการจัดตารางสอนในระดับอุดมศึกษาจะมีมากกว่า [3] นอกจากนี้วิชาเรียนวิชาสอนในระดับอุดมศึกษาจะมีทั้งแบบวิชาบังคับ (Mandatory courses) และวิชาเลือก (Elective courses) [4] โดยนิสิตในชั้นเดียวกันแต่ถ้ามีวิชาเลือกเรียนที่แตกต่างกันจะทำให้ตารางเรียนของนิสิตแต่ละคนแตกต่างกันออกไปตามวิชาที่เลือกเรียน ส่งผลให้การจัดตารางเรียนในชั้นดังกล่าวมีความยุ่งยากและซับซ้อนมากขึ้น

ข้อบังคับ (Constraints) ของปัญหาการจัดตารางเรียนตารางสอนโดยทั่วไปแล้วมี 2 ประเภทใหญ่คือ ข้อบังคับหลัก (Hard constraints: HC) และข้อบังคับรอง (Soft constraints: SC) [5-8] โดยที่เป้าประสงค์หลักของการแก้ปัญหาการจัดตารางเรียนตารางสอนคือ การค้นหาตารางเรียนตารางสอนที่ไม่มีการละเมิดข้อบังคับหลักและมีจำนวนการละเมิดข้อบังคับรองที่น้อยที่สุด [6, 9, 10] ในฟังก์ชันเป้าประสงค์ของการแก้ปัญหาการจัดตารางเรียนตารางสอนสามารถจำแนกข้อบังคับรองที่พบได้หลายประเภท ประกอบด้วย 1) แบบ Unary constraints เช่น การนับจำนวนครั้งในการจัดวิชาลงในคาบเวลาห้ามจัดตารางสอน (Unpermitted or booking periods) [11] เป็นต้น 2) แบบ Binary constraints เช่น การนับจำนวนครั้งการชนกันของตารางสอน (Event clash constraints) [12] เป็นต้น 3) แบบ Capacity constraints เช่น การนับจำนวนเก้าอี้ที่ไม่เพียงพอทั้งหมดในการจัดตารางสอน [13] เป็นต้น 4) แบบ Event spread constraints เช่น การนับจำนวนครั้งที่มีการเรียนการสอนต่อเนื่องเกินสองคาบติดกัน (Spreading-out or clumping-together) [14] เป็นต้น 5) แบบ Agent constraints เช่น การนับจำนวนครั้งที่มีรูปแบบการสอน (Lecturing format) ไม่เป็นไปตามที่อาจารย์ต้องการ (People preferences or requirements) [1] เป็น

ต้น และ 6) แบบ Stability or movement constraints เช่น การนับจำนวนครั้งที่วิชาเดียวกันแต่จัดตารางให้ใช้ห้องเรียนต่างกันทุกครั้ง [15] เป็นต้น

อย่างไรก็ตามจากการทบทวนวรรณกรรมจะพบว่า ยังมีข้อบังคับรองอีกประเภทคือ แบบ Cost constraints ซึ่งงานวิจัยที่พิจารณาฟังก์ชันเป้าประสงค์จากข้อบังคับรองที่เกี่ยวข้องกับการจัดตารางเรียนตารางสอนโดยเกิดค่าดำเนินการรวมจากการใช้ทรัพยากรทางการศึกษาที่น้อยที่สุด (Minimising total operating costs) นั้นพบได้น้อยมากเพียง 2 บทความจากฐานข้อมูล ISI web of knowledge ดังนี้ Seo และคณะ [16] ได้ทำการปรับปรุงระบบการจัดตารางสอนในองค์กรให้มีความรวดเร็วมากขึ้นโดยการพัฒนาโปรแกรม eClasSkeduler สำหรับใช้ในมหาวิทยาลัย Universidad de Chile โดยพิจารณาต้นทุนการเช่าห้องเรียน (Classrooms leasing/operating costs) ที่น้อยที่สุดในฟังก์ชันเป้าประสงค์ นอกจากนี้ Torres-Ovalle และคณะ [17] ได้แก้ปัญหาการจัดตารางสอนจริงของมหาวิทยาลัย Universidad de La Sabana ที่โคลัมเบียด้วยวิธีการเชิงเส้นแบบจำนวนเต็ม (Integer linear programming) โดยพิจารณาต้นทุนการจ้างอาจารย์ (Lecturing costs) ที่น้อยที่สุดในฟังก์ชันเป้าประสงค์ ดังนั้นในงานวิจัยนี้จึงได้เติมเต็มการพิจารณาฟังก์ชันเป้าประสงค์ในข้อบังคับรอง (SC) แบบ Cost constraints โดยจัดตารางเรียนตารางสอนเพื่อให้เกิดค่าดำเนินการรวมจากการใช้ทรัพยากรทางการศึกษาที่น้อยที่สุด ประกอบด้วย ต้นทุนค่าโสหุ้ย (Overhead costs) เช่น ค่าเช่าห้องเรียน ค่าน้ำ ค่าไฟ เป็นต้น ต้นทุนการจ้างอาจารย์ (Lecturing costs) และต้นทุนการจัดเตรียมหรือทำความสะอาดห้อง (Setup/cleaning costs) ซึ่งยังไม่พบงานวิจัยใดพิจารณาฟังก์ชันเป้าประสงค์ย่อย 3 แบบนี้ร่วมกันมาก่อน

โดยทั่วไปแล้วการจัดตารางเรียนตารางสอนสามารถทำได้โดยใช้มนุษย์ (Manually by academic staff) หรือโดยใช้โปรแกรมแบบกึ่งอัตโนมัติ (Semi-automatically) หรือโดยใช้โปรแกรมแบบอัตโนมัติ (Automatically) ปัจจุบันระบบการจัดตารางเรียนตารางสอนในสถาบันระดับอุดมศึกษาของประเทศไทยส่วนใหญ่จะยังเป็นการจัดตารางเรียนตารางสอนแบบกึ่งอัตโนมัติ ซึ่งเป็นการใช้ซอฟต์แวร์คอมพิวเตอร์มาช่วยเหลือมนุษย์เฉพาะการตรวจสอบการชนกันของตารางเรียนตารางสอนขณะทำการจัดตารางเท่านั้น ในขณะที่การเลือกห้องเรียนและคาบเวลาให้กับแต่ละวิชานั้นยังคงให้มนุษย์เป็นผู้ตัดสินใจทั้งหมด ซึ่งต้องใช้บุคลากรจำนวนมากและเวลาที่นานมากในการเลือกคาบเวลาที่ไม่ชนกันให้กับทุกวิชาของมหาวิทยาลัยในแต่ละเทอม ในขณะที่การจัดตารางเรียนตารางสอนแบบอัตโนมัติจะเป็นการพัฒนาซอฟต์แวร์คอมพิวเตอร์ขึ้นมาช่วยจัดตารางเรียนตารางสอนแทนมนุษย์ทั้งในส่วนของการตัดสินใจเลือกห้องเรียน/เวลาที่เหมาะสมให้กับแต่ละวิชาและการตรวจสอบข้อบังคับหลักต่างๆ ไปพร้อมๆ กัน ซึ่งมีข้อได้เปรียบกว่าวิธีการจัดตารางเรียนตารางสอน 2 แบบแรก คือ เหมาะกับการแก้ปัญหาขนาดใหญ่มากและมีข้อบังคับจำนวนมาก มีความสะดวกและสามารถการจัดตารางเรียนตารางสอนได้รวดเร็วกว่ามาก ใช้บุคลากรในการจัดตารางน้อยกว่ามาก ลดจำนวนขั้นตอนในการจัดตารางเรียนตารางสอนให้เหลือเพียงขั้นตอนเดียว สามารถจัดตารางเรียนตารางสอนตามที่นักเรียนและอาจารย์ต้องการได้

แม้ว่าโปรแกรมจัดตารางเรียนตารางสอนแบบอัตโนมัติของต่างประเทศมีอยู่หลายโปรแกรม แต่โปรแกรมดังกล่าวมีราคาขายที่แพง ไม่สะดวกในด้านการอบรมและการบำรุงรักษา อีกทั้งยังมีความแตกต่างทางด้าน

วัฒนธรรมและประเพณีของการศึกษาในแต่ละประเทศซึ่งมีความเฉพาะ จึงทำให้การประยุกต์ใช้โปรแกรมแบบอัตโนมัติของต่างประเทศกับสถานศึกษาในประเทศไทยอาจจะทำได้ค่อนข้างยาก ดังนั้นการการพัฒนาโปรแกรมในการจัดตารางเรียนตารางสอนระดับอุดมศึกษาแบบอัตโนมัติจึงเป็นอีกทางเลือกในงานวิจัยนี้

นอกจากนี้การปรับปรุงและพัฒนาโปรแกรมในการจัดตารางสอนให้มีประสิทธิภาพสูงขึ้นเป็นอีกประเด็นที่สำคัญ เนื่องจากปัญหาการจัดตารางเรียนตารางสอนในระดับอุดมศึกษามีลักษณะเป็นแบบ Combinatorial Optimisation (CO) [4, 18] และยังถูกจัดเป็นปัญหาแบบ Non-deterministic Polynomial (NP) hard problem [19, 20] หมายความว่า เมื่อขนาดของปัญหาเพิ่มขึ้นเพียงเล็กน้อยแต่เวลาที่ต้องการในการแก้ปัญหาจะเพิ่มขึ้นอย่างทวีคูณ [1] ดังนั้นหลายงานวิจัยในปัจจุบันจึงให้ความสนใจในเรื่องของการนำวิธีการแก้ปัญหาขนาดใหญ่และซับซ้อนที่มีประสิทธิภาพมาประยุกต์ใช้ร่วมกับโปรแกรมสำเร็จรูปมากขึ้น [21]

วิธีการในกลุ่มของเมต้าฮิวริสติกส์ (Metaheuristics) ถูกยอมรับอย่างแพร่หลายแล้วว่าเป็นอีกแนวทางสำหรับแก้ปัญหาแบบ NP-hard problems โดยอาศัยหลักการประมาณค่าในการค้นหาคำตอบที่มีคุณภาพดีภายในเวลาที่ยอมรับได้ [5] สามารถแก้ปัญหาที่มีขนาดใหญ่และมีความซับซ้อนสูงได้อย่างมีประสิทธิภาพและประสิทธิผล [22] วิธีการในกลุ่มนี้นิยมจำแนกได้เป็น 2 กลุ่มใหญ่ คือ วิธีการแบบ Single-solution based metaheuristics (S-meta) และแบบ Population based metaheuristics (P-meta) [18, 22] วิธีการในกลุ่ม P-meta จะได้เปรียบกว่าวิธีการในกลุ่ม S-meta ในด้านการเริ่มต้นจากคำตอบหลายคำตอบ (Population) ไว้สำหรับพัฒนาคำตอบใหม่ๆ โดยคำตอบที่ได้จะมีลักษณะเด่นที่เป็นกลุ่มของคำตอบที่มีความหลากหลาย (Diversification) จึงทำให้วิธีการนี้มีลักษณะเด่นในการค้นหาคำตอบเชิงสำรวจ (Exploration search) เป็นหลัก [22] วิธีการในกลุ่มของ P-meta ชนิดใหม่ๆ ถูกนำเสนอขึ้นมาใหม่อย่างต่อเนื่องในปัจจุบัน เช่น วิธีการ Cuckoo Search [23], วิธีการ Bat Algorithm [24], วิธีการ Firefly Algorithm [25], วิธีการ Krill Herd [26], วิธีการ Flower Pollination Algorithm [27], วิธีการ Invasive Weed Optimisation [28], วิธีการ Gravitational search [29], วิธีการ Intelligent water drop [30], วิธีการ Backtracking optimisation search [31], วิธีการ League championship algorithm [32] เป็นต้น

วิธีการคุกคูเสิร์ช (Cuckoo Search: CS) เป็นหนึ่งในวิธีการเมต้าฮิวริสติกส์แบบ P-meta แบบใหม่ที่ได้แรงดลใจมาจากธรรมชาติ (Nature-inspired algorithms) ถูกพัฒนาขึ้นโดย Yang และ Deb ในปี 2009 [23] วิธีการ CS นี้ได้แนวคิดมาจากพฤติกรรมอันชาญฉลาดในการฝากฟักไข่ของนกกาเหว่าไว้กับรังของนกสายพันธ์อื่นเพื่อความอยู่รอดของเผ่าพันธุ์ ถึงแม้จะเป็นวิธีการที่ค่อนข้างใหม่ แต่ในช่วง 4-5 ปีที่ผ่านมาวิธีการ CS กลับได้รับความนิยมนำมาประยุกต์ใช้แก้ปัญหาการหาค่าที่เหมาะสมที่สุดอย่างแพร่หลายและประสบผลสำเร็จเป็นอย่างดี เช่น ปัญหาการหาค่าที่เหมาะสมที่สุดเชิงตัวเลข (Numerical optimisation problems) [33], ปัญหาการหาค่าที่เหมาะสมที่สุดทางวิศวกรรม (Engineering optimisation problems) [23], ปัญหาการเดินทางของพนักงานขายแบบพื้นผิวทรงกลม (Spherical traveling salesman problems) [34], ปัญหาการจัดตารางการเข้าเวรของพยาบาล (Nurse scheduling problems) [35], ปัญหาการจัดตารางการผลิต (Production scheduling problems) [36] เป็นต้น

นอกจากนี้หลายงานวิจัยยังได้ทำการเปรียบเทียบประสิทธิภาพในการแก้ปัญหาของวิธีการ CS กับวิธีการเมต้าฮิวริสติกส์แบบอื่นอีกด้วย เช่น ปัญหาการหาค่าที่เหมาะสมที่สุดเชิงตัวเลขพบว่า วิธีการ CS มีประสิทธิภาพในการหาคำตอบที่ดีกว่า GA, PSO [37] และ ABC [33] สำหรับปัญหากระบวนการบด (Milling operation) พบว่า วิธีการ CS มีประสิทธิภาพในการหาคำตอบที่ดีกว่าทั้งวิธีการ GA, ACO, AIS และ PSO [38] เป็นต้น ดังนั้นจะเห็นได้ว่าวิธีการ CS เป็นวิธีการหนึ่งที่น่าสนใจมาก เพราะว่านอกจากจะมีจำนวนพารามิเตอร์ที่ต้องปรับค่าให้เหมาะสมน้อยกว่าวิธีการเมต้าฮิวริสติกส์แบบอื่นแล้ว เช่น GA และ PSO [37] เป็นต้น วิธีการ CS ยังมีประสิทธิภาพในการหาคำตอบที่ดีมากเมื่อเปรียบเทียบกับวิธีการเมต้าฮิวริสติกส์แบบอื่นๆ จากการประยุกต์ใช้แก้ปัญหาหลายประเภท อย่างไรก็ตามการประยุกต์ใช้วิธีการ CS แก้ปัญหาการจัดตารางเรียนตารางสอนระดับมหาวิทยาลัย เมื่อได้ทำการสืบค้นในฐานข้อมูลระดับนานาชาติ ISI web of knowledge, Scopus, และ IEEE Xplore ทั้งในส่วนที่เป็นชื่อเรื่อง (Title) บทคัดย่อ (Abstract) และคำสำคัญ (Keyword) จะพบเพียง 1 บทความเฉพาะในฐานข้อมูลของ Scopus เท่านั้น ในขณะที่ฐานข้อมูล ISI web of knowledge และ IEEE Xplore ยังไม่พบงานวิจัยที่ประยุกต์ใช้วิธีการ CS แก้ปัญหาการจัดตารางเรียนตารางสอนระดับมหาวิทยาลัยมาก่อน

ถึงแม้ว่าวิธีการเมต้าฮิวริสติกส์จะมีข้อดีต่างๆ มากมายและได้รับความนิยมอย่างมากในปัจจุบัน แต่วิธีการดังกล่าวก็มีปัญหาบางประการซึ่งส่งผลโดยตรงต่อประสิทธิภาพในการค้นหาคำตอบเช่นกัน เนื่องจากประสิทธิภาพในการหาคำตอบของวิธีการเมต้าฮิวริสติกส์นั้นขึ้นอยู่กับค่าพารามิเตอร์ที่ใช้ [22, 39] การค้นหาและการกำหนดค่าพารามิเตอร์ที่เหมาะสมให้กับวิธีการเมต้าฮิวริสติกส์จึงเป็นสิ่งที่ไม่อาจหลีกเลี่ยงไปได้ ดังนั้นหลายงานวิจัยจึงได้ศึกษาและค้นหาค่าพารามิเตอร์ที่เหมาะสมที่สุดให้กับวิธีการเมต้าฮิวริสติกส์เพื่อให้วิธีการดังกล่าวมีประสิทธิภาพสูงขึ้นกว่าเดิมในการแก้ปัญหา [40-42]

การกำหนดพารามิเตอร์แบบ Parameter tuning นั้น ค่าพารามิเตอร์ต่างๆ ของวิธีการเมต้าฮิวริสติกส์จะถูกกำหนดก่อนที่วิธีการดังกล่าวจะทำการประมวลผล [22] วิธีการกำหนดค่าพารามิเตอร์ในกลุ่มนี้ เช่น วิธีการ Ad hoc selection [43], วิธีการ Adopted approach [13], วิธีการ Best guess approach [44], วิธีการ One-factor-at-a-time [44], วิธีการ Factorial design [44] เป็นต้น อย่างไรก็ตามการค้นหาพารามิเตอร์ที่เหมาะสมด้วย 3 วิธีการแรกนั้น ไม่สามารถรับประกันได้เลยว่าจะทำให้วิธีการเมต้าฮิวริสติกส์สามารถค้นหาคำตอบที่ดีที่สุดได้ [44, 45] ขณะที่วิธีการ One-factor-at-a-time มีข้อเสียเปรียบตรงที่ไม่สามารถศึกษาผลกระทบร่วม (Interaction) ระหว่างปัจจัยได้ สุดท้ายแล้ววิธีการ Factorial design จะเป็นวิธีการที่ถูกต้องและมีประสิทธิภาพมากสุดสำหรับค้นหาค่าพารามิเตอร์ที่เหมาะสมที่สุดให้กับวิธีการเมต้าฮิวริสติกส์ เนื่องจากสามารถจัดการกับพารามิเตอร์จำนวนหลายตัวพร้อมกันได้โดยใช้หลักการออกแบบการทดลองและการวิเคราะห์ทางสถิติ (Experimental design and analysis) เข้ามาช่วย อีกทั้งยังสามารถศึกษาผลกระทบร่วม (Interaction) ระหว่างปัจจัยได้ด้วย [44]

การปรับปรุงประสิทธิภาพของวิธีการเมต้าฮิวริสติกส์ให้ดีขึ้นกว่าเดิมโดยการปรับปรุงกระบวนการ (Modification) ทำงานจัดเป็นอีกวิธีการหนึ่งที่พบได้บ่อย สำหรับปัญหาการจัดตารางเรียนตารางสอนพบว่าหลายงานวิจัยได้ปรับปรุงประสิทธิภาพของวิธีการเมต้าฮิวริสติกส์ในกระบวนการหรือในขั้นตอนที่แตกต่างกัน

ออกไปขึ้นอยู่กับวัตถุประสงค์ของการปรับปรุงในกระบวนการนั้นๆ ซึ่งผลของการปรับปรุงต่างก็ประสบผลสำเร็จเป็นอย่างดี เช่น กระบวนการ Initial solution [46, 47], กระบวนการ Solution evolution [48], กระบวนการ Fitness calculation [19], กระบวนการ Selection process [49], กระบวนการ Solution replacement [10], กระบวนการ Accepted rules [19], กระบวนการ Memory update [50] เป็นต้น

การปรับปรุงประสิทธิภาพของวิธีการเมต้าฮิวริสติกส์ให้ดีขึ้นกว่าเดิมโดยการผสมผสาน (Hybridisation) จัดเป็นอีกวิธีการหนึ่งที่พบได้บ่อย เนื่องจากการแก้ปัญหาการหาค่าที่เหมาะสมที่สุดหลายประเภทในปัจจุบันนั้น การใช้วิธีการเมต้าฮิวริสติกส์เพียงวิธีการเดียวปกติแล้วจะไม่เหมาะกับการนำไปประยุกต์ใช้แก้ปัญหาที่มีความยากมากๆ เพราะผลลัพธ์ที่ได้อาจจะไม่ดีเท่าที่ควร [22] วิธีการ P-meta ผสมผสานกับ S-meta จะเป็นแนวทางที่ได้รับความนิยมมากที่สุดของการผสมผสานและยังเป็นวิธีการที่มีประสิทธิภาพสูงในการค้นหาคำตอบ [22, 51, 52] เพราะว่า จะเป็นการใช้ข้อดีของการค้นหาคำตอบในวงกว้างซึ่งได้จาก P-meta (Exploration) ร่วมกับการใช้ข้อดีของการค้นหาคำตอบในวงแคบซึ่งได้จาก S-meta (Exploitation) ทำให้วิธีการผสมผสานแบบนี้เกิดความสมดุลกันจากหลักการค้นหาคำตอบทั้ง 2 แบบ (Diversification และ Intensification) [22, 51, 52] ดังนั้นวิธีการผสมผสานในกลุ่มนี้จึงได้รับความนิยมนำมาใช้แก้ปัญหาต่างๆ และประสบความสำเร็จเป็นอย่างดี ซึ่งรวมไปถึงปัญหาการจัดตารางการศึกษาด้วย ตัวอย่างเช่น วิธีการ Ant Colony System (ACS) กับ TS [53], วิธีการ GA+LS (Memetic Algorithm: MA) [54], วิธีการ GA+TS [12], วิธีการ PSO+LS [48, 55], วิธีการ Best-worst ACS กับ LS [13] เป็นต้น

1.2 วัตถุประสงค์ของโครงการ

- เพื่อออกแบบและพัฒนาโปรแกรมจัดตารางเรียนตารางสอนแบบอัตโนมัติโดยประยุกต์ใช้วิธีการ Cuckoo Search เพื่อแก้ปัญหาการจัดตารางสอนในระดับอุดมศึกษาแบบคำนึงถึงต้นทุนการดำเนินการรวมของมหาวิทยาลัยที่น้อยที่สุด (Minimising total university operating costs)
- เพื่อวิเคราะห์และเปรียบเทียบประสิทธิภาพของวิธีการ CS ในการแก้ปัญหาการจัดตารางสอน โดยการกำหนดพารามิเตอร์ที่เหมาะสม (Appropriate parameter settings) การปรับปรุงวิธีการโดยการปรับเปลี่ยนกระบวนการ (Modification) และ การปรับปรุงวิธีการโดยการผสมผสาน (Hybridisations)

1.3 ขอบเขตการวิจัย

1. ในงานวิจัยนี้จะพิจารณาเฉพาะในส่วนที่เป็นปัญหาการจัดตารางเรียนตารางสอนในระดับอุดมศึกษา (University course timetabling problem: UCTP) แบบอ้างอิงตามหลักสูตร (Curricula)

2. ในงานวิจัยนี้ โครงสร้างวิชาเรียนวิชาสอนของปัญหา CUCT จะประกอบด้วย กรณีวิชาเรียนวิชาสอนแบบวิชาบังคับและวิชาเลือก (Mandatory/Elective courses) กรณีวิชาเรียนวิชาสอนแบบภาคบรรยายและภาคปฏิบัติ (Lecture/Laboratory) กรณีวิชาเรียนวิชาสอนแบบหนึ่งหมู่เรียนและแบบหลายหมู่เรียน

(Single/Multiple sections) และกรณีวิชาเรียนวิชาสอนแบบมีอาจารย์ผู้สอนคนเดียวและผู้สอนหลายคน (Single/Multiple lecturers)

3. ในงานวิจัยนี้จะพิจารณาข้อมูลการจัดตารางเรียนตารางสอนของคณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร ทั้งในส่วนรายวิชาของคณะหรือสาขาวิชา (Elective courses) และรายวิชาศึกษาทั่วไป (Mandatory courses) ของมหาวิทยาลัย รวมถึงรายวิชาที่รับผิดชอบโดยคณะอื่นๆ แต่มีความเกี่ยวข้องกับอาจารย์หรือหลักสูตรของนิสิตของคณะวิศวกรรมศาสตร์ เช่น วิชาภาษาอังกฤษ วิชาฟิสิกส์ วิชาแคลคูลลัส เป็นต้น เพื่อให้ตารางเรียนตารางสอนของนิสิตทุกคนและอาจารย์ทุกคนในคณะวิศวกรรมศาสตร์มีจำนวนวิชาเรียนวิชาสอนในตารางเรียนตารางสอนครบทุกวิชาในแต่ละเทอมการศึกษา

4. ในงานวิจัยนี้จะพิจารณาข้อมูลการจัดตารางเรียนตารางสอนของคณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร ในเทอม 1 ปีการศึกษา 2555 ซึ่งเป็นข้อมูลที่มีความหลากหลายของหลักสูตรที่เปิดสอนเป็นอย่างมาก โดยมีการเปิดสอนทั้งในระดับปริญญาตรี (ภาคปกติและภาคพิเศษ) ระดับปริญญาโท (ภาคปกติและภาคพิเศษ) และระดับปริญญาเอก

5. ในงานวิจัยนี้จะพิจารณาการจัดตารางสอนให้กับอาจารย์ทุกคนภายในคณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร เป็นหลัก รวมไปถึงตารางสอนของอาจารย์จากคณะอื่นๆ ที่ทำการสอนในบางรายวิชาให้กับนิสิตในหลักสูตรจากคณะวิศวกรรมศาสตร์ เช่น วิชาฟิสิกส์ เป็นต้น อย่างไรก็ตามตารางสอนของอาจารย์จากคณะอื่นๆ จะถูกแสดงเฉพาะรายวิชาทีเกี่ยวข้องกับคณะวิศวกรรมศาสตร์เท่านั้น

6. ในงานวิจัยนี้จะพิจารณาการจัดตารางการใช้ห้องเรียนของคณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร เป็นหลัก รวมไปถึงห้องเรียนจากคณะอื่นๆ ที่ต้องใช้สอนในบางรายวิชาให้กับนิสิตในหลักสูตรจากคณะวิศวกรรมศาสตร์ เช่น ห้องปฏิบัติการฟิสิกส์และห้องปฏิบัติการเคมีของคณะวิทยาศาสตร์ เป็นต้น

7. ในงานวิจัยนี้ วิชาทุกวิชารวมถึงวิชาที่เป็นแบบวิทยานิพนธ์ (Thesis or Dissertation) การศึกษาค้นคว้าอิสระ (Independent study) หรือโครงงานวิจัย (Project) สำหรับนิสิตจากทุกหลักสูตรของคณะวิศวกรรมศาสตร์ สามารถกำหนดได้ว่าจะพิจารณาหรือไม่พิจารณาจัดลงในตารางเรียนตารางสอนได้ในข้อมูลนำเข้า

8. ในงานวิจัยนี้ อาจารย์แต่ละท่านจะถูกกำหนดวิชา (Courses) ที่ต้องสอนไว้ในข้อมูลนำเข้าแล้วก่อนการจัดตารางเรียนตารางสอนในแต่ละเทอม เพื่อให้อาจารย์แต่ละท่านได้เลือกสอนในวิชาที่ตนเองมีความชำนาญ

9. ในงานวิจัยนี้ รายวิชาที่ต้องกำหนดจำนวนอาจารย์ผู้สอนมากกว่า 1 ท่าน (Multiple lecturers) นั้นสามารถกำหนดจำนวนอาจารย์ผู้สอนได้อย่างไม่จำกัดไว้ในข้อมูลนำเข้าแล้วก่อนการจัดตารางเรียนตารางสอน

10. ในงานวิจัยนี้ วิชาพื้นฐานบางวิชาที่จะมีนิสิตลงเรียนเป็นจำนวนมากจะมีการแบ่งหมู่เรียน (Sections) ไว้หลายหมู่เรียน โดยนิสิตในแต่ละหลักสูตรที่จะต้องลงเรียนในวิชาดังกล่าวจะถูกกำหนดหมู่เรียนที่สามารถลงเรียนได้ในขั้นตอนของการมอบหมาย (Assignment) ทรัพยากรแล้วก่อนกระบวนการจัดตารางเรียนตารางสอน เพื่อกระจายจำนวนนิสิตให้สมดุลในแต่ละหมู่เรียนที่เปิดสอน รวมถึงป้องกันนิสิตในบางคณะ

หรือบางหลักสูตรอาจจะไม่สามารถลงทะเบียนเลือกหมู่เรียนหรือเรียนวิชาดังกล่าวได้ในขั้นตอนการลงทะเบียน อันเนื่องมาจากการชนกันของตารางเรียน

11. ในงานวิจัยนี้ วิชาที่มีจำนวนหมู่เรียนหลายหมู่เรียน ผู้จัดตารางสามารถกำหนดให้วิชาดังกล่าวมีการรวมหมู่เรียน (เช่น วิชาบรรยาย เป็นต้น) หรือแยกหมู่เรียน (เช่น วิชาปฏิบัติ เป็นต้น) ได้ โดยข้อมูลนี้จะถูกระบุไว้ในส่วนของข้อมูลนำเข้าแล้วก่อนการจัดตารางเรียนตารางสอน

12. ในงานวิจัยนี้ นิสิตหลักสูตรเดียวกันซึ่งมีเรียนวิชาพื้นฐานเหมือนกันแต่สามารถลงเรียนในวิชาเลือกต่างกันนั้น นิสิตในหลักสูตรดังกล่าวจะถูกกำหนดรหัสของหลักสูตรย่อยเพื่อแสดงถึงตารางเรียนที่แตกต่างกันออกไปในหลักสูตรเดียวกัน โดยกระบวนการนี้จะถูกระบุไว้ในส่วนของข้อมูลนำเข้าแล้วก่อนการจัดตารางเรียนตารางสอน

13. ในงานวิจัยนี้ การกำหนดจำนวนนิสิตที่เปิดรับของวิชาเลือก (Elective courses) ในแต่ละวิชาจะอยู่ในส่วนของขั้นตอนการมอบหมายทรัพยากรก่อนกระบวนการจัดตารางเรียนตารางสอน

14. ในงานวิจัยนี้จะไม่พิจารณาจัดตารางเรียนของนิสิตที่ตกแผนการเรียน เช่น นิสิตปริญญาตรีปี 5-8 เป็นต้น เนื่องจากนิสิตกลุ่มนี้ปกติจะมีจำนวนน้อยและสามารถลงทะเบียนเรียนเพิ่มเติมได้ในช่วงเพิ่มถอนรายวิชาหลังจากเปิดภาคเรียนแล้ว

15. ในงานวิจัยนี้จะใช้ข้อมูลห้องเรียนสำหรับจัดตารางเรียนตารางสอนจาก 4 อาคารเรียนหลักและ 4 อาคารปฏิบัติการ (Shop) ของคณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร ประกอบไปด้วย อาคารเรียนคณะวิศวกรรมศาสตร์ สาขาวิศวกรรมโยธา (CE) สาขาวิศวกรรมไฟฟ้า (EE) สาขาวิศวกรรมอุตสาหการ (IE) และอาคารเรียนรวมคณะวิศวกรรมศาสตร์ (EN) นอกจากนี้วิชาพื้นฐาน (Mandatory courses) ที่นิสิตคณะวิศวกรรมศาสตร์ต้องไปเรียนร่วมกับคณะอื่น เช่น วิชาภาษาไทย วิชาภาษาอังกฤษ เป็นต้น งานวิจัยนี้จะมีการพิจารณาข้อมูลอาคารเรียนภายนอกเพิ่มเติมเฉพาะรายวิชาที่เกี่ยวข้อง (เช่น อาคารเรียน QS อาคารเรียน SC เป็นต้น)

16. ในงานวิจัยนี้จะมีการกำหนดช่วงเวลาห้ามจัดการเรียนการสอนของอาจารย์แต่ละท่านได้ อันเนื่องมาจากอาจารย์บางท่านอาจมีภารกิจที่ไม่สามารถดำเนินการสอนได้ในบางวันหรือบางช่วงเวลา อย่างไรก็ตามข้อกำหนดในด้านวันและเวลาที่ไม่สามารถสอนได้ของอาจารย์แต่ละท่านสามารถปรับเปลี่ยนได้อย่างอิสระในส่วนของข้อมูลนำเข้า

17. ในงานวิจัยนี้จะมีการกำหนดช่วงเวลาเรียนของนิสิตในแต่ละหลักสูตรที่สามารถจัดตารางเรียนได้แตกต่างกันออกไป โดยในระดับปริญญาตรี ปริญญาโทและปริญญาเอกภาคปกติจะกำหนดให้อยู่ในช่วงวันจันทร์-ศุกร์เวลา 8.00-17.00 น. ในระดับปริญญาตรีภาคพิเศษจะกำหนดให้อยู่ในช่วงวันจันทร์-ศุกร์เวลา 13.00-22.00 น. และวันเสาร์-อาทิตย์เวลา 8.00-22.00 น. และในระดับปริญญาโทภาคพิเศษจะกำหนดให้อยู่ในช่วงวันเสาร์-อาทิตย์เวลา 8.00-22.00 น. อย่างไรก็ตามข้อกำหนดในด้านวันและเวลาที่สามารถจัดการเรียนได้ของแต่ละหลักสูตรสามารถปรับเปลี่ยนได้อย่างอิสระในส่วนของข้อมูลนำเข้า

18. ในงานวิจัยนี้จะพิจารณาต้นทุนค่าโสหุ้ย (Overhead costs) ที่เกิดจากการใช้ห้องเรียนแต่ละห้องซึ่งมีความเกี่ยวข้องกับข้อบังคับ SC แบบ Room suitability โดยจะเป็นค่าใช้จ่ายแบบถัวเฉลี่ย/ชั่วโมง เช่น

ค่าไฟ ค่าน้ำ ค่าเช่าห้อง เป็นต้น และจะผันแปรไปตามขนาดของห้องเรียน (เช่น ห้องบรรยาย ห้อง SLOP หอประชุม เป็นต้น) ประเภทของห้องเรียน (ห้องบรรยาย ห้องปฏิบัติการ) รวมถึงวันและเวลาที่จัดการเรียนการสอน นอกจากนี้ต้นทุนดังกล่าวจะไม่กระทบต่อคุณภาพการศึกษาในเชิงลบ

19. ในงานวิจัยนี้จะพิจารณาต้นทุนที่เกิดจากการทำความสะอาดห้องเรียนแต่ละห้อง (Setup or cleaning costs) ซึ่งจะมีความเกี่ยวข้องกับข้อบังคับ SC แบบ Consecutive lectures โดยจะเป็นค่าทำความสะอาดหรือจัดเตรียมห้องหลังการเรียนการสอนแบบถัวเฉลี่ย/ครั้ง และจะผันแปรไปตามขนาดของห้องเรียน ประเภทของห้องเรียน รวมถึงวันเวลาที่ทำความสะอาดหลังจากเรียนการสอน นอกจากนี้ต้นทุนดังกล่าวจะไม่กระทบต่อคุณภาพการศึกษาในเชิงลบ

20. ในงานวิจัยนี้จะพิจารณาต้นทุนที่เกิดจากค่าตอบแทนอาจารย์ผู้สอนแต่ละท่าน (Lecturing costs) ซึ่งจะมีความเกี่ยวข้องกับข้อบังคับ SC แบบ Timeslot preference โดยจะมีค่าตอบแทนเป็นแบบถัวเฉลี่ย/ชั่วโมง และค่าตอบแทนของอาจารย์แต่ละท่านจะแตกต่างกันไปตาม ความชอบ/ไม่ชอบในวันและคาบเวลาที่ต้องการสอน รวมไปถึงประสบการณ์ของอาจารย์ ระดับ/หลักสูตรของนิสิตที่สอน เนื่องจากอาจารย์แต่ละท่านจะถูกมอบหมายรายวิชาที่จะต้องทำการสอนไว้ในข้อมูลนำเข้าแล้วก่อนการจัดตารางเรียนตารางสอน ดังนั้นต้นทุนในส่วนนี้จะไม่กระทบต่อคุณภาพการศึกษาในเชิงลบ

21. ในงานวิจัยนี้จะประยุกต์ใช้วิธีการเมต้าฮิวริสติกส์ที่ชื่อว่า วิธีการ Cuckoo Search ร่วมกับโปรแกรมจัดตารางเรียนตารางสอนแบบอัตโนมัติที่ถูกพัฒนาขึ้นมา

22. ในงานวิจัยนี้จะพัฒนาโปรแกรมจัดตารางเรียนตารางสอนแบบอัตโนมัติในลักษณะเป็นโปรแกรมแบบแยกส่วน (Stand alone) สำหรับเป็นทางเลือกให้กับผู้จัดตาราง มิได้พัฒนาให้โปรแกรมมีการทำงานที่เชื่อมต่อโดยตรงกับระบบเดิมของมหาวิทยาลัย เนื่องด้วยเหตุผลด้านความปลอดภัยและความเสถียรของระบบเดิมในมหาวิทยาลัย

23. ในงานวิจัยนี้จะพัฒนาโปรแกรมจัดตารางเรียนตารางสอนแบบอัตโนมัติโดยใช้ภาษา Tcl/Tk สำหรับพัฒนาในส่วนติดต่อกับผู้ใช้งาน (Graphical user interface: GUI) และใช้ภาษา C สำหรับพัฒนาส่วนของการประมวลผลทั้งหมด

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้ปรับปรุงกระบวนการจัดตารางเรียนตารางสอนจริงของคณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร ให้มีประสิทธิภาพมากยิ่งขึ้นด้วยระบบการจัดตารางเรียนตารางสอนแบบอัตโนมัติ ซึ่งมีความได้เปรียบกว่าระบบเดิมทั้งในด้านการจัดตารางเรียนตารางสอนได้ตรงตามความต้องการของผู้ใช้ตาราง ช่วยลดความผิดพลาดจากการจัดตารางเรียนตารางสอนชนกันหรือไม่เป็นไปตามข้อบังคับที่ได้กำหนดไว้ รวมถึงช่วยลดเวลาจำนวนมากที่จะต้องสูญเสียไปเมื่อเทียบกับการจัดตารางเรียนตารางสอนแบบเดิม

2. ได้โปรแกรมจัดตารางเรียนตารางสอนแบบอัตโนมัติสำหรับจัดตารางเรียนตารางสอนในระดับอุดมศึกษาโดยมีความฉลาดของวิธีการ CS ซึ่งสามารถจัดตารางเรียนตารางสอนให้เป็นไปตามข้อบังคับหลักและข้อบังคับรองของปัญหานี้ นอกจากโปรแกรมดังกล่าวสามารถนำโปรแกรมดังกล่าวไปต่อยอดหรือประยุกต์ใช้กับสถาบันการศึกษาอื่นได้

3. ช่วยลดค่าใช้จ่ายในด้านการบริหารจัดการทรัพยากรทางการศึกษาได้ เช่น ลดจำนวนการจ้างอาจารย์เพิ่มโดยไม่จำเป็น ลดการสร้างอาคารเรียนเพิ่มโดยไม่จำเป็น ลดค่าไฟจากการใช้ห้องเรียนที่ไม่เหมาะสม และยังรวมไปถึงช่วยลดค่าใช้จ่ายในการซื้อลิขสิทธิ์โปรแกรมหรือระบบการจัดตารางเรียนตารางสอนแบบอัตโนมัติอีกด้วย เป็นต้น

4. ได้ทราบถึงปัจจัยของวิธีการ CS ที่อาจจะมีผลกระทบต่อประสิทธิภาพในการแก้ปัญหาการจัดตารางเรียนตารางสอนในแต่ละโจทย์ปัญหา ทั้งผลกระทบหลัก (Main effects) และผลกระทบร่วม (Interactions) โดยใช้วิธีการออกแบบการทดลองและการวิเคราะห์ทางสถิติ รวมถึงได้ทราบค่าพารามิเตอร์ที่เหมาะสมของวิธีการ CS ในการแก้ปัญหาแต่ละโจทย์

5. ได้พัฒนาวิธีการ Cuckoo Search (CS) ให้มีประสิทธิภาพที่สูงขึ้นในการแก้ปัญหาการจัดตารางเรียนตารางสอน ด้วยการปรับค่าพารามิเตอร์ที่เหมาะสม (Parameter tuning) การปรับปรุงกระบวนการ (Modification process) และการผสมผสานกับวิธีการอื่น (Hybridisation) โดยที่วิธีการปรับปรุงเหล่านี้สามารถนำไปประยุกต์ใช้แก้ปัญหาเชิงการจัดเรียงสลับสับเปลี่ยน (Combinatorial optimisation problems) ที่ต้องการค่าที่เหมาะสมที่สุดประเภทอื่นได้

1.5 แผนการดำเนินงานของโครงการวิจัย

กิจกรรม	เดือน (Months)											
	2	4	6	8	10	12	14	16	18	20	22	24
1. ทบทวนวรรณกรรมที่เกี่ยวข้องกับปัญหาการจัดตารางสอนและวิธีการที่ใช้												
2. ศึกษาวิธีการ Cuckoo search (CS) และค้นหาแนวทางการปรับปรุงประสิทธิภาพ												
3. เก็บและวิเคราะห์ข้อมูลปัญหาการจัดตารางเรียนตารางสอนและกำหนดขอบเขตที่เกี่ยวข้อง												
4. สร้างรูปแบบของปัญหาและกำหนดข้อบังคับหลัก ข้อบังคับรองที่เกี่ยวข้อง												
5. ออกแบบโปรแกรมจัดตารางเรียนตารางสอนแบบอัตโนมัติ												
6. พัฒนาโปรแกรมจัดตารางเรียนตารางสอนแบบอัตโนมัติด้วยภาษา TCL/TK และ C												
7. ทดสอบ ตรวจสอบและยืนยันความถูกต้องของโปรแกรม ทำการออกแบบการทดลอง												
8. ดำเนินการทดลองตามที่ได้ออกแบบไว้บนคอมพิวเตอร์ที่จัดเตรียม												
9. วิเคราะห์ผลการทดลองด้วยเครื่องมือทางสถิติ (ANOVA, t-test, etc.)												
10. สรุปผลการทดลองทั้งหมดที่เป็นไปตามวัตถุประสงค์ที่ได้กำหนดไว้												
11. เขียนบทความวิจัยเพื่อส่งตีพิมพ์ลงในวารสารในฐานข้อมูล Scopus												
12. เขียนบทความวิจัยเพื่อส่งตีพิมพ์ลงในวารสารในฐานข้อมูล ISI												
13. เขียนรายงานฉบับสมบูรณ์												

บทที่ 2
เอกสารและงานวิจัยที่เกี่ยวข้อง

2.1 ทบทวนวรรณกรรมที่เกี่ยวข้องกับปัญหาการจัดตารางเรียนตารางสอน

การจัดตารางการศึกษาเป็นกิจกรรมที่เกิดขึ้นเป็นประจำทุกภาคการศึกษาในทุกสถาบันการศึกษา โดยมีความเกี่ยวข้องกับการจัดสรรทรัพยากรทางการศึกษาที่มีอยู่อย่างจำกัดภายใต้ข้อบังคับต่างๆ ลงในช่วงเวลาที่มีความเหมาะสมและเป็นไปตามวัตถุประสงค์มากที่สุด [1] การสร้างตารางการศึกษา (เช่น ตารางเรียน ตารางสอน ตารางการใช้ห้องเรียน เป็นต้น) ที่มีคุณภาพดีย่อมทำให้การดำเนินการเรียนการสอนในสถาบันการศึกษาเกิดประสิทธิภาพที่ดีเช่นกันในแต่ละเทอม ดังนั้นตลอดระยะเวลา 50 กว่าปีที่ผ่านมาจนถึงปัจจุบัน การแก้ปัญหาการจัดตารางการศึกษาจึงได้รับความสนใจจากนักวิจัยจำนวนมากทั้งในกลุ่มของการวิจัยการดำเนินงาน (Operations research: OR) และในกลุ่มของความฉลาดประดิษฐ์ (Artificial intelligent: AI) และยังคงได้รับความสนใจเพิ่มมากขึ้นอย่างต่อเนื่องในช่วงหลายสิบปีหลัง [2]

ปกติแล้วการจัดตารางเรียนตารางสอนในสถาบันระดับอุดมศึกษา (University course timetabling: UCT) เช่น วิทยาลัย (College) มหาวิทยาลัย (University) เป็นต้น จะมีขนาดของปัญหาที่ใหญ่มาก มีจำนวนข้อบังคับและข้อจำกัดมากกว่าการจัดตารางการศึกษาในสถาบันการศึกษาขั้นพื้นฐาน เช่น โรงเรียนมัธยมศึกษา (Secondary school) เป็นต้น ซึ่งความซับซ้อนของปัญหาและความยุ่งยากในการจัดตารางสอนในระดับอุดมศึกษาจะมีมากกว่า [3] นอกจากนี้วิชาเรียนวิชาสอนในระดับอุดมศึกษาจะมีทั้งแบบวิชาบังคับ (Mandatory courses) และวิชาเลือก (Elective courses) [4] โดยนิสิตในชั้นเดียวกันแต่ถ้ามีวิชาเลือกเรียนที่แตกต่างกันจะทำให้ตารางเรียนของนิสิตแต่ละคนแตกต่างกันออกไปตามวิชาที่เลือกเรียน ส่งผลให้การจัดตารางเรียนในชั้นดังกล่าวมีความยุ่งยากและซับซ้อนมากขึ้น

2.2 ปัญหาการจัดตารางเรียนตารางสอนในระดับอุดมศึกษาแบบซับซ้อน

ปัญหาการจัดตารางเรียนตารางสอนในระดับอุดมศึกษาแบบซับซ้อน (Complex university course timetabling: CUCT) จะมีลักษณะพิเศษของปัญหาเพิ่มเข้ามาในปัญหา UCT 3 ส่วนที่สำคัญ [56, 57] คือ: (i) ปัญหาย่อยตามลำดับชั้นในสถาบันอุดมศึกษา (Sub problems of university-wide level); (ii) โครงสร้างของวิชาเรียนวิชาสอน (Course structures); และ (iii) ความหลากหลายของข้อบังคับ (Variety of constraints)

ส่วนแรก ปัญหาย่อยตามลำดับชั้นในสถาบันอุดมศึกษา (Sub problems of university-wide level) เป็นลักษณะของการจัดตารางเรียนตารางสอนของสถาบันการศึกษาโดยมีการพิจารณาปัญหาการจัดตารางย่อยมาจากการแบ่งระดับหรือขั้นตอนหลายขั้นตอน เช่น ระดับสถาบัน (Central or university timetabling) ระดับคณะ (Faculty timetabling) ระดับภาควิชา (Departmental timetabling) [56] ในระดับสถาบันจะเป็นการจัดตารางเรียนตารางสอนโดยหน่วยงานส่วนกลางของสถาบันนั้นๆ ปกติแล้วจะ

ดำเนินการจัดตารางในรายวิชาพื้นฐาน (Mandatory courses) ทั้งหมดของสถาบันซึ่งเกี่ยวข้องกับนิสิตเป็นจำนวนมากจากทุกคณะที่จะต้องลงเรียน เช่น วิชาภาษาอังกฤษ วิชาภาษาไทย เป็นต้น การจัดตารางในระดับนี้จะถูกดำเนินการเป็นลำดับแรก ขณะที่ในระดับคณะและระดับภาควิชาจะดำเนินการได้หลังจากส่วนกลางได้ดำเนินการจัดตารางเสร็จสิ้นแล้ว โดยในแต่ละคณะจะให้แต่ละภาควิชาดำเนินการจัดตารางในรายวิชาที่รับผิดชอบสำหรับทุกหลักสูตรที่ภาควิชาได้ทำการเปิดสอน โดยส่วนมากจะเป็นวิชาเฉพาะหรือวิชาเลือก (Elective courses) ของนิสิตแต่ละหลักสูตรซึ่งเป็นกลุ่มที่มีนิสิตจำนวนน้อยลง

รายวิชา (Courses) ต่างๆ ของนิสิตหรือของอาจารย์แต่ละท่านในปัญหา CUCT จะมีความสัมพันธ์กันทั้งในส่วนของปัญหาการจัดตารางในระดับสถาบัน (Central timetabling) ในระดับคณะ (Faculties) และในระดับภาควิชา (Departments) เนื่องจากวิชาในแต่ละระดับจะมีจำนวนนิสิตที่ลงเรียนแตกต่างกัน นิสิตที่ลงเรียนอาจมาจากคณะหรือภาควิชาที่แตกต่างหรือเหมือนกันได้ รวมถึงมีการแชร์ทรัพยากรทางการศึกษาระหว่างกันในทุกระดับ (เช่น ห้องเรียน อาจารย์ผู้สอน เป็นต้น) นอกจากนี้ปัญหาในสถาบันอุดมศึกษาขนาดใหญ่จะประกอบไปด้วยปัญหาการจัดตารางสอนย่อย (Sub-timetabling problems) ในแต่ละคณะและในแต่ละภาควิชา ซึ่งการแก้ปัญหาการจัดตารางเรียนตารางสอนไม่ควรแก้เฉพาะเจาะจงเพียงปัญหาย่อยหนึ่งปัญหา [56] ดังนั้นสถาบันอุดมศึกษาที่มีคณะเป็นจำนวนมากและในแต่ละคณะมีภาควิชาจำนวนมากด้วยแล้ว การแก้ปัญหาในสถาบันดังกล่าวจะมีความยุ่งยากและซับซ้อนในการแก้ปัญหาเป็นอย่างมาก

ส่วนที่สอง โครงสร้างของวิชาเรียนวิชาสอน (Course structures) สำหรับปัญหา UCT ปกติแล้วจะกำหนดเพียงแค่วิชาหรือเหตุการณ์ (Courses or events) สำหรับจัดตารางเรียนตารางสอนเท่านั้น [13, 58, 59] แต่โครงสร้างของวิชาเรียนวิชาสอนของปัญหา CUCT นอกจากจะกำหนดวิชาหรือเหตุการณ์สำหรับจัดตารางเรียนตารางสอนแล้ว ยังจะต้องกำหนดรายละเอียดของแต่ละวิชาเพิ่มเข้ามาด้วย เช่น วิชาบังคับ/วิชาเลือก (Mandatory/Elective courses) [4] วิชาแบบภาคบรรยาย/แบบปฏิบัติการ/แบบทบทวน (Lecture/Laboratory/Tutorial courses) [60] วิชาที่มีหมู่เรียนเดี่ยวและแบบมีหลายหมู่เรียน (Single/Multiple sections) [57] ในแต่ละวิชา เป็นต้น

สำหรับการพิจารณาวิชาบังคับ (Mandatory courses) ซึ่งส่วนใหญ่เป็นวิชาพื้นฐานของมหาวิทยาลัยสำหรับนิสิตชั้นปีแรกๆ [4] จะทำให้การจัดตารางมีความยุ่งยากอันเนื่องมาจากจำนวนของนักเรียนที่ลงทะเบียน ขณะที่การพิจารณาวิชาเลือก (Elective courses) ซึ่งส่วนใหญ่เป็นวิชาเฉพาะภายในคณะหรือภาควิชาสำหรับนิสิตชั้นปีสูงๆ จะทำให้การจัดตารางสอนมีความซับซ้อนมากยิ่งขึ้น [4] เนื่องจากวิชาเลือกบางรายวิชานั้นนิสิตต่างภาควิชาสามารถลงเรียนได้ ซึ่งจะทำให้นิสิตในหลักสูตรเดียวกันสามารถมีตารางเรียนแตกต่างกันออกไปตามวิชาที่เลือกเรียนและจะทำให้เกิดการชนกันของตารางเรียนตารางสอนได้ง่ายมาก สำหรับการพิจารณาวิชาทั้งแบบภาคบรรยาย/ปฏิบัติการ นอกจากจะต้องคำนึงถึงขนาดของห้องเรียน (Room capacity) ในการจัดตารางสอนแล้ว ยังจะต้องพิจารณาถึงความต้องการห้องเรียน (Room facility) ที่เฉพาะในแต่ละวิชา เช่น บางรายวิชาอาจมีการใช้ทั้งห้องเรียนบรรยายและห้องเรียนปฏิบัติ ขณะที่บางวิชาต้องการเฉพาะห้องบรรยาย เป็นต้น สำหรับการพิจารณาวิชาแบบหลายหมู่เรียน (Multiple sections) จัดว่าเป็นการเพิ่มความซับซ้อนอีกรูปแบบหนึ่งในปัญหา CUCT [57] โดยในกรณีที่มีอาจารย์ผู้สอนคนเดียวจะทำให้การจัด

ตารางสอนโดยไม่ให้มีการชนกันของเวลาในแต่ละหมู่เรียนมีความยากยิ่งขึ้นและอาจารย์ก็จะมีภาระงานมากขึ้นด้วย

นอกเหนือจากโครงสร้างของวิชาเรียนวิชาสอนทั้ง 3 ลักษณะที่ได้กล่าวมาแล้ว งานวิจัยนี้ได้พิจารณาโครงสร้างของวิชาเรียนวิชาสอนเพิ่มอีกหนึ่งลักษณะคือ วิชาที่มีอาจารย์ผู้สอนแบบหลายคน/หมู่เรียน (Multiple lecturers: ML) [61] และวิชาที่มีอาจารย์ผู้สอนแบบหนึ่งคน/หมู่เรียน (Single lecturer: SL) [62] ซึ่งจากการทบทวนวรรณกรรมที่ผ่านมานอกจากจะพบว่า การจัดตารางเรียนตารางสอนในระดับอุดมศึกษาโดยพิจารณาอาจารย์ผู้สอนมากกว่า 1 คนต่อวิชาจะพบได้น้อยมากแล้ว ในสถาบันอุดมศึกษาของไทยจะพบรายวิชาลักษณะนี้ได้บ่อยมาก โดยเฉพาะอย่างยิ่งวิชาที่มีอาจารย์ผู้สอนแบบ ML และอาจารย์ทุกคนช่วยกันสอนพร้อมกันในแต่ละคาบ (Simultaneous teaching: ST) [61] เนื่องจากรายวิชาในลักษณะนี้ จำเป็นต้องใช้อาจารย์หลายท่านเข้ามาช่วยควบคุมดูแลการเรียนการสอนของนิสิตพร้อมๆ กัน เช่น วิชาที่มีภาคปฏิบัติการต่างๆ และจำเป็นต้องใช้ห้องปฏิบัติการซึ่งเกี่ยวข้องกับอุปกรณ์เฉพาะทาง เครื่องมือ เครื่องจักร หรือสารเคมีต่างๆ ที่อาจเกิดอันตรายต่อนิสิตได้ง่าย หากไม่มีการดูแลอย่างทั่วถึงจากอาจารย์หลายๆ คนอย่างใกล้ชิด เป็นต้น ความยากของการจัดตารางจากโครงสร้างของวิชาเรียนวิชาสอนแบบ ML คือ จะต้องทำการจัดตารางสอนในรายวิชาดังกล่าวให้กับอาจารย์ทุกคนที่รับผิดชอบรายวิชานั้น ลงในคาบเวลาเดียวกัน ห้องเรียนเดียวกัน และจะต้องไม่มีการชนกันในตารางสอนของอาจารย์ทุกคนในวิชาดังกล่าวด้วย ซึ่งรายวิชาใดที่มีจำนวนอาจารย์ผู้สอนยิ่งมากจะทำให้การจัดตารางสอนในรายวิชาดังกล่าวมีความยากมากยิ่งขึ้นตามลำดับ

ดังนั้นจะเห็นได้ว่าปัญหาแบบ CUCT มีโครงสร้างของวิชาเรียนวิชาสอนแบบซับซ้อนจะทำให้การจัดตารางเรียนตารางสอนมีความยุ่งยากมากกว่าปัญหาแบบ UCT อย่างชัดเจน อย่างไรก็ตามหลายงานวิจัยที่ผ่านมาได้มุ่งไปที่การแก้ปัญหา UCT โดยใช้ข้อมูลประดิษฐ์ (Artificial dataset) หรือข้อมูลที่อยู่บนพื้นฐานปัญหาจริง (Based on actual problem) เป็นจำนวนมากซึ่งความซับซ้อนในโครงสร้างของวิชาประเภทต่างๆ ที่ได้กล่าวมาข้างต้นนั้นได้ถูกตัดออกไปเหลือเพียงแค่วิชาหรือเหตุการณ์ (Courses or events) เท่านั้น เพื่อให้โจทย์ปัญหามีลักษณะเป็นมาตรฐาน (Benchmark problems) แต่ปัญหาดังกล่าวมีความง่ายขึ้นอย่างมาก ดังนั้นวิธีการที่จะถูกพัฒนาขึ้นมาเพื่อใช้แก้ปัญหาจริงในระดับอุดมศึกษาซึ่งมีข้อมูลขนาดใหญ่และซับซ้อน (CUCT) จึงยังพบได้น้อยมาก [56]

ส่วนที่สาม ความหลากหลายของข้อบังคับ (Variety of constraints) โดยทั่วไปแล้วข้อบังคับ (Constraints) ของปัญหาการจัดตารางเรียนตารางสอนมี 2 ประเภทใหญ่คือ ข้อบังคับหลัก (Hard constraints: HC) และข้อบังคับรอง (Soft constraints: SC) [5-8] โดยที่เป้าประสงค์หลักของการแก้ปัญหาการจัดตารางเรียนตารางสอนคือ การค้นหาตารางเรียนตารางสอนที่ไม่มีการละเมิดข้อบังคับหลักและมีจำนวนการละเมิดข้อบังคับรองที่น้อยที่สุด [6, 9, 10]

2.3 ฟังก์ชันเป้าประสงค์ของการแก้ปัญหาการจัดตารางเรียนตารางสอน

ในฟังก์ชันเป้าประสงค์ของการแก้ปัญหาการจัดตารางเรียนตารางสอนสามารถจำแนกข้อบังคับรอง (SC) ที่พบได้หลายประเภท ประกอบด้วย: (i) แบบ Unary constraints เช่น การนับจำนวนครั้งในการจัดวิชาลง

ในคาบเวลาห้ามจัดตารางสอน (Unpermitted or booking periods) [11] เป็นต้น; (ii) แบบ Binary constraints เช่น การนับจำนวนครั้งการชนกันของตารางสอน (Event clash constraints) [12] เป็นต้น; (iii) แบบ Capacity constraints เช่น การนับจำนวนเก้าอี้ที่ไม่เพียงพอทั้งหมดในการจัดตารางสอน [13] เป็นต้น; (iv) แบบ Event spread constraints เช่น การนับจำนวนครั้งที่มีการเรียนการสอนต่อเนื่องเกินสองคาบติดกัน (Spreading-out or clumping-together) [14] เป็นต้น; (v) แบบ Agent constraints เช่น การนับจำนวนครั้งที่มีรูปแบบการสอน (Lecturing format) ไม่เป็นไปตามที่อาจารย์ต้องการ (People preferences or requirements) [1] เป็นต้น; และ (vi) แบบ Stability or movement constraints เช่น การนับจำนวนครั้งที่วิชาเดียวกันแต่จัดตารางให้ใช้ห้องเรียนต่างกันทุกครั้ง [15] เป็นต้น

อย่างไรก็ตามจากการทบทวนวรรณกรรมจะพบว่า ยังมีข้อบังคับรองอีกประเภทคือ แบบ Cost constraints ซึ่งงานวิจัยที่พิจารณาฟังก์ชันเป้าประสงค์จากข้อบังคับรองที่เกี่ยวข้องกับการจัดตารางเรียนตารางสอนโดยเกิดค่าดำเนินการรวมจากการใช้ทรัพยากรทางการศึกษาที่น้อยที่สุด (Minimising total operating costs) นั้นพบได้น้อยมากเพียง 2 บทความจากฐานข้อมูล ISI web of knowledge ดังนี้ Seo และคณะ [16] ได้ทำการปรับปรุงระบบการจัดตารางสอนในองค์กรให้มีความรวดเร็วมากขึ้นโดยการพัฒนาโปรแกรม eClasSkeduler สำหรับใช้ในมหาวิทยาลัย Universidad de Chile โดยพิจารณาต้นทุนการเช่าห้องเรียน (Classrooms leasing/operating costs) ที่น้อยที่สุดในฟังก์ชันเป้าประสงค์ นอกจากนี้ Torres-Ovalle และคณะ [17] ได้แก้ปัญหาการจัดตารางสอนจริงของมหาวิทยาลัย Universidad de La Sabana ที่โคลัมเบียด้วยวิธีการเชิงเส้นแบบจำนวนเต็ม (Integer linear programming) โดยพิจารณาต้นทุนการจ้างอาจารย์ (Lecturing costs) ที่น้อยที่สุดในฟังก์ชันเป้าประสงค์

ดังนั้นในงานวิจัยนี้จึงได้เติมเต็มการพิจารณาฟังก์ชันเป้าประสงค์ในข้อบังคับรองแบบ Cost constraints โดยจัดตารางเรียนตารางสอนเพื่อให้เกิดค่าดำเนินการรวมจากการใช้ทรัพยากรทางการศึกษาที่น้อยที่สุด (Minimising total operating costs) ประกอบด้วย ต้นทุนค่าโสหุ้ย (Overhead costs) เช่น ค่าเช่าห้องเรียน ค่าน้ำ ค่าไฟ เป็นต้น ต้นทุนการจ้างอาจารย์ (Lecturing costs) และต้นทุนการจัดเตรียมหรือทำความสะอาดห้อง (Setup/cleaning costs) ซึ่งยังไม่พบงานวิจัยใดพิจารณาฟังก์ชันเป้าประสงค์ย่อย 3 แบบนี้ร่วมกันมาก่อน (แสดงดังตาราง 1) นอกจากนี้ในงานวิจัยนี้ ต้นทุนจากการเช่าหรือใช้ห้องเรียนและต้นทุนการจัดเตรียมหรือทำความสะอาดห้อง จะพิจารณากำหนดความชอบทั้งสถาบัน (Campus preference: CP) โดยที่ต้นทุนทั้งสองประเภทจะแตกต่างกันไปตาม ขนาดของห้องเรียน ประเภทของห้องเรียน วันและช่วงเวลาการใช้หรือทำความสะอาดห้องเรียน ในขณะที่ต้นทุนในการจ้างอาจารย์ผู้สอน จะมีการพิจารณาความชอบแบบยืดหยุ่น (Flexible preference: FP) โดยที่ค่าตอบแทนของอาจารย์แต่ละท่านจะแตกต่างกันไปตาม วุฒิการศึกษา ประสบการณ์การทำงาน ระดับของนักศึกษาที่สอน วันและช่วงเวลาที่สอน ยิ่งไปกว่านั้นจากการทบทวนวรรณกรรมที่ผ่านมาพบว่า งานวิจัยที่เกี่ยวข้องกับปัญหาการจัดตารางเรียนตารางสอนระดับอุดมศึกษาที่พิจารณาการกำหนดข้อบังคับรองแบบ FP นั้นยังมีจำนวนน้อย

ตาราง 1 แสดงคุณลักษณะของปัญหาการจัดตารางเรียนตารางสอนที่ได้จากการทบทวนวรรณกรรม

Authors and years	Timetabling types				System		Period			Course types					Course section		Lecturer/ Section		Teaching types		Operating costs		
	CT	CTT	SS	TA	CA	MTS	DDS	Weekly	Non- Weekly	Lecture	Laboratory	Tutorial	Seminar	Event / Course	Single	Multiple	One	Many	Individually	Simultaneously	Overhead	Lecturing	Setup/cleaning
Abbas and Tsang [63]	✓				✓		✓	✓						✓	✓	✓	✓		✓				
Abdullah et al. [64]	✓				✓		✓	✓						✓	✓		✓		✓				
Abdullah and Turabieh [59]	✓				✓	✓		✓						✓	✓		✓		✓				
Abdullah et al. [65]	✓				✓		✓	✓						✓	✓		✓		✓				
Abdullah et al. [66]	✓				✓	✓	✓	✓						✓	✓		✓		✓				
Abuhamdah and Ayob [67]	✓				✓		✓	✓						✓	✓		✓		✓				
Abuhamdah et al. [68]	✓				✓		✓	✓						✓	✓		✓		✓				
Acha and Nieuwenhuis [69]	✓				✓	✓		✓						✓	✓		✓		✓				
Agustin-Blas et al. [70]			✓				✓	✓		✓	✓					✓							
Al-Betar and Khader [71]	✓				✓		✓	✓						✓	✓		✓		✓				
Al-Betar et al. [71]	✓				✓		✓	✓						✓	✓		✓		✓				
Al-Yakoob and Sherali [72]				✓			✓	✓		✓	✓					✓	✓		✓				
Al-Yakoob and Sherali [73]	✓			✓	✓	✓	✓	✓		✓	✓	✓	✓			✓	✓		✓				
Aladag and Hocaoglu [74]	✓				✓	✓		✓		✓	✓					✓	✓						
Aladag et al. [75]	✓				✓	✓		✓		✓	✓					✓	✓						
Alvarez-Valdes et al. [76]	✓		✓		✓	✓	✓	✓		✓	✓				✓	✓	✓		✓				
Amintoosi and Haddadnia [77]			✓				✓							✓		✓							
Avella and Vasil'Ev [78]	✓				✓	✓		✓						✓		✓							

Authors and years	Timetabling types					System		Period		Course types					Course section		Lecturer/ Section		Teaching types		Operating costs		
	CT	CTT	SS	TA	CA	MTS	DDS	Weekly	Non- Weekly	Lecture	Laboratory	Tutorial	Seminar	Event / Course	Single	Multiple	One	Many	Individually	Simultaneously	Overhead	Lecturing	Setup/cleaning
Badoni et al. [79]	✓				✓		✓	✓						✓	✓		✓		✓				
Bai et al. [80]	✓				✓		✓	✓						✓	✓		✓		✓				
Baker et al. [81]			✓				✓		✓					✓		✓							
Bakir and Aksop [82]	✓				✓		✓	✓		✓	✓					✓							
Banbara et al. [83]	✓				✓	✓		✓						✓	✓		✓		✓				
Bellio et al. [84]	✓				✓	✓		✓						✓	✓		✓		✓				
Bellio et al. [85]	✓				✓	✓		✓						✓	✓		✓		✓				
Beyrouthy et al. [86]	✓				✓	✓		✓		✓			✓		✓	✓							
Beyrouthy et al. [60]	✓				✓	✓		✓		✓	✓	✓	✓		✓	✓							
Bolaji et al. [10]	✓				✓	✓		✓						✓	✓		✓		✓				
Bolaji et al. [87]	✓				✓		✓	✓						✓	✓		✓		✓				
Bonutti et al. [88]	✓				✓	✓		✓						✓	✓		✓		✓				
Burke et al. [89]	✓				✓		✓	✓						✓	✓		✓		✓				
Burke et al. [7]	✓				✓		✓	✓						✓	✓		✓		✓				
Burke et al. [90]	✓				✓	✓		✓						✓	✓		✓		✓				
Burke et al. [91]	✓				✓	✓		✓						✓	✓		✓		✓				
Burke et al. [92]	✓				✓	✓		✓						✓	✓		✓		✓				
Cacchiani et al. [93]	✓				✓	✓		✓						✓	✓		✓		✓				
Cambazard et al. [6]	✓				✓		✓	✓						✓	✓		✓		✓				
Cambazard et al. [94]	✓				✓	✓		✓		✓	✓					✓							

Authors and years	Timetabling types					System		Period		Course types					Course section		Lecturer/ Section		Teaching types		Operating costs		
	CT	CTT	SS	TA	CA	MTS	DDS	Weekly	Non- Weekly	Lecture	Laboratory	Tutorial	Seminar	Event / Course	Single	Multiple	One	Many	Individually	Simultaneously	Overhead	Lecturing	Setup/cleaning
Carrasco and Pato [95]		✓				✓		✓		✓	✓					✓	✓		✓				
Chiarandini et al. [96]	✓				✓		✓	✓						✓	✓		✓		✓				
Chiarandini et al. [97]	✓				✓	✓			✓					✓	✓								
Ceschia et al. [98]	✓				✓		✓	✓						✓	✓		✓		✓				
Chen and Shih [48]	✓				✓	✓		✓				✓		✓									
Dammak et al. [99]	✓				✓	✓		✓		✓		✓				✓	✓		✓				
Daskalaki et al. [4]	✓				✓	✓		✓		✓	✓	✓			✓	✓	✓		✓				
Daskalaki and Birbas [100]	✓				✓	✓		✓		✓	✓	✓			✓	✓	✓		✓				
Datta et al. [3]	✓				✓	✓		✓		✓	✓	✓			✓	✓	✓		✓				
De Causmaecker et al. [61]	✓				✓	✓			✓	✓	✓				✓	✓		✓		✓			
Di Gaspero and Schaerf [101]	✓				✓	✓		✓						✓	✓		✓		✓				
Dimopoulou and Miliotis [102]	✓				✓	✓		✓						✓	✓	✓							
Fong et al. [103]	✓				✓		✓	✓						✓	✓		✓		✓				
Fong et al. [104]	✓				✓		✓	✓						✓	✓		✓		✓				
Gaspero et al. [62]	✓				✓	✓		✓						✓	✓		✓		✓				
Geiger [105]	✓				✓	✓		✓						✓	✓		✓		✓				
Gunawan et al. [106]	✓			✓	✓	✓		✓						✓		✓	✓		✓				
Hao and Benlic [15]	✓				✓	✓		✓						✓	✓		✓		✓				
He et al. [107]	✓				✓		✓	✓		✓	✓	✓			✓	✓							
Head and Shaban [108]	✓		✓		✓		✓	✓		✓	✓				✓	✓	✓		✓				

Authors and years	Timetabling types				System		Period			Course types					Course section		Lecturer/ Section		Teaching types		Operating costs		
	CT	CTT	SS	TA	CA	MTS	DDS	Weekly	Non- Weekly	Lecture	Laboratory	Tutorial	Seminar	Event / Course	Single	Multiple	One	Many	Individually	Simultaneously	Overhead	Lecturing	Setup/cleaning
Jaradat et al. [109]	✓				✓		✓	✓						✓	✓		✓		✓				
Jat and Yang [54]	✓				✓		✓	✓						✓	✓		✓		✓				
Jat and Yang [110]	✓				✓		✓	✓						✓	✓		✓		✓				
Jat and Yang [12]	✓				✓		✓	✓						✓	✓		✓		✓				
Junaedi and Maulidevi [111]	✓				✓	✓		✓						✓	✓		✓		✓				
Kalender et al. [112]	✓				✓	✓		✓		✓	✓	✓											
Kardan et al. [113]			✓				✓	✓						✓	✓	✓							
Khang and Nuong [114]	✓				✓	✓		✓		✓					✓		✓		✓				
Kostuch [115]	✓				✓		✓	✓						✓	✓		✓		✓				
Kostuch and Socha [116]	✓				✓		✓	✓						✓	✓		✓		✓				
Lach and Lubbecke [117]	✓				✓	✓		✓						✓	✓		✓		✓				
Lee et al. [118]	✓				✓	✓		✓						✓									
Legierski [119]		✓			✓	✓		✓	✓	✓	✓				✓		✓	✓		✓			
Lewis [11]	✓				✓		✓	✓						✓	✓		✓		✓				
Lewis and Paechter [120]	✓				✓		✓	✓						✓	✓		✓		✓				
Lewis and Paechter [121]	✓				✓		✓	✓						✓	✓		✓		✓				
Lewis and Thompson [122]	✓				✓		✓	✓						✓	✓		✓		✓				
Lewis et al. [123]	✓				✓		✓	✓						✓	✓		✓		✓				
Liu et al. [124]	✓				✓		✓	✓						✓	✓		✓		✓				
Lü and Hao [9]	✓				✓	✓		✓						✓	✓		✓		✓				

Authors and years	Timetabling types				System			Period		Course types					Course section		Lecturer/ Section		Teaching types		Operating costs		
	CT	CTT	SS	TA	CA	MTS	DDS	Weekly	Non- Weekly	Lecture	Laboratory	Tutorial	Seminar	Event / Course	Single	Multiple	One	Many	Individually	Simultaneously	Overhead	Lecturing	Setup/cleaning
Lu et al. [125]	✓				✓	✓		✓						✓	✓		✓		✓				
Malim et al. [126]	✓				✓	✓		✓						✓	✓		✓		✓				
Marquez et al. [127]	✓				✓	✓		✓						✓	✓		✓		✓				
McCollum et al. [128]	✓				✓		✓	✓						✓	✓		✓		✓				
Miranda et al. [129]	✓				✓		✓	✓		✓	✓	✓	✓		✓	✓							
MirHassani [130]	✓				✓	✓	✓	✓		✓	✓						✓		✓				
Mueller and Rudova [131]	✓				✓	✓	✓	✓						✓	✓	✓							
Muller [58]	✓				✓	✓	✓	✓						✓	✓		✓		✓				
Muller et al. [132]	✓				✓		✓	✓		✓	✓				✓	✓	✓		✓				
Murray et al. [56]	✓				✓		✓	✓		✓	✓				✓	✓	✓		✓				
Nothegger et al. [133]	✓				✓		✓	✓						✓	✓		✓		✓				
Ozer and Ozturan [134]	✓		✓		✓	✓	✓	✓						✓	✓	✓							
Pereira and Costa [135]	✓				✓	✓		✓		✓	✓												
Perzina [49]	✓				✓		✓	✓		✓			✓		✓								
Phillips et al. [136]	✓				✓	✓		✓						✓	✓		✓		✓				
Piechowiak and Kolski [137]	✓				✓	✓			✓	✓	✓	✓			✓	✓	✓						
Piechowiak et al. [138]	✓				✓	✓			✓	✓	✓	✓			✓	✓	✓						
Pongcharoen et al. [1]	✓				✓	✓		✓						✓	✓	✓	✓	✓					
Qaurooni and Akbarzadeh-T [139]	✓				✓		✓	✓						✓	✓		✓		✓				
Qu and Burke [140]	✓				✓		✓	✓						✓	✓		✓		✓				

Authors and years	Timetabling types				System		Period			Course types					Course section		Lecturer/ Section		Teaching types		Operating costs		
	CT	CTT	SS	TA	CA	MTS	DDS	Weekly	Non- Weekly	Lecture	Laboratory	Tutorial	Seminar	Event / Course	Single	Multiple	One	Many	Individually	Simultaneously	Overhead	Lecturing	Setup/cleaning
Qualizza and Serafini [141]	✓				✓	✓		✓						✓	✓								
Rossi-Doria et al. [142]	✓				✓		✓	✓						✓	✓		✓		✓				
Rossi-Doria et al. [19]	✓				✓		✓	✓						✓	✓		✓		✓				
Rudova and Murray [143]	✓				✓		✓	✓		✓	✓				✓	✓	✓		✓				
Rudova et al. [57]	✓				✓		✓	✓		✓	✓				✓	✓	✓		✓				
Sabar et al. [47]	✓				✓		✓	✓						✓	✓		✓		✓				
Salman and Hamdan [144]	✓				✓	✓		✓		✓	✓				✓	✓	✓		✓				
Santiago-Mozos et al. [145]	✓		✓				✓	✓		✓	✓					✓							
Santos et al. [146]		✓				✓		✓						✓		✓	✓		✓				
Santos et al. [147]		✓				✓		✓						✓		✓	✓		✓				
Sarin et al. [148]	✓				✓	✓		✓		✓	✓						✓		✓				
Schimmelpfeng and Helber [149]	✓				✓	✓		✓		✓		✓	✓		✓	✓	✓	✓		✓			
Seo et al. [16]	✓				✓		✓		✓	✓	✓										✓		
Shiau [55]	✓				✓	✓		✓		✓	✓				✓	✓	✓		✓				
Shih et al. [150]	✓				✓		✓	✓		✓	✓					✓	✓		✓				
Shimazaki et al. [151]	✓				✓		✓	✓						✓									
Socha [152]	✓				✓		✓	✓						✓	✓		✓		✓				
Socha et al. [20]	✓				✓		✓	✓						✓	✓		✓		✓				
Soria-Alcaraz et al. [153]	✓				✓	✓		✓						✓	✓		✓		✓				
Soria-Alcaraz et al. [154]	✓				✓	✓	✓	✓						✓	✓		✓		✓				

Authors and years	Timetabling types				System		Period			Course types					Course section		Lecturer/ Section		Teaching types		Operating costs		
	CT	CTT	SS	TA	CA	MTS	DDS	Weekly	Non- Weekly	Lecture	Laboratory	Tutorial	Seminar	Event / Course	Single	Multiple	One	Many	Individually	Simultaneously	Overhead	Lecturing	Setup/cleaning
Soza et al. [155]	✓				✓		✓	✓						✓	✓		✓		✓				
Studenovsky [156]	✓				✓		✓	✓						✓	✓		✓		✓				
Tam et al. [157]	✓				✓		✓	✓						✓	✓		✓		✓				
Thepphakorn and Pongcharoen [46]	✓				✓	✓		✓						✓	✓		✓		✓				
Thepphakorn et al. [13]	✓				✓	✓		✓						✓	✓		✓		✓				
Thepphakorn et al. [158]	✓				✓	✓		✓						✓	✓		✓		✓				
Thompson [159]	✓		✓		✓		✓	✓						✓	✓	✓							
Torres-Ovalle et al. [17]	✓				✓		✓	✓						✓		✓						✓	
van den Broek et al. [160]	✓		✓				✓	✓						✓		✓							
van den Broek and Hurkens [161]	✓				✓		✓	✓						✓	✓		✓		✓				
Wang [162]	✓				✓	✓		✓		✓	✓						✓		✓				
Wehrer and Yellen [163]	✓				✓	✓		✓		✓	✓				✓	✓	✓		✓				
Wu [164]	✓				✓	✓		✓						✓									
Wu [165]	✓				✓	✓		✓						✓									
Yang and Jat [166]	✓				✓		✓	✓						✓	✓		✓		✓				
Research objectives	✓				✓	✓		✓	✓	✓	✓				✓	✓	✓	✓	✓	✓	✓	✓	✓

2.4 ทบทวนวรรณกรรมที่เกี่ยวข้องกับวิธีการหาค่าที่เหมาะสมที่สุดในการแก้ปัญหา

โดยทั่วไปแล้วการจัดตารางเรียนตารางสอนสามารถทำได้โดยใช้มนุษย์ (Manually by academic staff) หรือโดยใช้โปรแกรมแบบกึ่งอัตโนมัติ (Semi-automatically) หรือโดยใช้โปรแกรมแบบอัตโนมัติ (Automatically) ปัจจุบันระบบการจัดตารางเรียนตารางสอนในสถาบันการศึกษาระดับสูงของประเทศไทยส่วนใหญ่จะยังเป็นการจัดตารางเรียนตารางสอนแบบกึ่งอัตโนมัติ ซึ่งเป็นการใช้ซอฟต์แวร์คอมพิวเตอร์มาช่วยเหลือมนุษย์เฉพาะการตรวจสอบการชนกันของตารางเรียนตารางสอนขณะทำการจัดตารางเท่านั้น ในขณะที่การเลือกห้องเรียนและคาบเวลาให้กับแต่ละวิชานั้นยังคงให้มนุษย์เป็นผู้ตัดสินใจทั้งหมด ซึ่งต้องใช้บุคลากรจำนวนมากและเวลาที่นานมากในการเลือกคาบเวลาที่ไม่ชนกันให้กับทุกวิชาของมหาวิทยาลัยในแต่ละเทอม ในขณะที่การจัดตารางเรียนตารางสอนแบบอัตโนมัติจะเป็นการพัฒนาซอฟต์แวร์คอมพิวเตอร์ขึ้นมาช่วยจัดตารางเรียนตารางสอนแทนมนุษย์ทั้งในส่วนของการตัดสินใจเลือกห้องเรียน/เวลาที่เหมาะสมให้กับแต่ละวิชาและการตรวจสอบข้อบังคับหลักต่างๆ ไปพร้อมๆ กัน ซึ่งมีข้อได้เปรียบกว่าวิธีการจัดตารางเรียนตารางสอน 2 แบบแรก คือ เหมาะกับการแก้ปัญหาขนาดใหญ่มากและมีข้อบังคับจำนวนมาก มีความสะดวกและสามารถการจัดตารางเรียนตารางสอนได้รวดเร็วกว่ามาก ใช้บุคลากรในการจัดตารางน้อยกว่ามาก ลดจำนวนขั้นตอนในการจัดตารางเรียนตารางสอนให้เหลือเพียงขั้นตอนเดียว สามารถจัดตารางเรียนตารางสอนตามที่นักเรียนและอาจารย์ต้องการได้

แม้ว่าโปรแกรมจัดตารางเรียนตารางสอนแบบอัตโนมัติของต่างประเทศมีอยู่หลายโปรแกรม แต่โปรแกรมดังกล่าวมีค่าใช้จ่ายสูง ไม่สะดวกในด้านการอบรมและการบำรุงรักษา อีกทั้งยังมีความแตกต่างทางด้านวัฒนธรรมและประเพณีของการศึกษาในแต่ละประเทศซึ่งมีความเฉพาะ จึงทำให้การประยุกต์ใช้โปรแกรมแบบอัตโนมัติของต่างประเทศกับสถานศึกษาในประเทศไทยอาจจะทำได้ค่อนข้างยาก ดังนั้นการการพัฒนาโปรแกรมในการจัดตารางเรียนตารางสอนระดับอุดมศึกษาแบบอัตโนมัติจึงเป็นอีกทางเลือกในงานวิจัยนี้

นอกจากนี้การปรับปรุงและพัฒนาโปรแกรมในการจัดตารางสอนให้มีประสิทธิภาพสูงขึ้นเป็นอีกประเด็นที่สำคัญ เนื่องจากปัญหาการจัดตารางเรียนตารางสอนในระดับอุดมศึกษามีลักษณะเป็นแบบ Combinatorial Optimisation (CO) [4, 18] และยังถูกจัดเป็นปัญหาแบบ Non-deterministic Polynomial (NP) hard problem [19, 20] หมายความว่า เมื่อขนาดของปัญหาเพิ่มขึ้นเพียงเล็กน้อยแต่เวลาที่ต้องการในการแก้ปัญหาจะเพิ่มขึ้นอย่างทวีคูณ [1] ดังนั้นหลายงานวิจัยในปัจจุบันจึงให้ความสนใจในเรื่องของการนำวิธีการแก้ปัญหาขนาดใหญ่และซับซ้อนที่มีประสิทธิภาพสูงมาประยุกต์ใช้ร่วมกับโปรแกรมสำเร็จรูปมากขึ้น [21]

2.5 วิธีการเมต้าฮิวริสติกส์

วิธีการในกลุ่มของเมต้าฮิวริสติกส์ (Metaheuristics) ถูกยอมรับอย่างแพร่หลายแล้วว่าเป็นอีกแนวทางสำหรับแก้ปัญหาแบบ NP hard problems โดยอาศัยหลักการประมาณค่าในการค้นหาคำตอบที่มีคุณภาพดีภายในเวลาที่ยอมรับได้ [5] สามารถแก้ปัญหาที่มีขนาดใหญ่และมีความซับซ้อนสูงได้อย่างมีประสิทธิภาพและประสิทธิผล [22] วิธีการในกลุ่มนี้นิยมจำแนกได้เป็น 2 กลุ่ม คือ Single-solution based metaheuristics

(S-meta) และ Population based metaheuristics (P-meta) [18, 22] วิธีการในกลุ่ม P-meta จะได้เปรียบกว่าวิธีการในกลุ่ม S-meta ในด้านการเริ่มต้นจากคำตอบหลายคำตอบ (Population) ไว้สำหรับพัฒนาคำตอบใหม่ๆ โดยคำตอบที่ได้จะมีลักษณะเด่นที่เป็นกลุ่มของคำตอบที่มีความหลากหลาย (Diversification) จึงทำให้วิธีการนี้มีลักษณะเด่นในการค้นหาคำตอบเชิงสำรวจ (Exploration search) เป็นหลัก [22] วิธีการในกลุ่มของ P-meta ชนิดใหม่ๆ ถูกนำเสนอขึ้นมาใหม่อย่างต่อเนื่องในปัจจุบัน เช่น วิธีการ Cuckoo Search [23], วิธีการ Bat Algorithm [24], วิธีการ Firefly Algorithm [25], วิธีการ Krill Herd [26], วิธีการ Flower Pollination Algorithm [27], วิธีการ Invasive Weed Optimisation [28], วิธีการ Gravitational search [29], วิธีการ Intelligent water drop [30], วิธีการ Backtracking optimisation search [31], วิธีการ League championship algorithm [32] เป็นต้น

วิธีการคุกคูเสิร์ช (Cuckoo Search: CS) เป็นหนึ่งในวิธีการเมต้าฮิวริสติกส์แบบ P-meta แบบใหม่ที่ได้แรงดลใจมาจากธรรมชาติ (Nature-inspired algorithms) ถูกพัฒนาขึ้นโดย Yang และ Deb ในปี 2009 [23] วิธีการ CS นี้ได้แนวคิดมาจากพฤติกรรมอันชาญฉลาดในการฝากฟักไข่ของนกกาเหว่าไว้กับรังของนกสายพันธ์อื่นเพื่อความอยู่รอดของเผ่าพันธุ์ ถึงแม้จะเป็นวิธีการที่ค่อนข้างใหม่ใหม่ แต่ในช่วง 4-5 ปีที่ผ่านมาวิธีการ CS กลับได้รับความนิยมนำมาประยุกต์ใช้แก้ปัญหาการหาค่าที่เหมาะสมที่สุดอย่างแพร่หลายและประสบผลสำเร็จเป็นอย่างดี เช่น ปัญหาการหาค่าที่เหมาะสมที่สุดเชิงตัวเลข (Numerical optimisation problems) [33], ปัญหาการหาค่าที่เหมาะสมที่สุดทางวิศวกรรม (Engineering optimisation problems) [23], ปัญหาการเดินทางของพนักงานขายแบบพื้นผิวทรงกลม (Spherical traveling salesman problems) [34], ปัญหาการจัดตารางการเข้าเวรของพยาบาล (Nurse scheduling problems) [35], ปัญหาการจัดตารางการผลิต (Production scheduling problems) [36] เป็นต้น

นอกจากนี้หลายงานวิจัยยังได้ทำการเปรียบเทียบประสิทธิภาพในการแก้ปัญหาของวิธีการ CS กับวิธีการเมต้าฮิวริสติกส์แบบอื่นอีกด้วย เช่น ปัญหาการหาค่าที่เหมาะสมที่สุดเชิงตัวเลขพบว่า วิธีการ CS มีประสิทธิภาพในการหาคำตอบที่ดีกว่า GA, PSO [37] และ ABC [33] สำหรับปัญหากระบวนการบด (Milling operation) พบว่า วิธีการ CS มีประสิทธิภาพในการหาคำตอบที่ดีกว่าทั้งวิธีการ GA, ACO, AIS และ PSO [38] เป็นต้น ดังนั้นจะเห็นได้ว่าวิธีการ CS เป็นวิธีการหนึ่งที่น่าสนใจมาก เพราะว่านอกจากจะมีจำนวนพารามิเตอร์ที่ต้องปรับค่าให้เหมาะสมน้อยกว่าวิธีการเมต้าฮิวริสติกส์แบบอื่นแล้ว เช่น GA และ PSO [37] เป็นต้น วิธีการ CS ยังมีประสิทธิภาพในการหาคำตอบที่ดีมากเมื่อเปรียบเทียบกับวิธีการเมต้าฮิวริสติกส์แบบอื่นๆ จากการประยุกต์ใช้แก้ปัญหาหลายประเภท อย่างไรก็ตามการประยุกต์ใช้วิธีการ CS แก้ปัญหาการจัดตารางเรียนตารางสอนระดับมหาวิทยาลัย เมื่อได้ทำการสืบค้นในฐานข้อมูลระดับนานาชาติ ISI web of knowledge, Scopus, และ IEEE Xplore ทั้งในส่วนที่เป็นชื่อเรื่อง (Title) บทคัดย่อ (Abstract) และคำสำคัญ (Keyword) จะพบเพียง 1 บทความเฉพาะในฐานข้อมูลของ Scopus เท่านั้น ในขณะที่ฐานข้อมูล ISI web of knowledge และ IEEE Xplore ยังไม่พบงานวิจัยที่ประยุกต์ใช้วิธีการ CS แก้ปัญหาการจัดตารางเรียนตารางสอนระดับมหาวิทยาลัยมาก่อน

ถึงแม้ว่าวิธีการเมต้าฮิวริสติกส์จะมีข้อดีต่างๆ มากมายและได้รับความนิยมอย่างมากในปัจจุบัน แต่วิธีการดังกล่าวก็มีปัญหาบางประการซึ่งส่งผลโดยตรงต่อประสิทธิภาพในการค้นหาคำตอบเช่นกัน เนื่องจากประสิทธิภาพในการหาคำตอบของวิธีการเมต้าฮิวริสติกส์นั้นขึ้นอยู่กับค่าพารามิเตอร์ที่ใช้ [22, 39] การค้นหาและการกำหนดค่าพารามิเตอร์ที่เหมาะสมให้กับวิธีการเมต้าฮิวริสติกส์จึงเป็นสิ่งที่ไม่อาจหลีกเลี่ยงไปได้ ดังนั้นหลายงานวิจัยจึงได้ศึกษาและค้นหาค่าพารามิเตอร์ที่เหมาะสมที่สุดให้กับวิธีการเมต้าฮิวริสติกส์เพื่อให้วิธีการดังกล่าวมีประสิทธิภาพสูงขึ้นกว่าเดิมในการแก้ปัญหา [40-42]

2.6 การกำหนดพารามิเตอร์ที่เหมาะสมให้กับวิธีการเมต้าฮิวริสติกส์

สำหรับแนวทางในการกำหนดพารามิเตอร์ที่เหมาะสมให้กับวิธีการเมต้าฮิวริสติกส์จำแนกได้เป็น 2 แนวทางหลัก [22, 167] คือ แบบ Parameter tuning และแบบ Parameter control

การกำหนดพารามิเตอร์แบบ Parameter tuning หรือแบบ Off-line parameter setting นั้น ค่าพารามิเตอร์ต่างๆ ของวิธีการเมต้าฮิวริสติกส์จะถูกกำหนดก่อนที่วิธีการดังกล่าวจะทำการประมวลผล [22] วิธีการกำหนดค่าพารามิเตอร์ในกลุ่มนี้ เช่น วิธีการ Ad hoc selection [43], วิธีการ Adopted approach [13], วิธีการ Best guess approach [44], วิธีการ One-factor-at-a-time [44], วิธีการ Factorial design [44] เป็นต้น อย่างไรก็ตามการค้นหาพารามิเตอร์ที่เหมาะสมด้วย 3 วิธีการแรกนั้น ไม่สามารถรับประกันได้เลยว่าจะทำให้วิธีการเมต้าฮิวริสติกส์สามารถค้นหาคำตอบที่ดีที่สุดได้ [44, 45] ขณะที่วิธีการ One-factor-at-a-time มีข้อเสียเปรียบตรงที่ไม่สามารถศึกษาผลกระทบร่วม (Interaction) ระหว่างปัจจัยได้ สุดท้ายแล้ววิธีการ Factorial design จะเป็นวิธีการที่ถูกต้องและมีประสิทธิภาพมากสุดสำหรับค้นหาค่าพารามิเตอร์ที่เหมาะสมที่สุดให้กับวิธีการเมต้าฮิวริสติกส์ เนื่องจากสามารถจัดการกับพารามิเตอร์จำนวนหลายตัวพร้อมกันได้โดยใช้หลักการออกแบบการทดลองและการวิเคราะห์ทางสถิติ (Experimental design and analysis) เข้ามาช่วย อีกทั้งยังสามารถศึกษาผลกระทบร่วม (Interaction) ระหว่างปัจจัยได้ด้วย [44]

การกำหนดพารามิเตอร์แบบ Parameter control หรือแบบ On-line parameter setting นั้น ค่าพารามิเตอร์ต่างๆ ของวิธีการเมต้าฮิวริสติกส์ จะถูกควบคุมและปรับเปลี่ยนค่าอยู่เสมอในขณะที่ Algorithms กำลังการประมวลผล [22] การกำหนดพารามิเตอร์แบบนี้มีความได้เปรียบในด้านการหลีกเลี่ยงการรันการทดลองจำนวนมากจากวิธีการออกแบบการทดลอง โดยเฉพาะอย่างยิ่งกับวิธีการเมต้าฮิวริสติกส์ชนิดที่มีพารามิเตอร์จำนวนมาก นอกจากนี้ยังไม่ต้องเสียเวลาและไม่ต้องใช้ทรัพยากรจำนวนมากในการรันการทดลองเพื่อหาค่าพารามิเตอร์ที่ดีที่สุดในแต่ละโจทย์แต่ละปัญหา สำหรับวิธีการกำหนดค่าพารามิเตอร์ในกลุ่มนี้จำแนกออกได้เป็น 2 แบบ [22] คือ วิธีการกำหนดค่าพารามิเตอร์แบบไม่คงที่ (Dynamic parameter update) และวิธีการกำหนดค่าพารามิเตอร์แบบปรับเปลี่ยนเองได้ (Adaptive parameter update)

อย่างไรก็ตามการกำหนดค่าพารามิเตอร์ที่ดีที่สุดแบบ Dynamic parameter update นอกจากจะมีความยากในการออกแบบกระบวนการแล้ว การปรับค่าของพารามิเตอร์โดยที่ไม่ได้พิจารณาถึงผลตอบสนองของคำตอบ (No feedback) ที่เปลี่ยนแปลงด้วยนั้นจัดเป็นข้อด้อยอีกข้อที่สำคัญของวิธีการในกลุ่มนี้ [167] ขณะที่วิธีการกำหนดค่าแบบ Adaptive parameter update จะมีความได้เปรียบกว่า คือ ค่าของพารามิเตอร์

จะถูกเปลี่ยนแปลงโดยที่จะพิจารณาถึงผลตอบสนอง (Feedback) ของคำตอบ (Solution) ที่เกิดขึ้นหลังจากปรับค่าขณะทำการค้นหาคำตอบด้วย [22] เช่น Self-adaptive parameter setting (SPS) เป็นต้น

ตัวอย่างวิธีการเมต้าฮิวริสติกส์หลายวิธีการที่ใช้การกำหนดค่าพารามิเตอร์แบบ Adaptive parameter update และประสบผลสำเร็จเป็นอย่างมากในการแก้ปัญหาการหาค่าที่เหมาะสมที่สุด เช่น วิธีการ Differential Evolution (DE) Algorithm [168, 169], วิธีการ Genetic Algorithm (GA) [49], วิธีการ Particle Swarm Optimisation (PSO) [170], วิธีการ Ant Colony Optimisation (ACO) [171] เป็นต้น แต่เป็นที่น่าเสียดายเมื่อสืบค้นในฐานข้อมูลวารสารทางวิชาการ (ISI และ Scopus) พบว่างานวิจัยที่ทำการศึกษาถึงประสิทธิภาพของวิธีการ CS โดยใช้วิธีการกำหนดค่าพารามิเตอร์แบบ Adaptive parameter update มีน้อยมาก โดยเฉพาะอย่างยิ่งกับปัญหาการจัดตารางเรียนตารางสอนซึ่งยังไม่พบงานวิจัยที่ประยุกต์ใช้วิธีการ CS โดยใช้วิธีการกำหนดค่าพารามิเตอร์แบบ SPS มาก่อน

2.7 การปรับปรุงประสิทธิภาพวิธีการเมต้าฮิวริสติกส์โดยการปรับปรุงกระบวนการ

การปรับปรุงประสิทธิภาพของวิธีการเมต้าฮิวริสติกส์ให้ดีขึ้นกว่าเดิมโดยการปรับปรุงกระบวนการ (Modification) ทำงานจัดเป็นอีกวิธีการหนึ่งที่พบได้บ่อย สำหรับปัญหาการจัดตารางเรียนตารางสอนพบว่าหลายงานวิจัยได้ปรับปรุงประสิทธิภาพของวิธีการเมต้าฮิวริสติกส์ในกระบวนการหรือในขั้นตอนที่แตกต่างกันออกไปขึ้นอยู่กับวัตถุประสงค์ของการปรับปรุงในกระบวนการนั้นๆ ซึ่งผลของการปรับปรุงต่างก็ประสบผลสำเร็จเป็นอย่างดี เช่น กระบวนการ Initial solution [46, 47], กระบวนการ Solution evolution [48], กระบวนการ Fitness calculation [19], กระบวนการ Selection process [49], กระบวนการ Solution replacement [10], กระบวนการ Accepted rules [19], กระบวนการ Memory update [50] เป็นต้น

2.8 การปรับปรุงประสิทธิภาพวิธีการเมต้าฮิวริสติกส์โดยการผสมผสาน

การปรับปรุงประสิทธิภาพของวิธีการเมต้าฮิวริสติกส์ให้ดีขึ้นกว่าเดิมโดยการผสมผสาน (Hybridisation) จัดเป็นอีกวิธีการหนึ่งที่พบได้บ่อย เนื่องจากการแก้ปัญหาการหาค่าที่เหมาะสมที่สุดหลายประเภทในปัจจุบันนั้น การใช้วิธีการเมต้าฮิวริสติกส์เพียงวิธีการเดียวปกติแล้วจะไม่เหมาะกับการนำไปประยุกต์ใช้แก้ปัญหาที่มีความยากมากๆ เพราะผลลัพธ์ที่ได้อาจจะไม่ดีเท่าที่ควร [22] จากการทบทวนวรรณกรรมที่ผ่านมาเกี่ยวกับการผสมผสานของวิธีการเมต้าฮิวริสติกส์ที่พบในปัญหาการจัดตารางเรียนตารางสอน สามารถจำแนกออกเป็น 3 แบบคือ วิธีการ S-meta ผสมผสานกับ S-meta, วิธีการ P-meta ผสมผสานกับ P-meta และวิธีการ S-meta ผสมผสานกับ P-meta

สำหรับวิธีการ S-meta ผสมผสานกับ S-meta จะเป็นการนำวิธีการในกลุ่มของ Single solution based metaheuristics มาทำการผสมผสานร่วมกัน เช่น วิธีการ Simulated Annealing (SA) กับวิธีการ Hill climbing algorithm [58], วิธีการ SA กับวิธีการ Large Neighborhood Search (VNS) [172], วิธีการ Greedy heuristic กับ Local search (LS) [125] เป็นต้น ในขณะที่วิธีการ P-meta ผสมผสานกับ P-meta

จะเป็นการนำวิธีการในกลุ่มของ Population based metaheuristics มาทำการผสมผสานร่วมกัน เช่น Genetic Algorithm (GA) กับ Particle Swarm Optimisation (PSO) [173] เป็นต้น แบบสุดท้ายเป็นวิธีการ S-meta ผสมผสานกับ P-meta จะเป็นแนวทางที่ได้รับความนิยมมากที่สุดของการผสมผสานและยังเป็นวิธีการที่มีประสิทธิภาพสูงในการค้นหาคำตอบ [22, 51, 52] เพราะว่า จะเป็นการใช้ข้อดีของการค้นหาคำตอบในวงกว้างซึ่งได้จาก P-meta (Exploration) ร่วมกับการใช้ข้อดีของการค้นหาคำตอบในวงแคบซึ่งได้จาก S-meta (Exploitation) ทำให้วิธีการผสมผสานแบบนี้เกิดความสมดุลกันจากหลักการค้นหาคำตอบทั้ง 2 แบบ (Diversification และ Intensification) [22, 51, 52] ดังนั้นวิธีการผสมผสานในกลุ่มนี้จึงได้รับความนิยมนำมาใช้แก้ปัญหาต่างๆ และประสบความสำเร็จเป็นอย่างดี ซึ่งรวมไปถึงปัญหาการจัดตารางการศึกษาด้วย ตัวอย่างเช่น วิธีการ Ant Colony System (ACS) กับ TS [53], วิธีการ GA+LS (Memetic Algorithm: MA) [54], วิธีการ GA+TS [12], วิธีการ PSO+LS [48, 55], วิธีการ Best-worst ACS กับ LS [13] เป็นต้น

บทที่ 3

วิธีดำเนินงานวิจัย

3.1 ทำการเก็บและวิเคราะห์ข้อมูลการจัดตารางเรียนตารางสอน

ปัญหาการจัดตารางเรียนตารางสอนของคณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร จัดเป็นกิจกรรมที่เกิดขึ้นเป็นประจำทุกเทอมการศึกษาและยังขาดวิธีการจัดตารางที่มีประสิทธิภาพ เนื่องจากระบบการจัดตารางเรียนตารางสอนของมหาวิทยาลัยนเรศวรยังเป็นแบบกึ่งอัตโนมัติและมีกระบวนการจัดตารางแบบ 3 ขั้นตอน เริ่มต้นขั้นตอนแรกจากวิชาบังคับพื้นฐานของมหาวิทยาลัยทั้งหมด (ปกติจะพบในนิสิตชั้นปีที่ 1-2) จะถูกนำมาจัดตารางก่อนโดยผู้จัดตารางจากส่วนกลางของมหาวิทยาลัย เมื่อขั้นตอนแรกเสร็จสิ้นแล้วขั้นตอนที่สองจะเป็นการจัดตารางเรียนตารางสอนในรายวิชาที่เหลือซึ่งเป็นรายวิชาพื้นฐานของแต่ละคณะโดยผู้จัดตารางของแต่ละคณะ และขั้นตอนที่สามจะเป็นการจัดตารางในระดับภาควิชาหลังจากรายวิชาพื้นฐานของแต่ละคณะถูกจัดเสร็จสิ้นแล้ว ดังนั้นจะเห็นได้ว่าการจัดตารางเรียนตารางสอนในแต่ละเทอมนั้นมีขั้นตอนจำนวนมาก ใช้เวลานาน และจะต้องใช้บุคคลากรจำนวนมากในการจัดตารางเรียนตารางสอนทั้งมหาวิทยาลัย

นอกจากนี้จากการสืบค้นข้อมูลจากฐานข้อมูลในระบบจัดตารางเรียนของมหาวิทยาลัยนเรศวร (www.reg.nu.ac.th) พบว่า ขนาดของข้อมูลในการตารางเรียนตารางสอนของคณะวิศวกรรมศาสตร์จะมีขนาดใหญ่มากและมีแนวโน้มเพิ่มขึ้นอย่างต่อเนื่อง ยกตัวอย่างเช่น ในเทอม 1 ปีการศึกษา 2555 คณะเปิดสอนทั้งหมด 51 หลักสูตรทั้งในระดับปริญญาตรีทุกชั้นปี (ทั้งภาคปกติและภาคพิเศษ) ปริญญาโททุกชั้นปี (ทั้งภาคปกติและภาคพิเศษ) และปริญญาเอกทุกชั้นปีรวมทั้งหมด 70 ชั้นเรียน (Classes) มีจำนวนอาจารย์ทั้งหมดมากกว่า 95 คน มีจำนวนนิสิตในคณะเฉพาะระดับปริญญาตรี 1800 คน มีวิชาเรียนวิชาสอนมากกว่า 230 วิชาต่อเทอมและต้องจำแนกหมู่เรียนมากกว่า 300 กลุ่ม มีห้องเรียนที่สามารถใช้ได้ 111 ห้องจากจำนวนทั้งหมด 134 ห้อง เป็นต้น ดังนั้นการจัดตารางเรียนตารางสอนทั้งหมดของคณะซึ่งเป็นแบบกึ่งอัตโนมัติจะมีความยุ่งยากและซับซ้อนมากในการจัดตารางทั้งหมดให้ไม่มีการชนกันและต้องเป็นไปตามความต้องการของผู้ใช้ตารางทั้งหมดอีกด้วย จึงทำให้การจัดตารางเรียนตารางสอนทั้งหมดของมหาวิทยาลัยจะต้องใช้เวลาที่นานมาก (ประมาณ 2-4 สัปดาห์) ในแต่ละเทอมการศึกษา ดังนั้นการนำข้อมูลการจัดตารางเรียนตารางสอนจริงของคณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร มาสร้างเป็นโจทย์ปัญหาในงานวิจัยนี้น่าจะเป็นอีกแนวทางหนึ่งในการพัฒนากระบวนการจัดตารางเรียนตารางสอนให้กับมหาวิทยาลัยนเรศวรให้ดียิ่งขึ้นต่อไป

การเก็บรวบรวมข้อมูล (Data collection) ข้อมูลการจัดตารางเรียนตารางสอนจริงของคณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร ถูกเก็บรวบรวมมาจากระบบการจัดตารางสอนของมหาวิทยาลัยโดยเลือกเก็บข้อมูลของปี 2555 เนื่องจากแผนการศึกษาในขณะนั้นมีความหลากหลายมากกว่าปัจจุบัน เช่น หลักสูตรระดับปริญญาตรีจะมีเปิดทั้งภาคปกติและภาคพิเศษ หลักสูตรระดับปริญญาโทมีเปิดทั้งภาคปกติและภาคพิเศษ เป็นต้น โดยที่นิสิตภาคปกติและภาคพิเศษของแต่ละหลักสูตรจะมีข้อจำกัดในเรื่องของเวลาที่ต้องจัดการเรียนการสอนที่แตกต่างกันออกไป จึงทำให้ปัญหาการจัดตารางสอนในปีดังกล่าวมีความยุ่งยากและ

ซับซ้อนอย่างมาก อย่างไรก็ตามรูปแบบโจทย์ปัญหาของข้อมูลดังกล่าวสามารถรองรับข้อมูลปัญหาการจัดตารางเรียนตารางสอนในปัจจุบันได้อีกด้วย สำหรับองค์ประกอบของข้อมูลที่สำคัญในการเก็บรวบรวมมีดังต่อไปนี้

1. ข้อมูลระดับการศึกษาและสาขาวิชา (Degrees/Programs)

สำหรับข้อมูลสาขาวิชาในหลักสูตรของคณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร จะอยู่ในความรับผิดชอบของ 4 ภาควิชาหลักคือ ภาควิชาวิศวกรรมโยธา ภาควิชาวิศวกรรมอุตสาหการ ภาควิชาวิศวกรรมเครื่องกล ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ ซึ่งในแต่ละภาควิชาก็จะมีการเปิดสอนในหลายสาขาวิชาและแผนการศึกษาที่แตกต่างกันออกไป ตั้งแต่หลักสูตรระดับปริญญาตรีไปจนถึงหลักสูตรระดับปริญญาเอก ยกตัวอย่างในเทอม 1 ปีการศึกษา 2555 นั้น คณะวิศวกรรมศาสตร์ได้เปิดการเรียนการสอนใน 11 สาขาวิชา

2. ข้อมูลของหลักสูตร (Curricula data)

เนื่องจากนโยบายการพัฒนาระบบการศึกษาของมหาวิทยาลัยจึงทำให้แต่ละสาขาวิชาที่เปิดสอนในทุกระดับการศึกษาจะต้องมีการปรับปรุงบางรายวิชาในหลักสูตรให้มีความเหมาะสมอยู่เสมอในทุก 2-4 ปี จึงทำให้ในหลายสาขาวิชาจะมีรหัสหลักสูตร (Curriculum codes) มากกว่า 1 รหัส ดังนั้นการกำหนดรหัสหลักสูตรในแต่ละสาขาวิชาจึงเป็นสิ่งที่สำคัญมากในการกำหนดแผนการจัดตารางสอนที่ถูกต้องให้กับแต่ละหลักสูตรในแต่ละสาขาวิชาของนิสิตในแต่ละชั้นปี เพราะเมื่อมีนิสิตสาขาวิชาเดียวกันแต่มีรหัสหลักสูตรต่างกันและกำลังศึกษาอยู่ เช่น นิสิตปี 1 ใช้หลักสูตรวิศวกรรมอุตสาหการปี 55 ขณะที่นิสิตปี 2 ถึงปี 4 ยังใช้หลักสูตรวิศวกรรมอุตสาหการปี 51 เป็นต้น จะทำให้การจัดตารางในบางเทอมอาจมีจำนวนวิชาเรียนวิชาสอนถูกเพิ่มเข้ามาหรือลดลงมากกว่าปกติ หรืออาจมีจำนวนหมู่เรียนในบางวิชาถูกเพิ่มเข้ามาหรือลดลงมากกว่าปกติได้

สำหรับรหัสหลักสูตรทั้งหมดจากทุกสาขาวิชาของคณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร เฉพาะระดับปริญญาตรี แบบ 2 ภาคการศึกษา (ปกติ) ประจำปีการศึกษา 2555 มีจำนวนมากถึง 48 รหัสหลักสูตร โดยข้อมูลจากตารางดังกล่าวจะประกอบไปด้วยรหัสหลักสูตร ชื่อสาขาวิชาและสถานะของหลักสูตรในปัจจุบัน

3. ข้อมูลวิชาเรียนวิชาสอน (Courses data)

สำหรับข้อมูลของวิชาเรียนวิชาสอนในแต่ละชั้นเรียน (Class) ในแต่ละเทอมการศึกษาจะพิจารณาตามหลักสูตรที่ได้กำหนดไว้ในแต่ละระดับการศึกษา โดยคณะวิศวกรรมศาสตร์จะมีทั้งวิชาพื้นฐานและวิชาเลือกหรือวิชาเฉพาะด้าน ซึ่งในแต่ละวิชาจะมีการกำหนดหน่วยกิจที่ต้องการและจำนวนคาบเวลาที่ต้องจัดการเรียนการสอน บางรายวิชาจะมีเฉพาะภาคบรรยายขณะบางรายวิชามีทั้งภาคบรรยายและภาคปฏิบัติ ในบางรายวิชาที่มีจำนวนผู้เรียนจำนวนมากกว่าขนาดหรือจำนวนของห้องเรียนก็อาจมีการแยกเป็นกลุ่มหรือหมู่เรียนมากกว่า 1 กลุ่ม โดยในแต่ละกลุ่มย่อยจะมีการกำหนดนิสิต/ชั้นเรียนที่ต้องเรียนวิชาดังกล่าวพร้อมทั้งระบุถึงจำนวนของนิสิตในแต่ละหมู่เรียนด้วย นอกจากนี้ในแต่ละกลุ่มอาจมีการกำหนดอาจารย์ผู้สอนคนเดียวกันหรือ

ผู้สอนคนละคนก็ได้ขึ้นอยู่กับภาระงานของอาจารย์แต่ละท่าน ยิ่งไปกว่านั้นบางรายวิชาอาจมีการกำหนดอาจารย์ผู้สอนมากกว่า 1 ท่านได้

4. ข้อมูลของนิสิต (Student data)

ในงานวิจัยนี้ข้อมูลของนิสิตจะพิจารณาจากจำนวนนิสิตในทุกระดับการศึกษา ทุกหลักสูตรของแต่ละสาขาวิชาที่มีนิสิตเข้าศึกษา ซึ่งทั้งหมดอยู่ในความรับผิดชอบของ 4 ภาควิชาของคณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร สำหรับจำนวนนิสิตที่รับเข้าศึกษาในแต่ละปีการศึกษาจะมีจำนวนที่แตกต่างกันออกไปในแต่ละระดับการศึกษาและสาขาวิชา โดยเฉพาะอย่างยิ่งในระดับปริญญาตรีจะมีจำนวนนิสิตเป็นจำนวนมาก ขณะที่ในระดับปริญญาโทและเอกจะมีจำนวนนิสิตน้อยลง นอกจากนิสิตในสาขาเดียวกันจะมีรหัสหลักสูตรที่ต่างกันในบางชั้นปีระหว่างปีการศึกษา 2552 ถึง 2555 แล้ว ยังพบอีกว่าจำนวนนิสิตทุกสาขาวิชาตั้งแต่ชั้นปี 1 ถึงปี 4 มีจำนวนมากถึง 1,796 คนซึ่งเมื่อรวมกับนิสิตปริญญาตรีที่ตกแผน (ชั้นปี 5 ขึ้นไป) และนิสิตปริญญาตรีภาคพิเศษ รวมถึงนิสิตทุกคนในระดับปริญญาโทและปริญญาเอกด้วยแล้ว จะทำให้มีจำนวนนิสิตที่อยู่ในระบบการศึกษาจำนวนมากกว่า 2500 คน และจะมีแนวโน้มเพิ่มขึ้นหากมีการเปิดสอนในสาขาวิชาอื่นเพิ่มเติมในอนาคต

5. ข้อมูลของห้องเรียน (Classroom data)

สำหรับข้อมูลของห้องเรียนจะอยู่ใน 4 อาคารเรียนของคณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร ประกอบไปด้วย อาคารเรียนคณะวิศวกรรมศาสตร์สาขาวิศวกรรมโยธา (CE) อาคารเรียนคณะวิศวกรรมศาสตร์สาขาวิศวกรรมไฟฟ้า (EE) อาคารเรียนรวมคณะวิศวกรรมศาสตร์ (EN) และอาคารเรียนคณะวิศวกรรมศาสตร์สาขาวิศวกรรมอุตสาหการ (IE) โดยรายละเอียดของห้องเรียนในแต่ละอาคารเรียนประกอบไปด้วย รหัสห้องเรียน ชื่อห้องเรียน ประเภทหรือคุณลักษณะเฉพาะของห้องเรียน ความจุห้องเรียน และสถานะของห้องเรียน

6. ข้อมูลวันและคาบเวลา (Days/timeslots data)

จำนวนวัน (Days: D) และจำนวนช่วงเวลา (Periods: P) สำหรับโจทย์ปัญหาการจัดตารางเรียนตารางสอนในระดับอุดมศึกษานั้น ปกติแล้วจำนวนวันสำหรับการเรียนการสอนต่อสัปดาห์ (Days/ Week) จะกำหนดให้เท่ากันในแต่ละสัปดาห์และใช้เหมือนกันทั้งสถาบัน เช่น กำหนด 5 วันต่อสัปดาห์ เป็นต้น ในขณะที่จำนวนคาบเรียนต่อวัน (Periods/Day) นั้นจะถูกกำหนดให้มีจำนวนช่องของเวลา (Timeslots) ที่เท่ากันในแต่ละวันและใช้เหมือนกันทั้งสถาบัน เช่น กำหนดให้มี 6 คาบหรือ 8 คาบต่อวัน เป็นต้น ดังนั้นจำนวนช่องเวลาทั้งหมดที่สามารถจัดตารางเรียนตารางสอนได้ต่อสัปดาห์ (Total timeslots/week) สำหรับหนึ่งห้องเรียน (Classroom) คือ ผลคูณของจำนวนวันต่อสัปดาห์ (Days/ Week) คูณกับจำนวนคาบเวลาต่อวัน (Periods/Day)

สำหรับข้อมูลวันและคาบเวลาของคณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวรนั้น จำนวนคาบเวลาที่สามารถจัดการเรียนการสอนได้ในแต่ละห้องเรียนนั้นสามารถจัดได้มากถึง 16 คาบต่อวัน ตั้งแต่เวลา 7.00-23.00 น. เนื่องจากต้องการรองรับกับนิสิตภาคพิเศษที่มีการเรียนการสอนในภาคค่ำด้วย ในแต่ละสัปดาห์จะมีการใช้ห้องเรียนทุกวันโดยวันจันทร์-ศุกร์จัดการเรียนการสอนสำหรับนิสิตภาคปกติ ส่วนวันเสาร์-อาทิตย์จัดการเรียนการสอนสำหรับนิสิตปริญญาโทภาคพิเศษหรือนิสิตปริญญาตรีในบางวิชา ดังนั้นเมื่อพิจารณาจำนวนคาบเวลาทั้งหมดจากทุกห้องเรียนที่สามารถรองรับการเรียนการสอนได้มากที่สุด 12,432 (111*7*16) คาบต่อสัปดาห์

7. ข้อมูลของอาจารย์ (Lecturer data)

สำหรับข้อมูลของอาจารย์จะพิจารณาจาก 4 ภาควิชาหลักของคณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร ประกอบไปด้วย อาจารย์ในภาควิชาวิศวกรรมโยธา (CE) อาจารย์ในภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ (EE & ComE) อาจารย์ในภาควิชาวิศวกรรมเครื่องกล (ME) และอาจารย์ในภาควิชาวิศวกรรมอุตสาหการ (IE) นอกจากนี้ยังมีอาจารย์บางท่านที่สังกัดโดยตรงกับคณะวิศวกรรมศาสตร์ (Eng) ด้วย เพื่อให้การศึกษาวิจัยการจัดตารางเรียนตารางสอนของคณะวิศวกรรมศาสตร์สามารถดำเนินไปได้และสามารถเผยแพร่ผลการศึกษาวิจัยได้ ดังนั้นข้อมูลเกี่ยวกับชื่อของอาจารย์แต่ละท่านในงานวิจัยนี้จะไม่ถูกแสดงแต่จะถูกกำหนดด้วยการใช้รหัสของอาจารย์แทน

3.2 การกำหนดข้อบังคับของปัญหา (Hard and Soft constraints)

สำหรับปัญหาการจัดตารางเรียนตารางสอนในระดับอุดมศึกษาในงานวิจัยนี้จะใช้ข้อบังคับหลัก (HC) และข้อบังคับรอง (SC) แบบที่มีการใช้อย่างแพร่หลาย (Global constraints) โดยข้อบังคับ HC ในงานวิจัยนี้จะมีจำนวน 6 ข้อ ประกอบไปด้วย

- HC1 (Lectures) จำนวนการเรียนการสอน (Lectures) ของแต่ละวิชา (Course) จะต้องถูกจัดลงตารางเรียนตารางสอนให้ครบทุกครั้ง (หรือครบตามจำนวนหน่วยกิจ) และในแต่ละครั้ง (Lecture) ต้องกำหนดลงในช่วงเวลาที่แตกต่างกัน

- HC2 (Conflicts) การเรียนการสอนทุกครั้ง (Lectures) ของทุกวิชา (Courses) ในแต่ละหลักสูตร (Curriculum/Student) และในแต่ละอาจารย์ผู้สอน (Lecturer) จะต้องถูกจัดลงในช่วงเวลาที่แตกต่างกัน

- HC3 (Room occupancy) ในแต่ละห้องเรียน (Classroom) ห้ามกำหนดการเรียนการสอนมากกว่า 1 วิชาในช่วงเวลาเดียวกัน

- HC4 (Lecturer/student availability) ห้ามจัดวิชาสอนใดๆ ให้กับอาจารย์/นิสิต ในวัน (Day) และคาบเวลา (Timeslots) ที่อาจารย์/นิสิต ไม่สามารถจัดการเรียนการสอนได้

- HC5 (Room suitability) ในแต่ละวิชาจะต้องถูกจัดลงในอาคารเรียน (Buildings) ประเภทของห้องเรียน (Lecture/laboratory) และห้องเรียนที่มีสิ่งอำนวยความสะดวกหรือคุณลักษณะ (Feature) ตามที่กำหนดไว้

- HC6 (Double/Multiple lectures) ในรายวิชาที่ต้องการจัดการเรียนการสอนในคาบเวลาที่ต่อเนื่องกัน เช่น 2 คาบหรือ 3 คาบต่อเนื่อง (Double/triple booking periods) เป็นต้น โดยเฉพาะอย่างยิ่งในรายวิชาที่มีหน่วยกิจในส่วนของภาคปฏิบัตินั้น จะต้องทำการจัดให้มีการเรียนการสอนอย่างต่อเนื่องตามที่ได้กำหนดไว้

สำหรับข้อบังคับรอง (SC) ในงานวิจัยนี้จะมีจำนวน 3 ข้อ ประกอบไปด้วย

- SC1 (Overhead costs) มีความเกี่ยวข้องกับข้อบังคับแบบ Room suitability โดยห้องเรียนแต่ละห้องจะมีค่าใช้จ่ายที่แตกต่างกันออกไปตามขนาดของห้อง ประเภทของห้อง ลักษณะพิเศษของห้อง รวมไปถึงค่าใช้จ่ายที่แตกต่างกันออกไปในแต่ละวันและช่วงเวลา ดังนั้นวิชาเรียนวิชาสอนทุกวิชาควรเลือกใช้ห้องสำหรับจัดการเรียนการสอนที่เหมาะสม เพื่อลดค่าใช้จ่ายในการเช่าหรือค่าดำเนินงานรวม

- SC2 (Lecturing costs) มีความเกี่ยวข้องกับข้อบังคับแบบ Timeslot preference โดยวิชาสอน (Courses) ของอาจารย์ควรถูกจัดลงในวัน (Day) และช่วงเวลา (Timeslots) ที่อาจารย์ชอบและหลีกเลี่ยงช่วงเวลาที่ไม่ชอบ โดยค่าถ่วงน้ำหนักที่แสดงถึงความชอบ (มีค่าน้อย) และความไม่ชอบ (มีค่ามาก) ของอาจารย์แต่ละท่านได้ถูกกำหนดไว้แล้วในแต่ละวันและในแต่ละช่วงเวลา ก่อนนำไปพิจารณาเป็นค่าใช้จ่ายในการจ้างอาจารย์

- SC3 (Setup/cleaning costs) มีความเกี่ยวข้องกับข้อบังคับแบบ Consecutive lectures โดยห้องเรียนแต่ละห้องจะมีค่าใช้จ่ายที่เกี่ยวข้องกับจัดเตรียมอุปกรณ์หรือการทำความสะอาดที่แตกต่างกันออกไปตามขนาดของห้อง ประเภทของห้อง ลักษณะพิเศษของห้อง ซึ่งรวมถึงค่าใช้จ่ายดังกล่าวจะมีความแตกต่างกันออกไปในแต่ละวันและช่วงเวลา ดังนั้นตารางการใช้ห้องเรียนแต่ละห้อง ควรมีการใช้อย่างต่อเนื่อง (หรือลดช่วงเวลาว่างระหว่าง 2 วิชาในแต่ละวันให้มากที่สุด) เพื่อลดค่าใช้จ่ายในการจัดเตรียมอุปกรณ์หรือการทำความสะอาดรวม

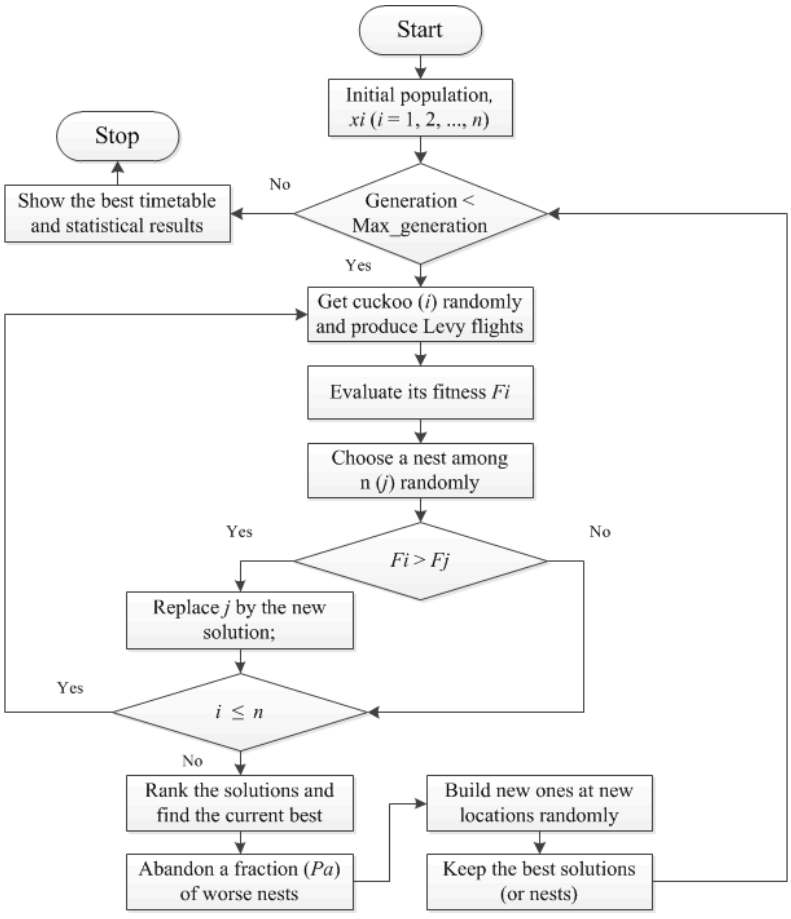
โดยงานวิจัยมีเป้าหมายเพื่อจัดตารางเรียนตารางสอนโดยให้มีต้นทุนรวม (Z) ที่เกิดจากผลรวมข้อบังคับรองทั้งสามข้อ (SC1-SC3) ให้น้อยที่สุด (Minimisation) ในขณะที่ข้อบังคับหลักทั้ง 6 ข้อ (HC1-HC6) จะต้องไม่มีการละเมิดอย่างเด็ดขาด

3.3 วิธีการคุกคูเสิร์ช (Cuckoo Search: CS)

วิธีการคุกคูเสิร์ช (Cuckoo Search: CS) จัดเป็นหนึ่งในวิธีการเมต้าฮิวริสติกส์แบบใหม่ในกลุ่มของ P-meta ที่ได้แรงดลใจมาจากธรรมชาติ (Nature-inspired metaheuristic algorithms) เหมือนกับวิธีการเมต้า

ฮิวริสติกส์แบบอื่นๆ โดยถูกพัฒนาขึ้นโดย Yang และ Deb ในปี 2009 [23] วิธีการ CS นี้ได้แนวคิดมาจากพฤติกรรมการฝากฟักไข่ของนกกาเหว่าไว้กับรังของนกสายพันธ์อื่น (Host birds) โดยพยายามลอกเลียนแบบไข่ของตนให้เหมือนกับไข่ของนกเจ้าของรังเพื่อความอยู่รอดของเผ่าพันธุ์ตนเอง จากพฤติกรรมอันชาญฉลาดดังกล่าวทำให้ Yang และ Deb เกิดแนวคิดในการนำพฤติกรรมการฝากฟักไข่ของนกกาเหว่ามาพัฒนาเป็น Algorithm แบบใหม่จนกระทั่งได้เป็นวิธีการ CS ขึ้นมา [37]

กระบวนการทำงานทั่วไปของวิธีการ CS สามารถแสดงดังภาพ 1 [23] โดยเริ่มต้นจากกระบวนการสร้างคำตอบเริ่มต้น (Initial population) ตามจำนวนประชากร (Host nests: n) ที่กำหนดไว้ ถ้าจำนวนรอบการค้นหายังไม่สิ้นสุดก็จะทำการสุ่มนกกาเหว่ามาพัฒนาคำตอบด้วยวิธีการ Levy flights แล้วทำการคำนวณความสมบูรณ์ (Fitness: F_i) หลังจากนั้นทำการสุ่มรังนกขึ้นมาแล้วนำความสมบูรณ์ของรังที่สุ่มได้ (F_j) มาเปรียบเทียบกับความสมบูรณ์ของนกกาเหว่าก่อนหน้านี้ (F_i) ถ้าคำตอบใหม่ดีกว่าก็ทำการแทนที่คำตอบเดิม เมื่อนกกาเหว่าทุกตัวได้ทำการพัฒนาคำตอบครบทุกตัวแล้วก็จะทำการเรียงความสมบูรณ์ของคำตอบทั้งหมด n คำตอบจากมากไปน้อย โดยตำตอบที่มีคุณภาพแย่จำนวน P_a ตัวจะถูกละทิ้งไปและทำการสุ่มขึ้นมาใหม่แทน กระบวนการนี้จะวนซ้ำไปจนกระทั่งสิ้นสุดเงื่อนไขของการประมวลผล



ภาพ 1 แสดงแผนผังการไหลของวิธีการ CS แบบทั่วไป [23]

ถึงแม้จะเป็นวิธีการที่ค่อนข้างใหม่ แต่ในช่วงหลายปีที่ผ่านมาวิธีการ CS กลับได้รับความนิยมนำมาประยุกต์ใช้แก้ปัญหาแบบ Optimisation อย่างแพร่หลายและประสบผลสำเร็จเป็นอย่างดี เช่น ปัญหาการหาค่าที่เหมาะสมที่สุดเชิงตัวเลข (Numerical optimisation problems) [33], ปัญหาการหาค่าที่เหมาะสมทางวิศวกรรม (Engineering optimisation problems) [23], ปัญหาการจัดตารางการเข้าเวรของพยาบาล (Nurse scheduling problems) [35], ปัญหาการจัดตารางการผลิต (Production scheduling problems) [36], ปัญหาการจัดถุงกระสอบ (Knapsack problems) [174], ปัญหาการบรรจุลงกล่อง (Bin packing problems) [175] เป็นต้น แต่เป็นที่น่าเสียดายที่ยังไม่พบว่ามีการนำวิธีการ CS มาประยุกต์ใช้แก้ปัญหาการจัดตารางเรียนตารางสอนระดับมหาวิทยาลัยมาก่อน เมื่อสืบค้นในฐานข้อมูลวารสารทางวิชาการระดับนานาชาติ ISI web of knowledge, Scopus, และ IEEE Xplore ทั้งในส่วนที่เป็นชื่อเรื่อง (Title) บทคัดย่อ (Abstract) และคำสำคัญ (Keyword) (โดยใช้คำสำคัญในการสืบค้น คือ (“cuckoo”) AND (“course timetabling” OR “course timetable” OR “course scheduling” OR “course schedule” OR “course assignment” OR “course allocation”)

นอกจากนี้หลายงานวิจัยยังได้ทำการเปรียบเทียบประสิทธิภาพในการแก้ปัญหาของวิธีการ CS กับวิธีการเมต้าฮิวริสติกส์แบบอื่นอีกด้วย เช่น ปัญหาการหาค่าที่เหมาะสมที่สุดเชิงตัวเลขพบว่า วิธีการ CS มีประสิทธิภาพในการหาคำตอบที่ดีกว่า GA, PSO [37] และ ABC [33] ในปัญหาปัญหาการจัดตารางการผลิตพบว่า วิธีการ CS มีประสิทธิภาพในการค้นหาคำตอบที่ดีกว่า GA [36] นอกจากนี้ในปัญหา Milling operation ยังพบอีกว่าวิธีการ CS มีประสิทธิภาพในการหาคำตอบที่ดีกว่าทั้งวิธีการ GA, ACO, AIS และ PSO [38] เป็นต้น ดังนั้นจะเห็นได้ว่าวิธีการ CS เป็นวิธีการหนึ่งที่น่าสนใจอย่างมาก เพราะว่านอกจากจะมีจำนวนพารามิเตอร์ที่ต้องปรับค่าให้เหมาะสมน้อยกว่าวิธีการเมต้าฮิวริสติกส์แบบอื่นแล้ว เช่น GA และ PSO [37] เป็นต้น วิธีการ CS มีประสิทธิภาพในการหาคำตอบที่ดีเมื่อเปรียบเทียบกับวิธีการเมต้าฮิวริสติกส์หลายวิธีการจากการประยุกต์ใช้กับปัญหาหลายประเภท

3.4 วิธีการคุกคูเสิร์ช (CS) ในการแก้ปัญหาการจัดตารางเรียนตารางสอน

กระบวนการทำงานของวิธีการ CS สำหรับงานวิจัยนี้สามารถแสดงขั้นตอนการทำงานได้ดังภาพ 2 และอธิบายได้ดังนี้

1. นำเข้าข้อมูลของปัญหาที่ได้ทำการออกแบบไว้แล้วและกำหนดฟังก์ชันวัตถุประสงค์ (Objective function) ของปัญหา

2. กำหนดจำนวนการค้นหาคำตอบ (*PI*) และค่าพารามิเตอร์ความน่าจะเป็นในการละทิ้งคำตอบ (P_a) สำหรับวิธีการ CS

3. สร้างคำตอบเริ่มต้น (initial population) ตามจำนวนประชากร (*P*) ที่ได้กำหนดไว้ โดยคำตอบ 1 คำตอบจะประกอบด้วย วิชาเรียนวิชาสอนทั้งหมดที่ถูกจัดลงในห้องเรียนที่กำหนดไว้แล้ว ซึ่งสามารถแสดงตารางสอนของอาจารย์ทุกท่าน ตารางเรียนของนักเรียนทุกคน และตารางการใช้ห้องเรียนทุกห้องได้

4. พัฒนาคำตอบใหม่ (Solution evolution) โดยการสุ่มเลือกคำตอบ x_i ในกลุ่มประชากร P ขึ้นมา 1 รัง หลังจากนั้นทำการปรับปรุงคำตอบ x_i ที่ถูกสุ่มเลือกมาด้วยหลักการ Levy Flights ดังสมการ (1) [23]

$$x_i^{(t+1)} = x_i^{(t)} + \alpha \oplus Levy(\lambda) \quad (1)$$

โดยที่ $x_i^{(t)}$ คือ คำตอบเดิม และ $x_i^{(t+1)}$ คือ คำตอบใหม่ทีได้รับการปรับปรุงแล้ว ในขณะที่ $\alpha \oplus$ Levy(λ) คือ ช่วงระยะห่างหรือระยะทางที่เกิดการเปลี่ยนแปลงตำแหน่งของคำตอบ สำหรับ α เป็นพารามิเตอร์ที่มีความเกี่ยวข้องกับขนาดในการปรับปรุงคำตอบ (step size) ซึ่งขึ้นอยู่กับขนาดของปัญหาที่ต้องการค้นหาคำตอบและจะมีค่ามากกว่า 0 ($\alpha > 0$) [23]

Begin Input course timetabling data and set objective function $f(x)$, $x = (x_1, ..., x_d)^T$
Set amount of CS's search including numbers of Population (P) and MaxIteration (I)
Create initial population of P solutions x_i (i = 1, 2, 3, ..., P)
While t < MaxIteration **do**
For (i = 1, i <= P, i++) **do**
Get a cuckoo (say, x_i) randomly
Generate a solution x_i' by Lévy flights (CSLF) or Gaussian random walks (CSGRW)
If (x_i' = an infeasible timetable) **do**
Repair the x_i' to be a feasible timetable
End if
Evaluate its fitness $f(x_i')$
Choose a nest among n (say, x_j) randomly
If $f(x_i') > f(x_j)$ **do**
Replace the x_j by the new solution x_i'
End if
If Local Search **do**
Apply Exchange Operator (EO) or Insertion Operator (IO)
If (x_i' = an infeasible timetable) **do**
Repair the x_i' to be a feasible timetable
End if
End if
End for
Rank the solutions and find the current best solution
If ($rand < P_a$ of worse nests: x_{worse}) **do**
Build/generate new solutions x_{new}
Replace the x_{worse} by the new solution x_{new}
End if
Keep the best so far timetable
End while
Postprocess results and visualisation
End

ภาพ 2 รหัสเทียมของวิธีการ CS เพื่อแก้ปัญหาการจัดตารางสอน

อย่างไรก็ตามงานวิจัยนี้ได้ปรับปรุงเปลี่ยนแปลงกระบวนการค้นหาคำตอบของวิธีการ CS โดยใช้การเดินสุ่มแบบ Gaussian Random Walks (หรือเรียกว่า วิธีการ CSGRW) ได้ดังสมการ (2)

$$x_i^{(t+1)} = x_i^{(t)} + \alpha_0 \oplus randn() \quad (2)$$

โดย α_0 คือค่าคงที่ระหว่าง 0.01-0.001 ในขณะที่ตัวแปร *randn*() จะเป็นค่าสุ่มที่เกิดจาก Gaussian distribution หรือการแจกแจงแบบมาตรฐานหรือ $N(0,1)$ ที่มีค่าเฉลี่ยเป็น 0 และค่าเบี่ยงเบนมาตรฐานเป็น 1

5. หากพบการชนกันของตารางใดๆ (Infeasible solution) ให้ทำการซ่อมแซมคำตอบที่มีการชนกันของตารางเรียน ตารางสอน และตารางการใช้ห้องเรียนทั้งหมด

6. ประเมินค่าความเหมาะสม (Fitness: F_i) หรือคุณภาพของคำตอบ ที่ได้รับการปรับปรุงแล้วหรือ $x_i^{(t+1)}$ โดยใช้ฟังก์ชันวัตถุประสงค์ $f(x)$ เพื่อค้นหาตารางเรียนตารางสอนที่มีต้นทุนรวมน้อยที่สุด

7. ทำการสุ่มเลือกคำตอบในกลุ่มประชากร P ขึ้นมา 1 คำตอบและกำหนดให้เป็น x_j และทำการเปรียบเทียบคุณภาพคำตอบกับ $x_i^{(t+1)}$ หากคำตอบ $x_i^{(t+1)}$ มีค่าความเหมาะสมที่ดีกว่าคำตอบ x_j (หรือ $F_i > F_j$) ให้ทำการแทนที่คำตอบ x_j ด้วยคำตอบ $x_i^{(t+1)}$ มิฉะนั้นให้ทำการเก็บคำตอบ x_j ไว้และละทิ้งคำตอบ $x_i^{(t+1)}$

8. ในกรณีที่วิธีการ CS จะถูกประยุกต์ใช้กลักการค้นหาเฉพาะพื้น (Local Search) ที่สามารถใช้หลักการ Exchange Operator (EO) หรือ Insertion Operator (IO) เพื่อปรับปรุงประสิทธิภาพของวิธีการ CS

9. เป็นการละทิ้งคำตอบ (Abandoned nests) ที่มีคุณภาพไม่ดีด้วยความน่าจะเป็นที่ P_a โดย $P_a \in [0,1]$ [23] พร้อมทั้งทำการสร้างคำตอบใหม่แบบสุ่มมาทดแทนหากคำตอบดังกล่าวถูกละทิ้ง

10. จัดเรียงค่าคำตอบ (Solution ranking) ของวิธีการ CS โดยพิจารณาจากค่าความเหมาะสมมากที่สุดไปน้อยที่สุด จากนั้นทำการกำหนดให้คำตอบในลำดับที่ 1 ซึ่งมีค่าความเหมาะสมมากที่สุดเป็นคำตอบที่ดีที่สุดตั้งแต่เริ่มต้นการประมวลผล (Best so far solution: BSF)

11. ในกรณีที่เงื่อนไขการสิ้นสุดการประมวลผล (เช่น จำนวนรอบการค้นหาคำตอบ เป็นต้น) ยังไม่เป็นจริง ให้วนซ้ำกลับไปที่ขั้นตอนที่ 4 ถึงขั้นตอนที่ 10 อีกครั้งและจะวนซ้ำจนกว่าเงื่อนไขดังกล่าวจะเป็นจริงจึงจะหยุดการทำงาน โดยที่คำตอบ BSF ที่ได้หลังจากการประมวลผลจะเป็นผลเฉลยของปัญหา

3.5 การปรับปรุงประสิทธิภาพวิธีการคุกคูเสิร์ช (CS) ในการแก้ปัญหาการจัดตารางสอน

การปรับปรุงประสิทธิภาพในการค้นหาคำตอบของวิธีการ CS สามารถจำแนกเป็น 3 แนวทางหลัก [176] คือ การกำหนดค่าพารามิเตอร์ (Parameter setting) การปรับปรุงกระบวนการ (Process modification) และวิธีการแบบผสมผสาน (Hybridisation)

1. การปรับปรุงประสิทธิภาพ CS ในด้านการกำหนดค่าพารามิเตอร์ที่เหมาะสมนั้นนอกจากเป็นการลดข้อด้อยของวิธีการในกลุ่มของวิธีการเมต้าฮิวริสติกส์ [22] แล้ว ยังเป็นวิธีการปรับปรุงประสิทธิภาพของวิธีการ CS ที่ไม่ต้องทำการปรับเปลี่ยนหรือเพิ่มกระบวนการใดๆ เข้ามาในโปรแกรมซึ่งอาจส่งผลกระทบต่อเวลาในการประมวณผลที่เพิ่มขึ้นได้ วิธีการกำหนดค่าพารามิเตอร์ให้กับวิธีการเมต้าฮิวริสติกส์มีหลายวิธีการ เช่น วิธีการ Ad hoc selection [43], วิธีการ Adopted approach [176], วิธีการ Best guess approach [44], วิธีการ

One-factor-at-a-time [44], วิธีการ Factorial design [44] เป็นต้น โดยที่วิธีการ Factorial design จะเป็นวิธีการที่ถูกต้องและมีประสิทธิภาพมากสุดสำหรับค้นหาค่าพารามิเตอร์ที่เหมาะสมให้กับวิธีการเมต้าฮิวริสติกส์ เนื่องจากสามารถจัดการกับพารามิเตอร์จำนวนหลายตัวพร้อมกันได้โดยใช้หลักการออกแบบการทดลองและการวิเคราะห์ทางสถิติ (Experimental design and analysis) หรือวิธีการ DOE เข้ามาช่วย อีกทั้งยังสามารถศึกษาผลกระทบร่วม (Interaction) ระหว่างปัจจัยได้ด้วย [44] ซึ่งในงานวิจัยนี้ได้ประยุกต์ใช้วิธีการออกแบบการทดลองและการวิเคราะห์ทางสถิติเข้ามาช่วยในการกำหนดค่าพารามิเตอร์ที่เหมาะสมในแต่ละโจทย์ปัญหา

2. การปรับปรุงประสิทธิภาพของวิธีการเมต้าฮิวริสติกส์ให้ดีขึ้นกว่าเดิมโดยการปรับปรุงกระบวนการ (Modification) ทำงานจัดเป็นอีกวิธีการหนึ่งที่พบได้บ่อย สำหรับปัญหาการจัดตารางเรียนตารางสอนพบว่าหลายงานวิจัยได้ปรับปรุงประสิทธิภาพของวิธีการเมต้าฮิวริสติกส์ในกระบวนการหรือในขั้นตอนที่แตกต่างกันออกไปขึ้นอยู่กับวัตถุประสงค์ของการปรับปรุงในกระบวนการนั้นๆ ซึ่งผลของการปรับปรุงต่างก็ประสบผลสำเร็จเป็นอย่างดี เช่น กระบวนการ Initial solution [46, 47], กระบวนการ Solution evolution [48], กระบวนการ Fitness calculation [19], กระบวนการ Selection process [49], กระบวนการ Solution replacement [10], กระบวนการ Accepted rules [19], กระบวนการ Memory update [50] เป็นต้น ในงานวิจัยนี้จะทำการปรับปรุงประสิทธิภาพ CS ในส่วนของกระบวนการ Solution evolution ซึ่งจัดอยู่ในขั้นตอนการเคลื่อนที่ของ CS โดยประยุกต์ใช้หลักการเคลื่อนที่แบบ Gaussian random walks (CSGRW) และแบบ Lévy flights (CSLF)

3. การปรับปรุงประสิทธิภาพของวิธีการเมต้าฮิวริสติกส์ให้ดีขึ้นกว่าเดิมโดยการผสมผสาน (Hybridisation) จัดเป็นอีกวิธีการหนึ่งที่พบได้บ่อย เนื่องจากการแก้ปัญหาการหาค่าที่เหมาะสมที่สุดหลายประเภทในปัจจุบันนั้น การใช้วิธีการเมต้าฮิวริสติกส์เพียงวิธีการเดียวปกติแล้วจะไม่เหมาะกับการนำไปประยุกต์ใช้แก้ปัญหาที่มีความยากมากๆ เพราะผลลัพธ์ที่ได้อาจจะไม่ดีเท่าที่ควร [22] จากการทบทวนวรรณกรรมที่ผ่านมาเกี่ยวกับการผสมผสานของวิธีการเมต้าฮิวริสติกส์ที่พบในปัญหาการจัดตารางเรียนตารางสอน สามารถจำแนกออกเป็น 3 แบบคือ วิธีการ S-meta ผสมผสานกับ S-meta, วิธีการ P-meta ผสมผสานกับ P-meta และวิธีการ S-meta ผสมผสานกับ P-meta

วิธีการ S-meta ผสมผสานกับ P-meta จะเป็นแนวทางที่ได้รับความนิยมมากที่สุดของการผสมผสานและยังเป็นวิธีการที่มีประสิทธิภาพสูงในการค้นหาคำตอบ [22, 51, 52] เพราะว่า จะเป็นการใช้ข้อดีของการค้นหาคำตอบในวงกว้างซึ่งได้จาก P-meta (Exploration) ร่วมกับการใช้ข้อดีของการค้นหาคำตอบในวงแคบซึ่งได้จาก S-meta (Exploitation) ทำให้วิธีการผสมผสานแบบนี้เกิดความสมดุลกันจากหลักการค้นหาคำตอบทั้ง 2 แบบ (Diversification และ Intensification) [22, 51, 52] ดังนั้นวิธีการผสมผสานในกลุ่มนี้จึงได้รับความนิยมนำมาใช้แก้ปัญหาต่างๆ และประสบความสำเร็จเป็นอย่างดี ซึ่งรวมไปถึงปัญหาการจัดตารางการศึกษาด้วย ตัวอย่างเช่น วิธีการ Ant Colony System (ACS) กับ TS [53], วิธีการ GA+LS (Memetic Algorithm: MA) [54], วิธีการ GA+TS [12], วิธีการ PSO+LS [48, 55], วิธีการ Best-worst ACS กับ LS [13] เป็นต้น

ในงานวิจัยนี้จะทำการปรับปรุงประสิทธิภาพ CS ในด้านการแบบผสมผสาน (Hybridisation) แบบ S-meta ผสมผสานกับ P-meta โดยการประยุกต์ใช้วิธีการค้นหาคำตอบเฉพาะพื้นที่ (Local Search: LS) แบบ

Insertion Operator (IO) และแบบ Exchange Operator (EO) (เรียกว่าวิธีการ CSLF+IO และวิธีการ CSLF+EO) เข้ามาทำการผสมผสาน

3.6 ทำการออกแบบรูปแบบของข้อมูลนำเข้า

การออกแบบรูปแบบของข้อมูลปัญหาสำหรับนำเข้า (Input data) ให้กับโปรแกรมจัดตารางเรียนตารางสอนแบบอัตโนมัติ ซึ่งเป็นอีกขั้นตอนของการมอบหมายทรัพยากรทางการศึกษาหลังจากที่ได้ทำการเก็บรวบรวมข้อมูลที่เกี่ยวข้องแล้ว โดยข้อมูลการจัดตารางสอนของปีการศึกษา 2555 ได้ถูกนำมาออกแบบและสร้างเป็นโจทย์ปัญหา (Datasets) ซึ่งในแต่ละโจทย์ปัญหาจะเก็บข้อมูลเป็นแบบตัวอักษรและตัวเลข (Alphabet and number) แบบเว้นบรรทัดและจะถูกเก็บไว้ในไฟล์นามสกุล .text (Text file) สำหรับโครงสร้างของข้อมูลในแต่ละไฟล์จะถูกกำหนดออกเป็น 6 ส่วนหลัก ประกอบไปด้วย รายละเอียดของปัญหา รายละเอียดของวิชาเรียนวิชาสอน รายละเอียดของห้องเรียน รายละเอียดข้อบังคับของนิสิต รายละเอียดข้อบังคับของอาจารย์ และรายละเอียดของค่าตอบแทนของอาจารย์แสดงดังภาพ 3

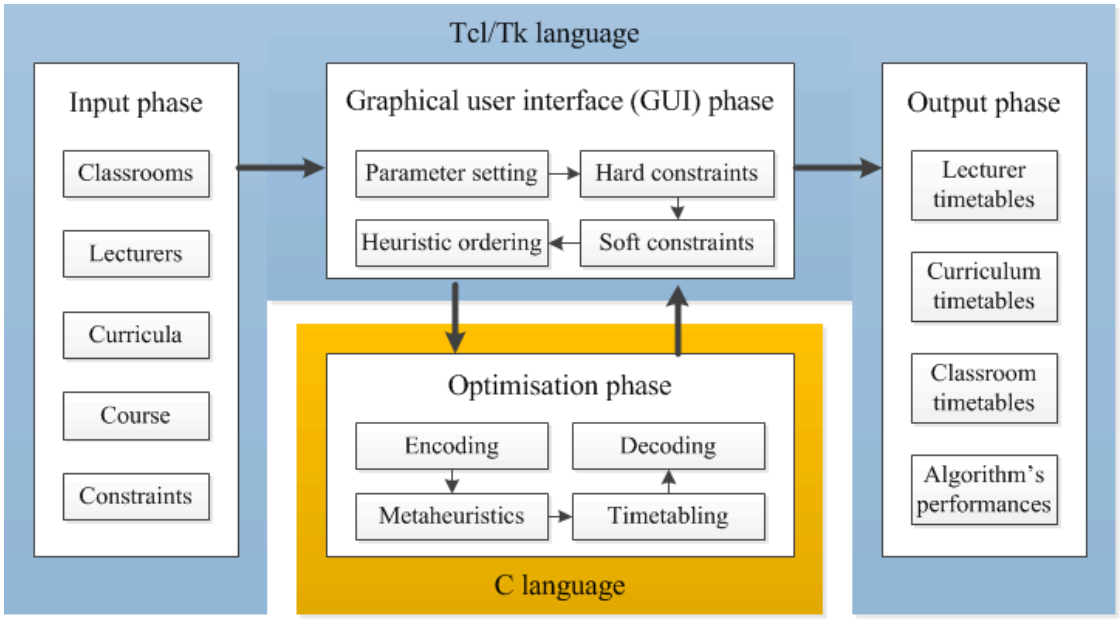
```
DATASET_NAME:                                   Toy
SET_OF_DEGREES:                                 {13  23  33}
NUMBER_OF_COURSES:                              7
NUMBER_OF_CLASSROOMS:                           2
NUMBER_OF_DAYS/WEEK:                            5
NUMBER_OF_PERIODS/DAY:                          5
NUMBER_OF_CURRICULA_UNAVAI_PERIOD_LINE:         15
NUMBER_OF_LECTURER_UNAVAI_PERIOD_LINE:          2
NUMBER_OF_LECTURER_COSTS_LINE:                  4

COURSES
13   302151    1_(lec) 3   0   81   IE          0       B       0       0               G03005          13_0701_1
13   302151    1_(lab) 2   1   81   0           EN_312  0       0       0               G03005          13_0701_1
23   302502    1       4   0   10   IE          0       0       {2 4}   {{1 2} {1 2}}   G03021          23_0270_1
33   303622    1_(lec) 2   1   5    {IE EN}     0       0       0       0               {G03020 G04022} 33_60118_1
33   303622    1_(lab) 2   1   5    EN          0       N       0       0               {G03020 G04022} 33_60118_1
13   302155    1       2   1   50   {IE EN}     0       0       0       0               G03005          13_0701_1
23   302156    1       2   1   50   {IE EN}     0       0       0       0               G03021          23_0270_1

CLASSROOMS
EN   EN_312   50   N    1.0   1.0   {1 1 1 1 1.5}   {1 1 1 1 1.5}   {1 1 1 1 1.5}   {1 1 1 1 1.5}   {1 1 1 1 1.5}
IE   IE_509   80   B    1.2   1.2   {1 1 1 1 1.5}   {1 1 1 1 1.5}   {1 1 1 1 1.5}   {1 1 1 1 1.5}   {1 1 1 1 1.5}

CURRICULA_UNAVAILABLE_PERIODS
13      1       {3}
13      2       {3}
13      3       {3}
13      4       {3}
13      5       {3}
23      1       {3}
23      2       {3}
23      3       {3}
23      4       {3}
23      5       {3}
33      1       {3}
33      2       {3}
33      3       {3}
33      4       {3}
33      5       {3}

LECTURER_UNAVAILABLE_PERIODS
G03020  1       {1 2}
G03020  3       {3 4}

HIRED_LECTURER_COSTS
G03005  1.4     1.7     2.5     {1 1 1 1 1.5}   {1 1 1 1 1.5}   {1 1 1 1 1.5}   {1 1 1 1 1.5}   {1 1 1 1 1.5}
G03021  1.0     1.2     1.8     {1 1 1 1 1.5}   {1 1 1 1 1.5}   {1 1 1 1 1.5}   {1 1 1 1 1.5}   {1 1 1 1 1.5}
G03020  1.8     2.2     3.2     {1 1 1 1 1.5}   {1 1 1 1 1.5}   {1 1 1 1 1.5}   {1 1 1 1 1.5}   {1 1 1 1 1.5}
G04022  2.4     2.9     4.3     {1 1 1 1 1.5}   {1 1 1 1 1.5}   {1 1 1 1 1.5}   {1 1 1 1 1.5}   {1 1 1 1 1.5}

END.
```

ภาพ 3 ตัวอย่างโครงสร้างไฟล์ข้อมูลนำเข้าสำหรับโปรแกรมจัดตารางเรียนตารางสอน

3.7 ทำการออกแบบโปรแกรมจัดตารางสอนแบบอัตโนมัติ

ในงานวิจัยนี้แนวคิดสำหรับการพัฒนาโปรแกรมจัดตารางเรียนตารางสอนแบบอัตโนมัติสามารถแสดงได้ดังภาพ 4 โดยจะประกอบไปด้วย 4 ส่วนหลักคือ ส่วนการนำเข้าข้อมูล (Input phase) ส่วนการใช้ภาพเป็นตัวประสานกับผู้ใช้ (GUI phase) ส่วนการจัดตารางที่เหมาะสมที่สุด (Optimisation phase) และส่วนการนำออกข้อมูล (Output phase)

- ส่วนการนำเข้าข้อมูล (Input phase) เป็นการนำข้อมูลเกี่ยวกับการจัดตารางทั้งหมด เช่น ห้องเรียน อาจารย์ นิสิต วิชาเรียนวิชาสอน ข้อบังคับต่างๆ เป็นต้น จากที่ได้เก็บไว้ในไฟล์ .text เข้าสู่โปรแกรมโดยผ่าน GUI
- ส่วนการใช้ภาพเป็นตัวประสานกับผู้ใช้ (GUI phase) เป็นการติดต่อสื่อสารในการรับ/ส่งค่าตัวแปรระหว่างผู้ใช้โปรแกรมและตัวโปรแกรม เช่น การกำหนดค่าพารามิเตอร์ การกำหนดข้อบังคับหลักและข้อบังคับรอง การกำหนดการจัดลำดับรายวิชา เป็นต้น ก่อนที่จะส่งข้อมูลไปยังขั้นตอนการจัดตารางที่เหมาะสมที่สุด
- ส่วนการจัดตารางที่เหมาะสมที่สุด (Optimisation phase) เป็นการนำข้อมูลที่ได้รับมาทำการจัดตารางเรียนตารางสอนที่ดีที่สุด โดยเริ่มจากการเข้ารหัสข้อมูลก่อนที่จะใช้วิธีการ Cuckoo search (CS) วิธีการ Firefly algorithm (FA) วิธีการ Hybrid แบบต่างๆ ทำการจัดตารางเรียนตารางสอน เมื่อได้คำตอบแล้วก็จะทำการถอดรหัสก่อนที่จะส่งผลไปยัง GUI อีกครั้ง
- ส่วนการนำออกข้อมูล (Output phase) เป็นการนำผลลัพธ์ที่ได้มาทำการแสดงเป็นตารางเรียนตารางสอนและตารางการใช้ห้องเรียน รวมถึงแสดงประสิทธิภาพที่ได้ของวิธีการที่ใช้ในการจัดตารางด้วย



ภาพ 4 แสดงแนวคิดสำหรับการพัฒนาโปรแกรมจัดตารางเรียนตารางสอนแบบอัตโนมัติ [177]

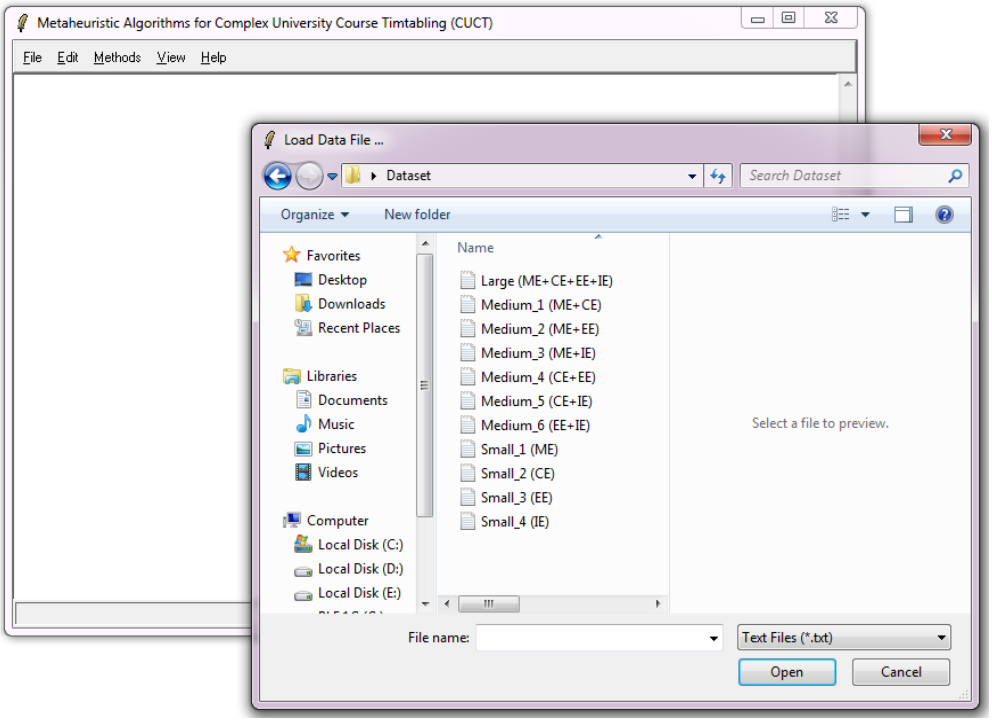
นอกจากนี้ในส่วนของการพัฒนาโปรแกรมจะใช้ภาษา Tcl/Tk และ C ร่วมกันแบบ Extension [178] โดยใช้ภาษา Tcl/Tk ทำการพัฒนาในส่วนของการนำเข้าข้อมูล (Input phase) การใช้ภาพเป็นตัวประสานกับ

ผู้ใช้ (GUI phase) และส่วนการนำออกข้อมูล (Output phase) เพราะว่าการใช้ Tcl Script เป็นคำสั่งที่สั้นง่าย ทำให้สามารถพัฒนาโปรแกรมได้เร็ว โดยเฉพาะอย่างยิ่งการสร้าง GUI ต่างๆ ในขณะที่ในส่วนของส่วนการจัดตารางที่เหมาะสมที่สุด (Optimisation phase) จะใช้ภาษา C ในการพัฒนาโปรแกรมเนื่องจากเป็นส่วนที่ต้องการความเร็วในการประมวลผลที่สูงมาก

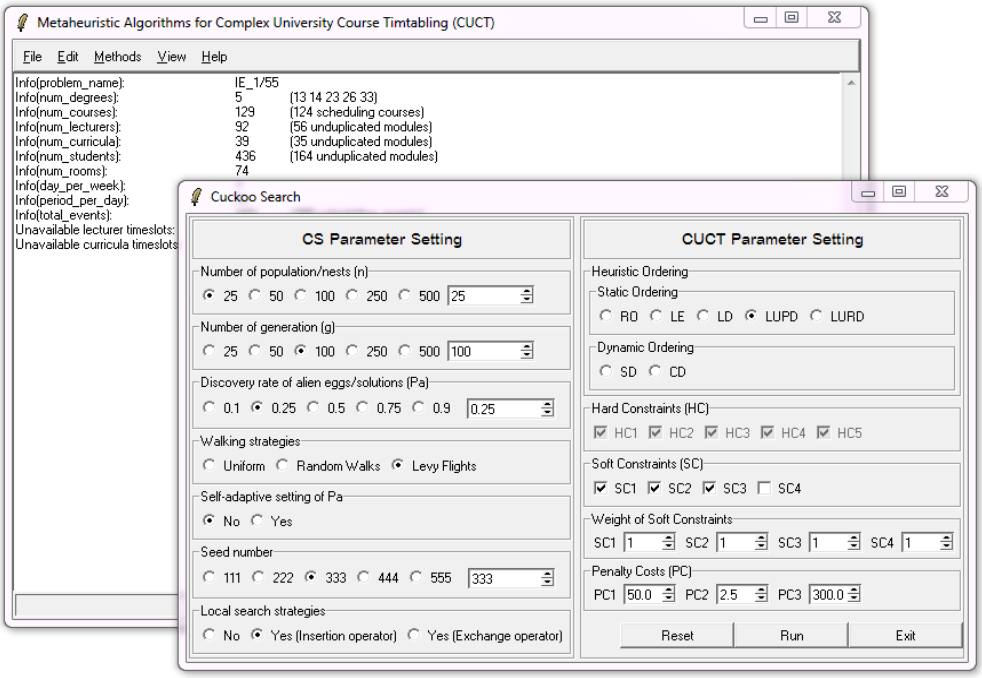
3.8 ทำการพัฒนาโปรแกรมจัดตารางสอนตามที่ได้ออกแบบไว้โดยใช้ภาษา TCL/TK และ C

โปรแกรมการจัดตารางเรียนตารางสอนที่ถูกพัฒนาขึ้น โดยจะประกอบไปด้วย 3 ส่วนหลักคือ ส่วนของข้อมูลนำเข้าของโปรแกรม ส่วนของการประมวลผลของโปรแกรม และส่วนของข้อมูลนำออกของโปรแกรม

1. ส่วนของข้อมูลนำเข้าของโปรแกรม ในส่วนนี้จะเป็นการนำเข้าของข้อมูลปัญหาการจัดตารางเรียนตารางสอนที่ได้ถูกสร้างไว้แล้ว โดยแต่ละโจทย์จะถูกกำหนดในรูปแบบของไฟล์ .text ซึ่งผู้ใช้โปรแกรมสามารถเลือกได้ว่าจะให้โปรแกรมใช้โจทย์ใดทำการประมวลผลโดยผ่านส่วนติดต่อกับผู้ใช้ (Graphical user interface: GUI) ที่ถูกพัฒนาขึ้นด้วยภาษา TCL/TK ดังภาพ 5 หลังจากนำเข้าข้อมูลปัญหาเรียบร้อยแล้ว ผู้ใช้โปรแกรมสามารถเลือกวิธีการที่จะใช้ในการจัดตารางเรียนตารางสอนให้กับโปรแกรมได้ซึ่งประกอบด้วยวิธีการ CS และวิธีการผสมผสานแบบต่างๆ ซึ่งในแต่ละวิธีการที่เลือกจะมีหน้าต่างสำหรับทำการกำหนดค่าพารามิเตอร์ทั้งในส่วนของปัญหาและวิธีการ ผู้ใช้งานสามารถกำหนดได้อย่างอิสระและโปรแกรมจะทำการกำหนดค่าเริ่มต้นไว้ด้วยสำหรับผู้งานทั่วไปดังภาพ 6

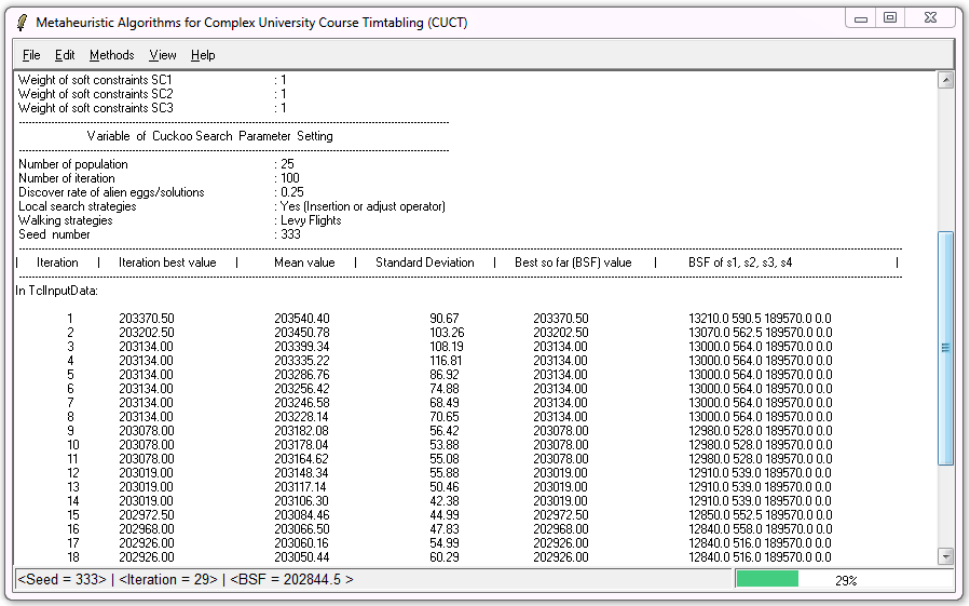


ภาพ 5 ตัวอย่างการนำเข้าข้อมูลของโปรแกรมจัดตารางเรียนตารางสอนที่ถูกพัฒนาขึ้น



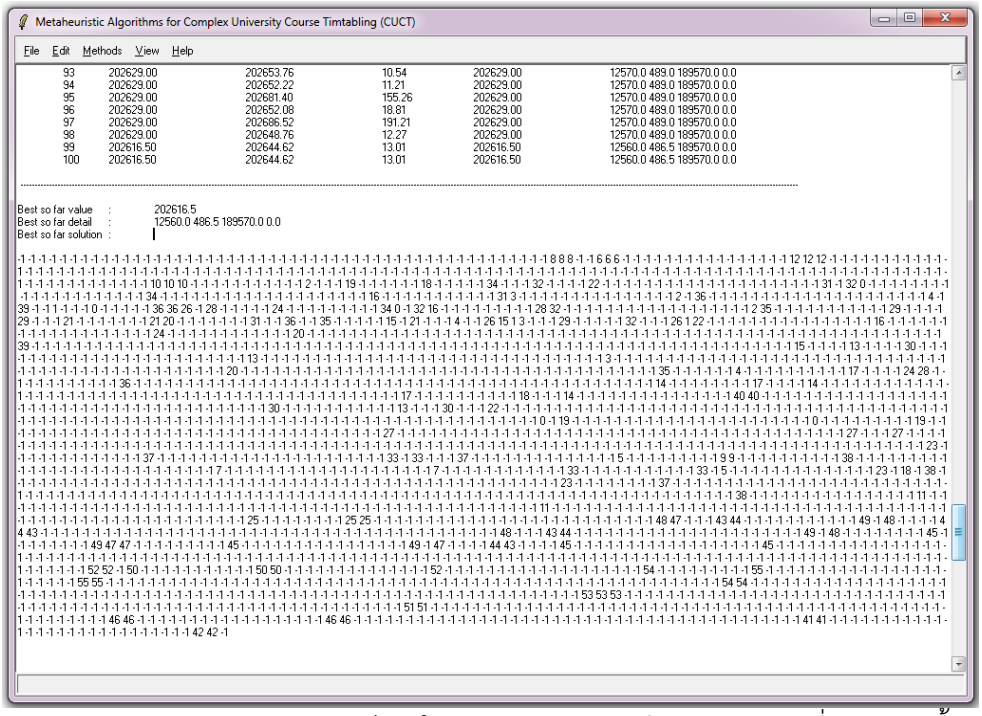
ภาพ 6 ตัวอย่างการกำหนดค่าพารามิเตอร์ให้กับโปรแกรมที่ถูกพัฒนาขึ้น

2. ส่วนของการประมวลผลของโปรแกรม หลังจากที่ผู้ใช้ทำการนำเข้าข้อมูลและเลือกวิธีการพร้อมทั้งกำหนดค่าพารามิเตอร์เรียบร้อยแล้ว ผู้ใช้สามารถทำการกดรันโปรแกรมได้โดยที่โปรแกรมจะแสดงข้อมูลระหว่างการประมวลผล ซึ่งประกอบด้วย รอบการประมวลผลปัจจุบัน คำตอบที่ดีที่สุดในแต่ละรอบ ค่าเฉลี่ยของคำตอบในแต่ละรอบ ส่วนเบี่ยงเบนมาตรฐานในแต่ละรอบ คำตอบที่ดีที่สุดตั้งแต่เริ่มการประมวลผล และต้นทุนที่เกิดจากข้อบังคับรองแต่ละข้อดังภาพ 7



ภาพ 7 ตัวอย่างการประมวลผลของโปรแกรมจัดตารางเรียนตารางสอนที่ถูกพัฒนาขึ้น

หลังจากเสร็จสิ้นการประมวลผลแล้วโปรแกรมจะแสดงคำตอบและค่าของคำตอบที่ดีที่สุดตั้งแต่เริ่มต้นการประมวลผล (Best so far solution and value) ออกมาดังภาพ 8 ซึ่งจะเห็นได้ว่าลักษณะของคำตอบที่ดีที่สุดที่แสดงออกมานั้นจะอยู่ในรูปของตัวเลขที่ยังเข้ารหัส (Encoded) อยู่



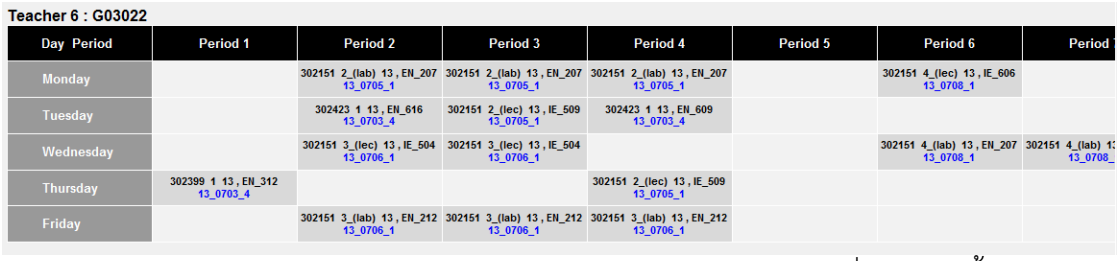
ภาพ 8 ตัวอย่างการแสดงผลลัพธ์ของโปรแกรมจัดตารางเรียนตารางสอนที่ถูกพัฒนาขึ้น

3. <u>ส่วนของข้อมูลนำออกของโปรแกรม</u> ในส่วนนี้จะเป็นการนำคำตอบที่ดีที่สุดที่แสดงออกมาหลังจากการประมวลผลมาทำการถอดรหัส (Decoded) ให้อยู่ในรูปแบบของตารางการใช้ห้องเรียนดังภาพภาพ 9 ตารางสอนของอาจารย์ดังภาพ 10 และตารางเรียนของนักเรียนดังภาพ 11 ซึ่งทำให้ผู้ใช้โปรแกรมสามารถเข้าใจได้ง่ายและสามารถนำผลดังกล่าวไปใช้งานได้

Room 2 : EN_207

Day Period	Period 1	Period 2	Period 3	Period 4	Period 5	Period 6	Period 7
Monday		302151 2_(lab) 13 , G03022 13_0705_1	302151 2_(lab) 13 , G03022 13_0705_1	302151 2_(lab) 13 , G03022 13_0705_1			302151 1_(lab) 13 , G 13_0701_1
Tuesday							
Wednesday						302151 4_(lab) 13 , G03022 13_0708_1	302151 4_(lab) 13 , G 13_0708_1
Thursday							
Friday							

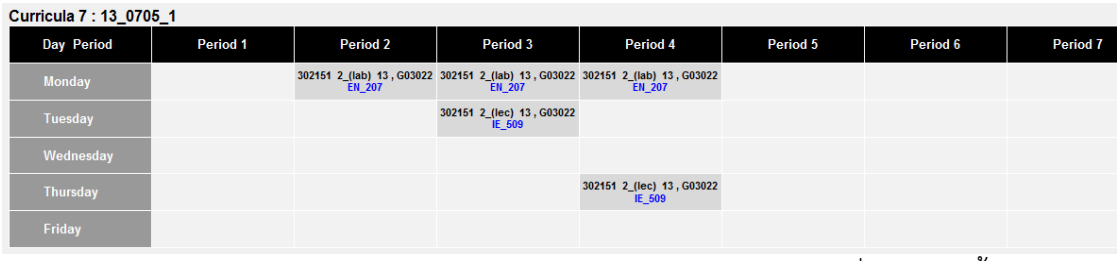
ภาพ 9 ตัวอย่างแสดงตารางการใช้ห้องเรียนของโปรแกรมที่ถูกพัฒนาขึ้น



Teacher 6 : G03022

Day Period	Period 1	Period 2	Period 3	Period 4	Period 5	Period 6	Period
Monday		302151 2_(lab) 13 , EN_207 13_0705_1	302151 2_(lab) 13 , EN_207 13_0705_1	302151 2_(lab) 13 , EN_207 13_0705_1		302151 4_(lec) 13 , IE_606 13_0708_1	
Tuesday		302423 1 13 , EN_616 13_0703_4	302151 2_(lec) 13 , IE_509 13_0705_1	302423 1 13 , EN_609 13_0703_4			
Wednesday		302151 3_(lec) 13 , IE_504 13_0706_1	302151 3_(lec) 13 , IE_504 13_0706_1			302151 4_(lab) 13 , EN_207 13_0708_1	302151 4_(lab) 1 13_0708_
Thursday	302399 1 13 , EN_312 13_0703_4			302151 2_(lec) 13 , IE_509 13_0705_1			
Friday		302151 3_(lab) 13 , EN_212 13_0706_1	302151 3_(lab) 13 , EN_212 13_0706_1	302151 3_(lab) 13 , EN_212 13_0706_1			

ภาพ 10 ตัวอย่างแสดงตารางสอนอาจารย์ของโปรแกรมที่ถูกพัฒนาขึ้น



Curricula 7 : 13_0705_1

Day Period	Period 1	Period 2	Period 3	Period 4	Period 5	Period 6	Period 7
Monday		302151 2_(lab) 13 , G03022 EN_207	302151 2_(lab) 13 , G03022 EN_207	302151 2_(lab) 13 , G03022 EN_207			
Tuesday			302151 2_(lec) 13 , G03022 IE_509				
Wednesday							
Thursday				302151 2_(lec) 13 , G03022 IE_509			
Friday							

ภาพ 11 ตัวอย่างแสดงตารางเรียนนักเรียนของโปรแกรมที่ถูกพัฒนาขึ้น

3.9 การออกแบบการทดลองและการทดสอบโปรแกรม

หลังจากที่ได้ทำการพัฒนาโปรแกรมจัดตารางเรียนตารางสอนเสร็จสิ้นและได้ทำการตรวจสอบความถูกต้องแล้ว ขั้นตอนถัดไปจะเป็นการทดสอบโปรแกรมตามวัตถุประสงค์ที่ได้กำหนดไว้ เนื่องจากการกำหนดค่าพารามิเตอร์ที่เหมาะสมสำหรับวิธีการเมต้าฮิวริสติกส์นั้นมีความสำคัญเป็นอย่างมากต่อประสิทธิภาพของวิธีการดังกล่าว นอกจากนี้ค่าพารามิเตอร์ที่เหมาะสมของแต่ละวิธีการจะถูกนำไปใช้ในหลายวัตถุประสงค์ถัดไปอีกด้วย ดังนั้นการออกแบบการทดลองจึงเป็นสิ่งที่จำเป็นอย่างยิ่งในการสืบทราบค่าพารามิเตอร์ที่เหมาะสมสำหรับวิธีการ CS ก่อนทำการศึกษาเปรียบเทียบในการทดลองอื่นๆ ต่อไป

จำนวนพารามิเตอร์ของวิธีการ CS จะประกอบด้วย 2 ค่า [23] คือ จำนวนการค้นหาคำตอบ ซึ่งเกิดจากจำนวนประชากร (Population) คูณกับจำนวนรอบทั้งหมด (Iteration) ที่ใช้ในการค้นหา(หรือ *PI*) และค่าความน่าจะเป็นในการละทิ้งคำตอบ (P_a) เนื่องจากชุดของพารามิเตอร์ที่เหมาะสมในแต่ละปัญหาและแต่ละวิธีการย่อมต่างกันออกไป [22] ดังนั้นในงานวิจัยนี้จึงต้องทำการหาค่าพารามิเตอร์ที่เหมาะสมที่สุดของวิธีการ CS ทั้งหมด 11 โจทย์ปัญหาที่ได้เสนอขึ้นมา

สำหรับวิธีการ CS เนื่องจากมีจำนวนพารามิเตอร์ที่น้อย จึงได้เลือกใช้การออกแบบการทดลองเชิงแฟกทอเรียลแบบสมบูรณ์ 3^2 (Full factorial experimental design) [44] เพื่อค้นหาค่าพารามิเตอร์ที่เหมาะสมทั้งหมด 11 โจทย์ ซึ่งแต่ละโจทย์จะมีจำนวนครั้งของการทดลอง 9 ครั้งต่อ 1 ค่าของการสุ่ม (Random seed) และในแต่ละโจทย์จะถูกทดลองซ้ำ 30 ครั้ง โดยใช้หมายเลขของการสุ่มที่แตกต่างกันออกไป ดังนั้นในการทดลองนี้จะมีจำนวนรันทั้งหมดเท่ากับ 2,970 (11*9*30) รัน โดยถูกทดสอบบนเครื่องคอมพิวเตอร์ Intel Core i7 ที่ความถี่ 3.4 GHz และมีหน่วยความจำ 4 GB (DDR3) ซึ่งผลการทดลองที่ได้ทั้งหมดที่ได้จะถูกนำมาวิเคราะห์ผลการทดลองต่อไป

3.10 การวิเคราะห์ผลจากการออกแบบการทดลอง

สำหรับการทดลองเพื่อค้นหาค่าพารามิเตอร์ที่เหมาะสมสำหรับวิธีการ CS นั้น ผลการทดลองที่ได้จากการรันตามแผนการทดลองทั้งหมดในแต่ละโจทย์ปัญหา จะถูกนำมาวิเคราะห์ความแปรปรวน (ANOVA) ในรูปแบบเชิงเส้นแบบทั่วไป (General linear model) ซึ่งประกอบไปด้วย ผลรวมกำลังสอง (Sum of square: *SS*) ระดับความอิสระ (Degree of freedom: *DF*) ค่าเฉลี่ยกำลังสอง (Mean square: *MS*) ค่า F (*F* value) และ ค่า P (*P* value) ด้วยโปรแกรมประยุกต์ทางด้านสถิติ Minitab เวอร์ชัน 14 เพื่ออธิบายถึงผลกระทบของปัจจัยหลัก (Main effect) และผลกระทบร่วม (Interaction) ซึ่งจะนำไปสู่ข้อสรุปได้ว่า ปัจจัยหรือพารามิเตอร์ใดบ้างที่จะมีผลกระทบต่อการทดลองโดยนัยสำคัญ (Significance) ทางสถิติ หลังจากนั้นจะทำการสรุปช่วงหรือค่าของพารามิเตอร์ที่เหมาะสมสำหรับวิธีการ CS โดยใช้ผลจากภาพการวิเคราะห์ผลกระทบของปัจจัยหลัก (Main effect plot) ควบคู่ไปกับผลจากการเปรียบเทียบหลายระดับเป็นเครื่องมือชี้วัด

สำหรับการเปรียบเทียบหลายระดับ (Multiple comparison) จะถูกนำมาใช้ในการค้นหาว่าระดับของปัจจัยคู่ใดที่แตกต่างไปจากระดับอื่นอย่างมีนัยสำคัญทางสถิติด้วยโปรแกรมประยุกต์ทางด้านสถิติ Minitab เวอร์ชัน 14 ซึ่งปัจจัยที่จะนำมาทำการเปรียบเทียบหลายระดับได้นั้นควรจะเป็นปัจจัยที่ได้ทำการวิเคราะห์ ANOVA แล้วพบว่ามีแตกต่างที่ระดับความเชื่อมั่นที่ 95 เปอร์เซ็นต์ เพียงแต่ยังไม่ทราบว่าเป็นระดับของปัจจัยคู่ใดที่แตกต่างจากระดับอื่น ซึ่งวิธีการเปรียบเทียบหลายระดับที่ใช้ในงานวิจัยนี้คือ วิธีการ Tukey [44] เนื่องจากวิธีการนี้เป็นที่นิยมใช้และเป็นวิธีการที่สามารถเชื่อมั่นได้ว่าค่า α ของทุกคู่ที่เปรียบเทียบจะยังคงเป็น 0.05 เท่าเดิม

นอกจากนี้ในส่วนของการวิเคราะห์ผลการทดลองอื่นที่เกี่ยวข้องกับการเปรียบเทียบประสิทธิภาพของวิธีการต่างๆ จะมีการใช้หลักการทางสถิติ เช่น ค่าน้อยที่สุด (Minimum) ค่ามากที่สุด (Maximum) การหาค่าเฉลี่ย (Average) การหาค่าเบี่ยงเบนมาตรฐาน (SD) การวิเคราะห์แบบจับคู่ (Pairwise comparisons) เป็นต้น เข้ามาช่วยในการสรุปผลการทดลองให้มีความน่าเชื่อถือมากยิ่งขึ้น

บทที่ 4

ผลการวิจัย

หลังจากที่ได้ทำการพัฒนาโปรแกรมจัดตารางเรียนตารางสอนเสร็จสิ้นและได้ทำการตรวจสอบความถูกต้องแล้ว ขั้นตอนถัดไปจะเป็นการทดสอบโปรแกรมตามวัตถุประสงค์ที่ได้กำหนดไว้ เนื่องจากการกำหนดค่าพารามิเตอร์ที่เหมาะสมสำหรับวิธีการเมต้าฮิวริสติกส์นั้นมีความสำคัญเป็นอย่างมากต่อประสิทธิภาพของวิธีการดังกล่าว นอกจากนี้ค่าพารามิเตอร์ที่เหมาะสมของแต่ละวิธีการจะถูกนำไปใช้ในหลายวัตถุประสงค์ถัดไปอีกด้วย ดังนั้นในการทดลองที่ 1 จึงจะทำการออกแบบและวิเคราะห์ผลการทดลองในการสืบทราบค่าพารามิเตอร์ที่เหมาะสมสำหรับวิธีการ CS ก่อน จากนั้นในการทดลองที่ 2 จะทำการศึกษาเปรียบเทียบประสิทธิภาพของวิธีการ CS ที่มีการปรับเปลี่ยนกระบวนการ (Modification) และในการทดลองที่ 3 จะทำการศึกษาเปรียบเทียบประสิทธิภาพของวิธีการ CS แบบที่มีการผสมผสาน (Hybridisation) กับวิธีการค้นหาแบบเฉพาะพื้นที่

4.1 ผลการทดลองที่ 1: การค้นหาค่าพารามิเตอร์ที่เหมาะสม

การค้นหาค่าพารามิเตอร์ที่เหมาะสมสำหรับวิธีการ CS เนื่องจากจำนวนพารามิเตอร์ของวิธีการ CS จะประกอบด้วย 2 ค่า คือ จำนวนการค้นหาคำตอบ ซึ่งเกิดจากจำนวนประชากร (Population) คูณกับจำนวนรอบทั้งหมด (Iteration) ที่ใช้ในการค้นหา (หรือ *PI*) และค่าความน่าจะเป็นในการละทิ้งคำตอบ (P_a) ดังนั้นการทดลองนี้ได้ทำการออกแบบการทดลองเชิงแฟกทอเรียลแบบสมบูรณ์ 3^2 (Full factorial experimental design) [44] เพื่อค้นหาค่าพารามิเตอร์ที่เหมาะสมสำหรับวิธีการ CS ในการแก้ปัญหาการจัดตารางเรียนตารางสอนที่ได้สร้างไว้แล้วทั้งหมด 11 โจทย์แสดงดังตาราง 2

ตาราง 2 แสดงข้อมูลโจทย์ปัญหาการจัดตารางเรียนตารางสอนที่ถูกเสนอขึ้นของมหาวิทยาลัยนเรศวร

Problems	Characteristics of the NU course timetabling problems						
	No. Courses	No. Events	No. Classrooms	No. Days/ week	No. Periods/ day	No. Lecturers	No. Curricula
1	56	173	53	5	10	30	19
2	103	323	77	7	10	62	36
3	123	353	86	7	10	49	27
4	124	380	74	7	11	56	35
5	144	452	91	7	10	78	43
6	162	486	99	7	10	71	34
7	163	499	88	7	11	72	38
8	204	639	114	7	10	96	52
9	208	647	99	7	11	102	56
10	221	687	108	7	12	94	44
11	323	1,009	142	7	13	143	66

ในแต่ละโจทย์ปัญหานั้น ได้กำหนดระดับค่าพารามิเตอร์ในแต่ละปัจจัยของวิธีการ CS ออกเป็น 3 ระดับ คือ ระดับต่ำ (-1) ระดับกลาง (0) และระดับสูง (1) ดังตาราง 3 โดยจำนวนประชากร (*P*) ที่เหมาะสมสำหรับปัญหาการหาค่าที่เหมาะสมที่สุดได้ถูกทดลองและถูกแนะนำไว้ที่ 15 ถึง 40 ประชากร [23] และเนื่องจากโปรแกรมใช้เวลาในการประมวลผลที่ไม่นานมาก งานวิจัยนี้จึงได้กำหนดจำนวนการค้นหาทั้งหมด (*PI*) ไว้ที่ 24,000 คำตอบ ในขณะที่ค่าความน่าจะเป็นในการละทิ้งคำตอบ (P_a) ที่เหมาะสมสำหรับปัญหาการหาค่าที่เหมาะสมที่สุดได้ถูกทดลองและแนะนำไว้ที่ 0.25 [23] ดังนั้นงานวิจัยนี้จึงได้กำหนดค่าดังกล่าวไว้ที่ระดับกลาง ขณะที่ค่าระดับต่ำและสูงจะกำหนดค่าไว้ที่ ±0.15

ตาราง 3 แสดงการกำหนดค่าพารามิเตอร์ของวิธีการ CS ในแต่ละปัจจัย

Factors	Levels	Values		
		Low (-1)	Medium (0)	High (1)
Number of population * Number of iterations (*PI*)	3	15*1600	25*960	40*600
Probability of abandon (P_a)	3	0.1	0.25	0.4

ในแต่ละโจทย์จะมีจำนวนวิธีปฏิบัติในการทดลอง 9 (3^2) วิธีปฏิบัติต่อ 1 ค่าของการสุ่ม (Random seed) และแต่ละโจทย์จะถูกทดลองซ้ำ 30 ครั้งโดยใช้หมายเลขของการสุ่มที่แตกต่างกัน เนื่องจากมีจำนวนโจทย์ปัญหาทั้งหมด 11 โจทย์ ดังนั้นการทดลองนี้จะมีจำนวนรันทั้งหมดเท่ากับ 2,970 (11*9*30) รัน นอกจากนี้ในการทดลองนี้ได้กำหนดวิธีการจัดลำดับรายวิชาแบบ LUPD และการประมวลผลการทดลองทั้งหมดได้ถูกทดสอบบนเครื่องคอมพิวเตอร์ Intel Core i7 ที่ความถี่ 3.4 GHz และมีหน่วยความจำ 4 GB (DDR3)

ผลการทดลองที่ได้จากการประมวลผลทั้งหมดจะถูกนำมาวิเคราะห์ความแปรปรวน (ANOVA) ในรูปแบบเชิงเส้นแบบทั่วไป (General linear model) ซึ่งประกอบไปด้วย ผลรวมกำลังสอง (*SS*) ระดับความอิสระ (*DF*) ค่าเฉลี่ยกำลังสอง (MS) ค่า *F* (*F* value) และ ค่า *P* (*P* value) ด้วยโปรแกรมประยุกต์ทางด้านสถิติ Minitab เวอร์ชัน 14 เพื่ออธิบายถึงผลกระทบของปัจจัยหลัก (Main effect) และผลกระทบร่วม (Interaction) ว่าปัจจัยหรือพารามิเตอร์ใดบ้างที่จะมีผลกระทบต่อการทดลองโดยนัยสำคัญ (Significance) ทางสถิติดังตาราง 4

ผลการวิเคราะห์ความแปรปรวนของวิธีการ CS สำหรับโจทย์ปัญหา 3, 4, 5, 7, และ 10 จากตาราง 4 พบว่าผลกระทบของปัจจัยหลัก (Main effect) และผลกระทบของปัจจัยร่วม (Interaction) จาก 2 ปัจจัย ประกอบด้วย จำนวนการค้นหาคำตอบทั้งหมด (*PI*) และค่าความน่าจะเป็นในการละทิ้งคำตอบ (P_a) ไม่มีนัยสำคัญทางสถิติที่ระดับความเชื่อมั่น 95% โดยพิจารณาได้จากค่า *P* ที่ได้มากกว่า 0.05 นอกจากนี้ยังพบว่าหมายเลขของการสุ่ม (*Seeds*) นั้นไม่มีนัยสำคัญทางสถิติที่ระดับความเชื่อมั่น 95% อีกด้วย

ในขณะที่ผลการวิเคราะห์ความแปรปรวนของวิธีการ CS สำหรับโจทย์ปัญหา 1, 2, 6, 8, 9 และ 11 จากตาราง 4 พบว่า ผลกระทบของปัจจัยหลักของจำนวนการค้นหาคำตอบทั้งหมด (*PI*) มีนัยสำคัญทางสถิติที่ระดับความเชื่อมั่น 95% โดยพิจารณาได้จากค่า *P* ที่ได้จะน้อยกว่าหรือเท่ากับ 0.05 ขณะที่ผลกระทบของ

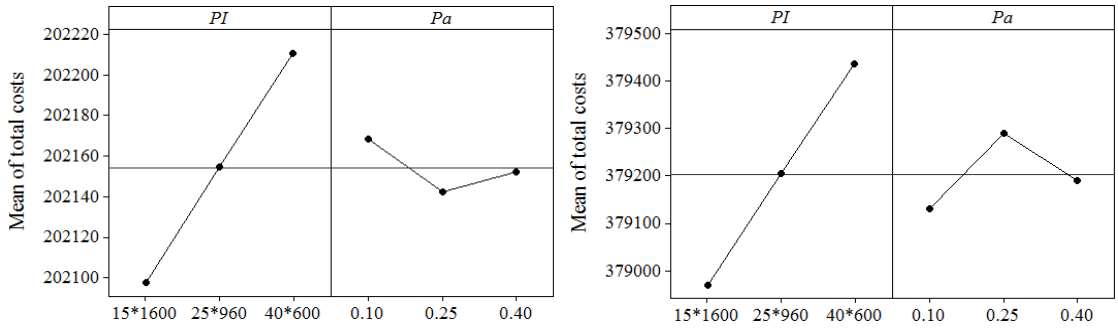
ปัจจัยร่วมนั้นไม่พบว่ามีนัยสำคัญทางสถิติที่ระดับความเชื่อมั่น 95% นอกจากนี้ยังพบว่าหมายเลขของการสุ่มเฉพาะในโจทย์ปัญหา 6 มีนัยสำคัญทางสถิติที่ระดับความเชื่อมั่น 95%

ตาราง 4 แสดงผลการวิเคราะห์ความแปรปรวนของวิธีการ CS สำหรับ 11 โจทย์

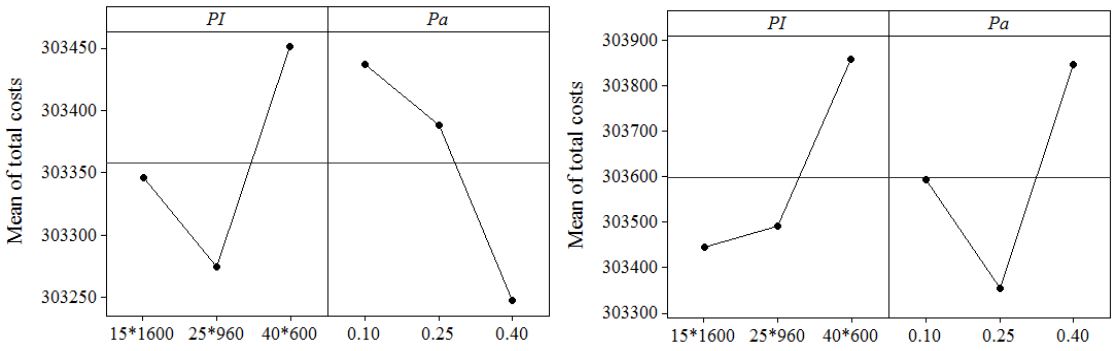
Problems	*Source*	*DF*	*SS*	*MS*	*F*	*P*
1	*PI*	2	576,316	288,158	43.89	**0.000**
	Pa	2	30,338	15,169	2.31	0.102
	*PI*Pa*	4	32,133	8,033	1.22	0.302
	Seeds	29	247,749	8,543	1.30	0.148
	Error	232	1,523,225	6,566		
	Total	269	2,409,761			
2	*PI*	2	9,728,333	4,864,167	23.34	**0.000**
	Pa	2	1,170,584	585,292	2.81	0.062
	*PI*Pa*	4	320,577	80,144	0.38	0.820
	Seeds	29	6,977,415	240,601	1.15	0.276
	Error	232	48,353,774	208,421		
	Total	269	66,550,684			
3	*PI*	2	1,417,618	708,809	0.79	0.456
	Pa	2	1,749,929	874,964	0.97	0.380
	*PI*Pa*	4	3,050,857	762,714	0.85	0.497
	Seeds	29	33,060,258	1,140,009	1.27	0.173
	Error	232	208,932,710	900,572		
	Total	269	248,211,371			
4	*PI*	2	9,208,263	4,604,132	0.95	0.388
	Pa	2	10,839,612	5,419,806	1.12	0.328
	*PI*Pa*	4	8,930,587	2,232,647	0.46	0.764
	Seeds	29	171,994,153	5,930,833	1.23	0.206
	Error	232	1,122,530,543	4,838,494		
	Total	269	1,323,503,159			
5	*PI*	2	1,101,544	550,772	1.38	0.253
	Pa	2	1,424,859	712,429	1.79	0.169
	*PI*Pa*	4	305,787	76,447	0.19	0.942
	Seeds	29	11,690,452	403,119	1.01	0.452
	Error	232	92,302,925	397,857		
	Total	269	106,825,567			
6	*PI*	2	5,689,355	2,844,678	3.35	**0.037**
	Pa	2	1,853,119	926,560	1.09	0.337
	*PI*Pa*	4	3,341,783	835,446	0.98	0.416
	Seeds	29	37,621,130	1,297,280	1.53	**0.047**
	Error	232	196,798,215	848,268		
	Total	269	245,303,602			
7	*PI*	2	23,156,156	11,578,078	2.75	0.066
	Pa	2	11,461,652	5,730,826	1.36	0.258
	*PI*Pa*	4	21,070,907	5,267,727	1.25	0.290
	Seeds	29	103,618,664	3,573,057	0.85	0.692
	Error	232	976,837,736	4,210,507		
	Total	269	1,136,145,115			

Problems	*Source*	*DF*	*SS*	*MS*	*F*	*P*
8	*PI*	2	24,363,035	12,181,517	7.82	**0.001**
	Pa	2	3,130,959	1,565,479	1.00	0.368
	*PI*Pa*	4	13,162,120	3,290,530	2.11	0.080
	Seeds	29	32,649,099	1,125,831	0.72	0.852
	Error	232	361,618,994	1,558,703		
	Total	269	434,924,207			
9	*PI*	2	71,907,246	35,953,623	4.89	**0.008**
	Pa	2	7,519,674	3,759,837	0.51	0.600
	*PI*Pa*	4	22,509,946	5,627,486	0.77	0.549
	Seeds	29	291,826,850	10,062,995	1.37	0.107
	Error	232	1,706,503,430	7,355,618		
	Total	269	2,100,267,145			
10	*PI*	2	27,130,907	13,565,453	0.92	0.400
	Pa	2	36,203,923	18,101,961	1.23	0.295
	*PI*Pa*	4	88,284,774	22,071,193	1.50	0.204
	Seeds	29	531,114,905	18,314,307	1.24	0.192
	Error	232	3,420,245,184	14,742,436		
	Total	269	4,102,979,692			
11	*PI*	2	148002875	74001438	4.68	**0.010**
	Pa	2	47204008	23602004	1.49	0.227
	*PI*Pa*	4	119012484	29753121	1.88	0.115
	Seeds	29	531110852	18314167	1.16	0.272
	Error	232	3669965471	15818817		
	Total	269	4515295691			

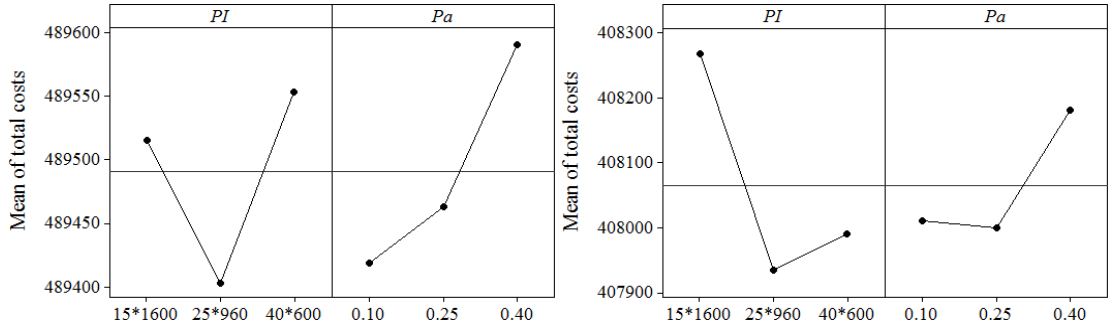
ผลการสรุปค่าพารามิเตอร์ที่เหมาะสมของวิธีการ CS สำหรับแนวทางการกำหนดค่าพารามิเตอร์ที่เหมาะสมในแต่ละปัจจัยของวิธีการ CS สำหรับ 11 ปัญหา จะพิจารณาได้จากจุดต่ำที่สุด (ค่าเฉลี่ยต้นทุนรวมที่น้อยที่สุด) จากกราฟผลกระทบปัจจัยหลัก (Main effect plot) ซึ่งจะแสดงถึงระดับของแต่ละพารามิเตอร์ทั้ง 3 ระดับ (แกน x) กับต้นทุนรวมทั้งหมดเฉลี่ย (แกน y) สำหรับกราฟแสดงผลกระทบหลักสำหรับวิธีการ CS กับ 11 โจทย์ปัญหาแสดงดังภาพ 12 ถึงภาพ 17



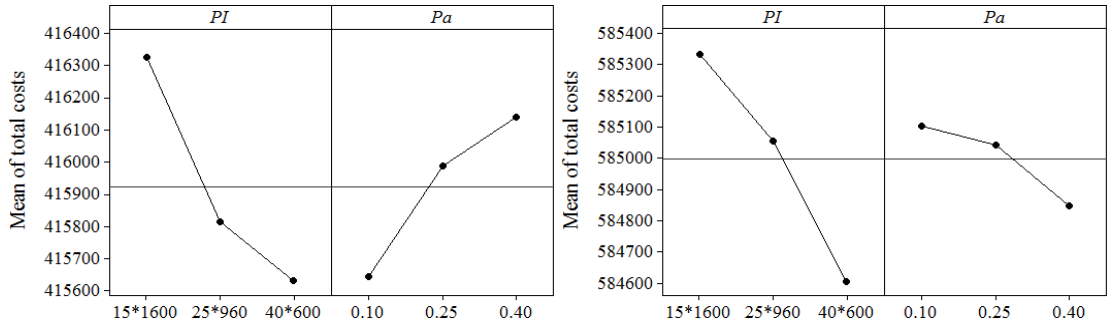
ภาพ 12 แสดงกราฟผลกระทบหลักของปัจจัย *PI* และ P_a สำหรับปัญหา 1 (ภาพซ้าย) และปัญหา 2 (ภาพขวา)



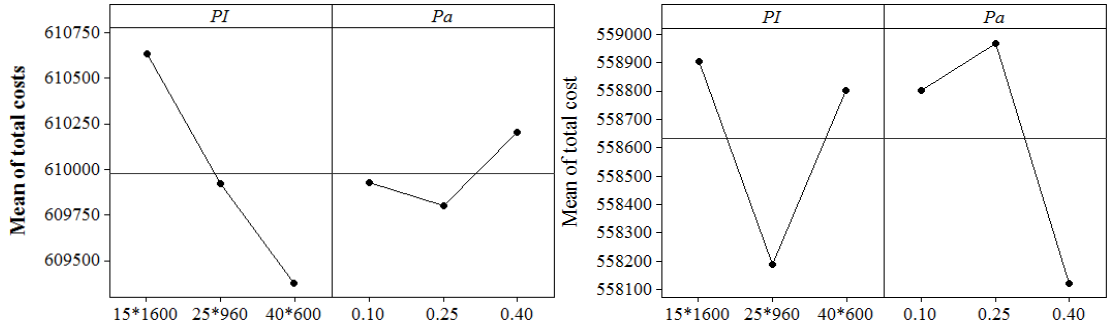
ภาพ 13 แสดงกราฟผลกระทบหลักของปัจจัย *PI* และ P_a สำหรับปัญหา 3 (ภาพซ้าย) และปัญหา 4 (ภาพขวา)



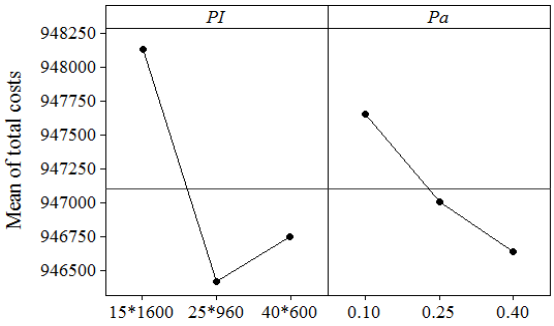
ภาพ 14 แสดงกราฟผลกระทบหลักของปัจจัย *PI* และ P_a สำหรับปัญหา 5 (ภาพซ้าย) และปัญหา 6 (ภาพขวา)



ภาพ 15 แสดงกราฟผลกระทบหลักของปัจจัย *PI* และ P_a สำหรับปัญหา 7 (ภาพซ้าย) และปัญหา 8 (ภาพขวา)



ภาพ 16 แสดงกราฟผลกระทบหลักของปัจจัย *PI* และ P_a สำหรับปัญหา 9 (ภาพซ้าย) และปัญหา 10 (ภาพขวา)



ภาพ 17 แสดงกราฟผลกระทบหลักของปัจจัย *PI* และ P_a สำหรับปัญหา 11

ผลการสรุปค่าพารามิเตอร์ที่เหมาะสมจากภาพ 12 ถึงภาพ 17 จะเห็นได้ว่าระดับของปัจจัยที่เหมาะสมซึ่งจะทำให้เกิดต้นทุนรวมทั้งหมดมีค่าน้อยที่สุดสำหรับวิธีการ CS กับปัญหาทั้ง 11 โจทย์นั้น ปัจจัย P_a ควรกำหนดในช่วง 0.1 ถึง 0.4 เพราะว่าผลการวิเคราะห์ความแปรปรวน แสดงให้เห็นว่าทุกคู่ระดับของปัจจัย P_a แตกต่างแบบไม่มีนัยสำคัญทางสถิติในทุกโจทย์ปัญหา ในขณะที่ปัจจัย *PI* สำหรับโจทย์ 1, 2, 6, 8, 9, และ 11 นั้นผลการวิเคราะห์ความแปรปรวน แสดงให้เห็นว่ามีบางคู่ระดับของปัจจัย *PI* แตกต่างแบบมีนัยสำคัญทางสถิติ

ดังนั้นค่าพารามิเตอร์ที่เหมาะสมที่สุดสำหรับวิธีการ CS ที่ประกอบด้วยปัจจัย *PI* และ P_aทั้ง 11 โจทย์ซึ่งถูกค้นหาด้วยวิธีการออกแบบการทดลองและการวิเคราะห์ทางสถิติ (Experimental design and analysis: EDA) ได้ถูกสรุปดังตาราง 5 (คอลัมน์ซ้าย) อย่างไรก็ตามค่าพารามิเตอร์ที่เหมาะสมของบางปัจจัยอาจถูกสรุปไว้หลายค่า แต่ในทางปฏิบัติแต่ละปัจจัยจะต้องเลือกค่าที่เหมาะสมที่สุดเพียงหนึ่งค่าสำหรับนำไปใช้ในการทดลองหรือนำไปใช้ต่อไป (Selected appropriate parameter settings for CS) ซึ่งได้ได้ถูกสรุปดังตาราง 5 (คอลัมน์ขวา)

ตาราง 5 แสดงสรุปค่าพารามิเตอร์ที่เหมาะสมสำหรับวิธีการ CS สำหรับ 11 โจทย์

Problems	Concluded appropriate parameter settings for		Selected appropriate parameter	
	PI	*Pa*	*PI*	*Pa*
1	15*1600	0.1, 0.25, 0.4	15*1600	0.25
2	15*1600	0.1, 0.25, 0.4	15*1600	0.10
3	15*1600, 25*960, 40*600	0.1, 0.25, 0.4	25*960	0.40
4	15*1600, 25*960, 40*600	0.1, 0.25, 0.4	15*1600	0.25
5	15*1600, 25*960, 40*600	0.1, 0.25, 0.4	25*960	0.10
6	25*960, 40*600	0.1, 0.25, 0.4	25*960	0.25
7	15*1600, 25*960, 40*600	0.1, 0.25, 0.4	40*600	0.10
8	40*600	0.1, 0.25, 0.4	40*600	0.40
9	25*960, 40*600	0.1, 0.25, 0.4	40*600	0.25
10	15*1600, 25*960, 40*600	0.1, 0.25, 0.4	25*960	0.40
11	25*960, 40*600	0.1, 0.25, 0.4	25*960	0.40

4.2 ผลการทดลองที่ 2: การปรับเปลี่ยนกระบวนการวิธีการ CS

การทดลองนี้ได้ถูกออกแบบเพื่อทดสอบและเปรียบเทียบประสิทธิภาพของวิธีการ CS ที่มีการปรับเปลี่ยนกระบวนการ (Process modification) โดยประยุกต์ใช้การเดินสุ่มแบบ Lévy flights (CSLF) และวิธีการ CS โดยประยุกต์ใช้การเดินสุ่มแบบ Gaussian random walks (CSGRW) โดยทั้งสองวิธีการจะใช้ค่าพารามิเตอร์ที่เหมาะสมที่ได้จากการออกแบบการทดลองและการวิเคราะห์เชิงสถิติ (Experimental design and analysis) จากการทดลองแรกในทุกโจทย์ปัญหา สำหรับการเปรียบเทียบประสิทธิภาพของวิธีการต่างๆ ในการจัดตารางเรียนตารางสอนเพื่อให้เกิดต้นทุนรวมจากการใช้ทรัพยากรทางการศึกษาของมหาวิทยาลัยน้อยที่สุดกับโจทย์ปัญหาที่ได้สร้างไว้จำนวน 11 โจทย์นั้น (แสดงดังตาราง 2) จะถูกวิเคราะห์โดย ค่าที่น้อยที่สุด (Minimum) ค่าที่มากที่สุด (Maximum) ค่าเฉลี่ย (Average) ค่าเบี่ยงเบนมาตรฐาน (Standard deviation: SD) และเวลาเฉลี่ยที่ใช้ในการประมวล (Average execution time: Time) ซึ่งมีหน่วยเป็นนาที (Minutes) นอกจากนี้ค่า *T* ที่ได้จากการวิเคราะห์ *t*-test และค่า *P* value ได้ถูกนำมาวิเคราะห์ประสิทธิภาพของวิธีการที่ได้นำเสนออีกด้วยดังตาราง 6 โดยในการทดลองนี้ได้กำหนดจำนวนการค้นหาคำตอบในแต่ละโจทย์ปัญหาจะถูกกำหนดไว้ที่ 24,000 คำตอบ ซึ่งทั้ง 2 วิธีการจะทำการทดลองซ้ำจำนวน 30 ครั้งด้วยค่าหมายเลขในการสุ่มที่แตกต่างกันในแต่ละโจทย์ปัญหา ดังนั้นจำนวนการรันทั้งหมดในการทดลองนี้ทั้งหมด 11 โจทย์จะเท่ากับ 660 (2*11*30) รัน

ผลการทดลองดังตาราง 6 พบว่า วิธีการ CSLF สามารถจัดตารางสอนโดยมีต้นทุนรวมเฉลี่ย (Average) น้อยกว่าวิธีการ CSGRW เกือบทุกโจทย์ปัญหา ในขณะที่วิธีการ CSGRW จะสามารถจัดตารางสอนได้ผลลัพธ์ที่ดีกว่าวิธีการ CSLF เฉพาะปัญหาที่ 9 สำหรับค่าที่น้อยที่สุด (Minimum) ที่ได้จากวิธีการ CSLF จะดีกว่าค่าดังกล่าวที่ได้จากวิธีการ CSGRW ในบางโจทย์ปัญหา นอกจากนี้ค่าค่าเบี่ยงเบนมาตรฐาน (SD) และเวลาที่ใช้ในการประมวลผลเฉลี่ยที่ได้จากทั้งสองวิธีการจะมีความแตกต่างกันไม่มากนัก (หน่วย: นาที)

ผลการทดลองในด้านการวิเคราะห์ทางสถิติพบว่า ประสิทธิภาพของวิธีการ CSLF และวิธีการ CSGRW จะแตกต่างกันอย่างมีนัยสำคัญทางสถิติที่ระดับความเชื่อมั่นที่ 95 เปอร์เซ็นต์จากการทดสอบแบบ *t*-test (*P* value ≤ 0.05) ในโจทย์ปัญหาที่ 1 ปัญหาที่ 4 และปัญหาที่ 7 ซึ่งจะเห็นได้ว่าเป็นปัญหาที่อยู่ในขนาดเล็กและขนาดกลางเท่านั้น ดังนั้นจึงสามารถสรุปได้ว่า ประสิทธิภาพของวิธีการ CSLF และวิธีการ CSGRW แตกต่างกันอย่างไม่มีนัยสำคัญทางสถิติเกือบทุกโจทย์ปัญหาโดยเฉพาะในปัญหาขนาดใหญ่

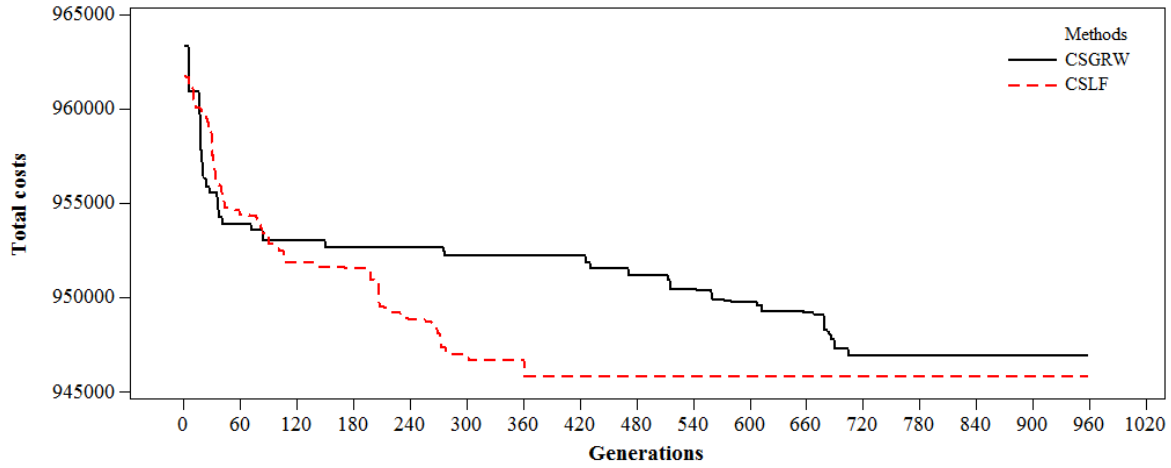
นอกจากนี้ ผลการเปรียบเทียบประสิทธิภาพในด้านความเร็วในการลู่เข้าสู่คำตอบที่ดีที่สุดตั้งแต่เริ่มต้นการประมวลผล (Best so far solution: BSF) ของวิธีการ CSLF และวิธีการ CSGRW ยกตัวอย่างในโจทย์ปัญหาขนาดใหญ่ที่ 11 แสดงดังภาพ 18 ซึ่งจะเห็นได้ว่า วิธีการ CSGRW แม้ว่าจะลู่เข้าสู่คำตอบที่ดีได้เร็วกว่าวิธีการ CSLF ในช่วงแรก แต่ในรอบการประมวลผลช่วงกลางๆ ไปจนถึงช่วงการประมวลผลรอบการค้นหาท้ายๆ วิธีการ CSLF จะมีการลู่เข้าสู่คำตอบที่ดีกว่าวิธีการ CSGRW อย่างเห็นได้ชัด

ตาราง 6 แสดงผลการเปรียบเทียบระหว่างวิธีการ CSLF และ CSGRW

Prob.	Methods	Best so far solutions					*t*-test	
		Minimum	*Maximum*	*Average*	*SD*	*Time* (*minutes*)	*T* values	*P* values
1	CSGRW	202,054.50	202,272.50	202,147.83	54.10	2.44	3.11	**0.003**
	CSLF	**201,998.00**	**202,225.00**	**202,103.62**	56.02	2.53		
2	CSGRW	378,485.50	379,806.00	379,039.18	341.62	10.05	1.26	0.211
	CSLF	**377,927.25**	**379,754.50**	**378,919.28**	390.96	9.86		
3	CSGRW	**301,028.25**	305,228.75	303,160.19	1,052.36	21.79	0.06	0.956
	CSLF	301,125.25	**305,010.75**	**303,146.30**	866.33	24.49		
4	CSGRW	**299,674.50**	308,693.75	304,171.83	2,322.36	12.16	2.05	**0.045**
	CSLF	300,210.00	**307,450.25**	**302,937.73**	2,337.29	13.25		
5	CSGRW	**488,054.75**	490,183.00	489,362.45	534.46	25.99	0.62	0.540
	CSLF	488,141.50	**490,141.00**	**489,286.09**	416.09	28.24		
6	CSGRW	**406,083.25**	410,164.75	407,872.34	910.48	30.06	0.46	0.645
	CSLF	406,149.75	**409,963.50**	**407,758.56**	991.47	35.00		
7	CSGRW	412,589.00	418,821.25	416,126.10	1,488.51	21.85	2.96	**0.005**
	CSLF	**412,089.00**	**418,062.50**	**414,820.20**	1,901.27	23.14		
8	CSGRW	**581,132.00**	587,924.25	584,719.17	1,569.00	58.34	0.66	0.514
	CSLF	582,654.75	**587,650.00**	**584,485.26**	1,155.05	57.98		
9	CSGRW	604,809.25	**612,667.00**	**609,530.52**	2,146.03	32.66	-0.12	0.902
	CSLF	**602,771.25**	615,903.25	609,619.51	3,317.79	40.79		
10	CSGRW	551,052.50	565,639.25	558,339.95	3,861.85	37.06	1.49	0.143
	CSLF	**546,314.50**	**563,997.25**	**556,667.98**	4,798.57	37.73		
11	CSGRW	**932,438.50**	955,456.00	947,073.63	4,194.05	141.78	1.15	0.256
	CSLF	939,531.25	**955,056.25**	**945,859.18**	4,000.94	145.29		

4.3 ผลการทดลองที่ 3: การปรับปรุงประสิทธิภาพวิธีการ CS แบบผสมผสาน

การทดลองนี้ได้ถูกออกแบบเพื่อทำการทดสอบและเปรียบเทียบประสิทธิภาพของวิธีการ CSLF ที่ได้รับการปรับปรุงประสิทธิภาพแบบผสมผสาน (Hybridisation) กับวิธีการ Local search (LS) ซึ่งวิธีการ LS ในการทดลองนี้จะประกอบไปด้วย 2 วิธีการคือ Insertion operator (IO) และ Exchange operator (EO) ดังนั้นในการทดลองนี้จะมีวิธีการทั้งหมด 3 วิธีการสำหรับทำการเปรียบเทียบในแต่ละโจทย์ปัญหา ประกอบด้วย วิธีการ CSLF วิธีการ CSLF+IO และวิธีการ CSLF+EO



ภาพ 18 แสดงกราฟเปรียบเทียบการลู่เข้าสู่คำตอบที่ดีของวิธีการ CSLF และ CSGRW สำหรับปัญหา 11

สำหรับการเปรียบเทียบประสิทธิภาพของวิธีการต่างๆ จะถูกทดสอบกับโจทย์ปัญหาที่ได้สร้างไว้จำนวน 11 โจทย์ (แสดงดังตาราง 2) และจะถูกวิเคราะห์โดย ค่าที่น้อยที่สุด (Minimum) ค่าที่มากที่สุด (Maximum) ค่าเฉลี่ย (Average) ค่าเบี่ยงเบนมาตรฐาน (SD) และเวลาเฉลี่ยที่ใช้ในการประมวล (Time) ซึ่งมีหน่วยเป็นนาที (Minutes) ส่วนค่า *T* ที่ได้จากการวิเคราะห์ด้วยวิธีการ Tukey และค่า *P* value จะถูกนำมาวิเคราะห์ประสิทธิภาพระหว่างวิธีการแบบเดิมกับวิธีการที่ได้รับการผสมผสานอีกด้วย นอกจากนี้ในแต่ละโจทย์ปัญหาจะกำหนดจำนวนการค้นหาคำตอบไว้ที่ 24,000 คำตอบและใช้วิธีการจัดลำดับรายวิชาแบบ LUPD โดยค่าพารามิเตอร์ต่างๆ ที่กำหนดให้กับวิธีการ CSLF นำมาจากตาราง 5 ตามลำดับ ยิ่งไปกว่านั้นทั้ง 3 วิธีการจะถูกทำการทดลองซ้ำจำนวน 30 ครั้งด้วยค่าหมายเลขในการสุ่มที่แตกต่างกันในแต่ละโจทย์ปัญหา ดังนั้นจำนวนการรันทั้งหมดในการทดลองนี้ทั้งหมด 11 โจทย์จะเท่ากับ 990 (3*11*30) รัน แสดงผลดังจากตาราง 7

ตาราง 7 แสดงผลการเปรียบเทียบวิธีการ CSLF ผสมผสานกับวิธีการ LS

Prob.	Methods	Best so far solutions					*Tukey's method*	
		Minimum	*Maximum*	*Average*	*SD*	*Time* (*minutes*)	*T* values	*P* values
1	CSLF	201,998.00	202,225.00	202,103.62	56.02	2.38		
	CSLF+IO	202,032.50	202,202.00	202,111.42	41.21	1.86	-0.70	0.767
	CSLF+EO	**201,878.50**	**201,999.50**	**201,933.75**	**28.73**	1.33	15.14	**0.000**
2	CSLF	377,927.25	379,754.50	378,919.28	390.96	13.34		
	CSLF+IO	**374,599.50**	379,625.25	378,993.15	869.67	7.93	-0.52	0.864
	CSLF+EO	378,097.25	**378,759.75**	**378,578.39**	**128.49**	4.70	2.38	0.051
3	CSLF	**301,125.25**	305,010.75	303,146.30	866.33	20.62		
	CSLF+IO	301,457.25	305,675.75	303,555.73	875.72	16.05	-2.01	0.116
	CSLF+EO	301,621.50	**304,104.00**	**302,957.78**	**594.69**	9.64	0.93	0.626

Prob.	Methods	Best so far solutions					*Tukey's method*	
		Minimum	*Maximum*	*Average*	*SD*	*Time* (*minutes*)	*T* values	*P* values
4	CSLF	300,210.00	307,450.25	302,937.73	2,337.29	14.07		
	CSLF+IO	300,121.50	303,155.00	301,810.26	602.98	8.90	3.11	**0.007**
	CSLF+EO	**299,378.25**	**300,493.00**	**299,818.81**	**283.40**	5.71	8.60	**0.000**
5	CSLF	488,141.50	490,141.00	489,286.09	416.09	19.76		
	CSLF+IO	488,741.00	490,562.50	489,143.93	**402.93**	16.28	1.30	0.399
	CSLF+EO	**487,315.00**	**489,495.25**	**488,342.48**	450.17	9.23	8.63	**0.000**
6	CSLF	406,149.75	409,963.50	407,758.56	991.47	35.00		
	CSLF+IO	406,073.00	409,059.75	407,526.06	692.80	25.65	1.13	0.497
	CSLF+EO	**405,637.75**	**408,083.75**	**407,048.07**	**657.90**	15.39	3.46	**0.002**
7	CSLF	412,089.00	418,062.50	414,820.20	1,901.27	23.14		
	CSLF+IO	410,564.75	413,794.75	412,277.72	799.64	15.75	8.11	**0.000**
	CSLF+EO	**408,279.50**	**410,263.25**	**409,320.88**	**409.69**	9.07	17.54	**0.000**
8	CSLF	582,654.75	587,650.00	584,485.26	1,155.05	57.98		
	CSLF+IO	581,817.00	585,534.50	583,827.22	765.29	31.10	2.93	**0.012**
	CSLF+EO	**581,335.25**	**583,897.00**	**582,670.56**	**597.82**	18.75	8.07	**0.000**
9	CSLF	602,771.25	615,903.25	609,619.51	3,317.79	40.79		
	CSLF+IO	602,841.25	607,679.50	605,291.43	1,161.05	24.98	7.95	**0.000**
	CSLF+EO	**598,125.75**	**602,827.50**	**600,343.48**	**988.88**	16.07	17.04	**0.000**
10	CSLF	546,314.50	563,997.25	556,667.98	4,798.57	37.73		
	CSLF+IO	547,759.50	553,854.50	550,632.50	1,495.93	33.62	7.93	**0.000**
	CSLF+EO	**540,221.00**	**543,150.50**	**541,621.70**	**913.53**	24.64	19.76	**0.000**
11	CSLF	939,531.25	955,056.25	945,859.18	4,000.94	145.29		
	CSLF+IO	926,254.75	933,329.00	930,400.41	**1,874.13**	61.96	21.46	**0.000**
	CSLF+EO	**909,172.75**	**917,453.00**	**914,026.72**	1,954.70	36.86	44.20	**0.000**

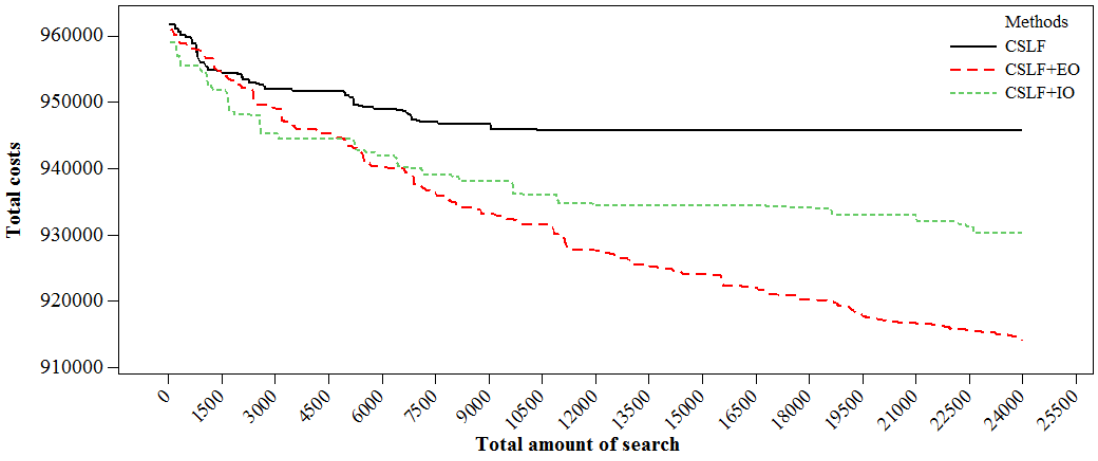
ผลการทดลองโดยสรุปจากตาราง 7 พบว่า วิธีการ CSLF+EO จะมีประสิทธิภาพในการจัดตารางเรียนตารางสอนโดยมีต้นทุนรวมเฉลี่ยน้อยกว่าวิธีการ CSLF+IO และวิธีการ CSLF ในทุกโจทย์ปัญหาและใช้เวลาในการจัดตารางน้อยที่สุดในทุกโจทย์ปัญหา นอกจากนี้ยังพบว่า วิธีการ CSLF+EO จะมีประสิทธิภาพในการจัดตารางเรียนตารางสอนโดยมีต้นทุนรวมเฉลี่ยน้อยที่สุดดีกว่าวิธีการ CSLF แบบทั่วไปในทุกโจทย์ปัญหา แม้ว่าวิธีการ CSLF+IO จะมีประสิทธิภาพที่ด้อยกว่าวิธีการ CSLF ในโจทย์ปัญหา 1 ปัญหา 2 และปัญหา 3 แต่ในปัญหา 4 ถึงปัญหา 11 ซึ่งเป็นปัญหาขนาดกลางไปจนถึงขนาดใหญ่นั้นวิธีการ CSLF+IO จะมีประสิทธิภาพที่ดีกว่าวิธีการ CSLF นอกจากนี้ค่า Minimum ค่า Maximum และค่าเบี่ยงเบนมาตรฐาน (SD) ที่ได้จากวิธีการ

CSLF+EO จะมีค่าที่น้อยกว่าค่าดังกล่าวในวิธีการ CSLF และวิธีการ CSLF+IO เกือบทุกโจทย์ปัญหา โดยเฉพาะในปัญหาขนาดกลางไปจนถึงปัญหาขนาดใหญ่

สำหรับเวลาเฉลี่ยที่ใช้ในการประมวลผลของวิธีการ CSLF ที่ได้รับการผสมผสานกับวิธีการ LS นั้นมีค่าน้อยกว่าวิธีการ CSLF ที่ไม่ได้รับการผสมผสานกับวิธีการ LS อย่างมากในทุกโจทย์ปัญหา โดยเฉพาะอย่างยิ่งวิธีการ CSLF+EO นั้นใช้เวลาในการประมวลผลน้อยที่สุด เหตุผลเนื่องจากว่าในการทดลองนี้ได้กำหนดให้มีจำนวนการค้นหาคำตอบที่เท่ากันที่ 24,000 คำตอบ ซึ่งวิธีการ CSLF+EO และ วิธีการ CSLF+IO ในแต่ละรอบการวนซ้ำ (iteration) จะทำให้เกิดการพัฒนาคำตอบเป็น 2 เท่าของวิธีการ CSLF แบบทั่วไปจึงทำให้ต้องลดจำนวนรอบการวนซ้ำลงครึ่งหนึ่งเพื่อให้จำนวนการค้นหายังคงเท่ากับ 24,000 คำตอบก่อนทำการเปรียบเทียบ นอกจากนี้วิธีการ LS ทั้งสองวิธีการนั้นมีขั้นตอนการทำงานที่ง่ายและไม่ซับซ้อนจึงทำให้ใช้เวลาเพียงเล็กน้อยในการพัฒนาคำตอบเมื่อเปรียบเทียบกับการพัฒนาคำตอบของวิธีการ CSLF ด้วยวิธีการ random keys ซึ่งจะมีขั้นตอนและการคำนวณที่ยุ่งยากมากกว่า

สำหรับผลจากการวิเคราะห์การเปรียบเทียบแบบหลายระดับด้วยวิธีการ Tukey แสดงดังตาราง 7 จะเห็นได้ว่า คำตอบที่ได้จากวิธีการ CSLF ที่ได้รับการผสมผสานกับวิธีการ LS (วิธีการ CSLF+IO และ/หรือ วิธีการ CSLF+EO) จะแตกต่างอย่างมีนัยสำคัญทางสถิติกับวิธีการ CSLF ที่ระดับความเชื่อมั่นที่ 95 เปอร์เซ็นต์ เกือบทุกโจทย์ปัญหา ยกเว้นปัญหา 2 และปัญหา 3 ซึ่งจัดอยู่ในกลุ่มของปัญหาขนาดเล็ก

นอกจากนี้ ผลการเปรียบเทียบประสิทธิภาพในด้านความเร็วการลู่เข้าสู่คำตอบที่ดีตั้งแต่เริ่มต้นการประมวลผล (BSF) ของวิธีการ CSLF วิธีการ CSLF+EO และวิธีการ CSLF+IO ยกตัวอย่างในโจทย์ปัญหาขนาดใหญ่ที่ 11 ดังภาพ 19 จะเห็นได้ว่าวิธีการ CSLF+EO มีประสิทธิภาพในการลู่เข้าสู่คำตอบที่ดีได้เร็วกว่าวิธีการอื่นๆ ซึ่งวิธีการ CSLF+EO และ CSLF+IO จะลู่เข้าสู่คำตอบที่ดีได้เร็วกว่าวิธีการ CSLF ตั้งแต่เริ่มต้นการประมวลผลไปจนสิ้นสุดการประมวลผล ดังนั้นจึงสามารถสรุปได้ว่าวิธีการ CSLF ที่ได้รับการผสมผสานกับวิธีการ LS จะมีประสิทธิภาพในการค้นหาคำตอบที่ดีได้เร็วขึ้นเมื่อกำหนดจำนวนการค้นหาคำตอบที่เท่ากัน



ภาพ 19 แสดงกราฟเปรียบเทียบการลู่เข้าสู่คำตอบที่ดีของวิธีการ CSLF, CSLF+IO และ CSLF+IO สำหรับปัญหา 11

บทที่ 5
สรุปผลการวิจัย

โปรแกรมช่วยในการจัดตารางเรียนตารางสอนในระดับอุดมศึกษาแบบอัตโนมัติโดยประยุกต์ใช้วิธีการคุกคูเสิร์ช (CST) ได้ถูกพัฒนาขึ้นเพื่อแก้ปัญหาการจัดตารางสอนในระดับอุดมศึกษา สำหรับโปรแกรมที่ถูกพัฒนาขึ้นจะพิจารณาการจัดตารางเรียนตารางสอนโดยคำนึงถึงต้นทุนการดำเนินการรวมของมหาวิทยาลัย (Total university operating costs) ที่น้อยที่สุดตามข้อบังคับรองของงานวิจัย ซึ่งประกอบไปด้วย ต้นทุนการใช้ห้องเรียนแต่ละห้อง ต้นทุนการจ้างอาจารย์ และต้นทุนการทำความสะอาดหรือจัดเตรียมห้อง โดยต้นทุนในแต่ละส่วนจะมีการกำหนดค่าถ่วงน้ำหนักที่แตกต่างกันไปตามคุณสมบัติและความชอบของแต่ละบุคคล ในแต่ละวัน ในแต่ละคาบเวลา โดยโจทย์ปัญหาการจัดตารางสอนได้ถูกออกแบบและสร้างขึ้นมาจากข้อมูลจริงของคณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร ประกอบด้วย 11 โจทย์ที่มีขนาดเล็กไปจนถึงขนาดใหญ่ โดยพบว่าโปรแกรมจัดตารางเรียนตารางสอนที่ได้ถูกพัฒนาขึ้นนั้นสามารถทำงานได้อย่างถูกต้องและมีประสิทธิภาพ

ผลการค้นหาอิทธิพลของปัจจัยและการกำหนดค่าพารามิเตอร์ที่เหมาะสมสำหรับการทำงานของวิธีการ CS โดยการออกแบบการทดลองและการวิเคราะห์ความแปรปรวน พบว่า ผลกระทบของปัจจัยหลัก (Main effect) ของวิธีการ CS คือ จำนวนการค้นหาคำตอบทั้งหมด (*PI*) จะมีนัยสำคัญทางสถิติที่ระดับความเชื่อมั่น 95% ในบางโจทย์ปัญหา นอกจากนี้ค่าความน่าจะเป็นในการละทิ้งคำตอบ (P_a) และผลกระทบของปัจจัยร่วม (Interaction) จาก 2 ปัจจัย จะไม่มีนัยสำคัญทางสถิติที่ระดับความเชื่อมั่น 95% ในทุกโจทย์ปัญหา อีกทั้งยังพบว่า หมายเลขของการสุ่ม (Seeds) นั้นไม่มีนัยสำคัญทางสถิติที่ระดับความเชื่อมั่น 95% เกือบทุกโจทย์ปัญหา ในขณะที่ค่าพารามิเตอร์ที่เหมาะสมของวิธีการ CS จะมีความแตกต่างกันออกไปเมื่อโจทย์ปัญหาที่ได้กำหนดขึ้นมีขนาดแตกต่างกัน

ผลการทดสอบและเปรียบเทียบประสิทธิภาพของวิธีการ CS ที่มีการปรับเปลี่ยนกระบวนการ (Process modification) โดยประยุกต์ใช้การเดินสุ่มแบบ Lévy Flights (CSLF) และวิธีการ CS โดยประยุกต์ใช้การเดินสุ่มแบบ Gaussian Random Walks (CSGRW) ในการจัดตารางเรียนตารางสอนเพื่อให้เกิดต้นทุนรวมจากการใช้ทรัพยากรทางการศึกษาของมหาวิทยาลัยน้อยที่สุดนั้น พบว่าวิธีการ CSLF สามารถจัดตารางสอนโดยมีต้นทุนรวมเฉลี่ย (Average) น้อยกว่าวิธีการ CSGRW เกือบทุกโจทย์ปัญหา ในขณะที่วิธีการ CSGRW จะสามารถจัดตารางสอนได้ผลลัพธ์ที่ดีกว่าวิธีการ CSLF ในบางโจทย์ปัญหา นอกจากนี้ค่าค่าเบี่ยงเบนมาตรฐาน (SD) และเวลาที่ใช้ในการประมวลผลเฉลี่ยที่ได้จากทั้งสองวิธีการจะมีความแตกต่างกันไม่มาก สำหรับผลการวิเคราะห์ทางสถิติพบว่า ประสิทธิภาพของวิธีการ CSLF และวิธีการ CSGRW แตกต่างกันอย่างไม่มีนัยสำคัญทางสถิติที่ระดับความเชื่อมั่นที่ 95 เปอร์เซ็นต์เกือบทุกโจทย์ปัญหาโดยเฉพาะในปัญหาขนาดใหญ่ นอกจากนี้ประสิทธิภาพในด้านความเร็วในการลู่เข้าสู่คำตอบที่ดีที่สุดตั้งแต่เริ่มต้นการประมวลผล (BSF) ของวิธีการ CSLF และ CSGRW ยังพบว่า วิธีการ CSGRW จะลู่เข้าสู่คำตอบที่ดีได้เร็วกว่าวิธีการ CSLF ในช่วงแรก แต่ใน

รอบการประมวลผลช่วงกลางๆ ไปจนถึงช่วงการประมวลผลรอบการค้นหาท้ายๆ วิธีการ CSLF จะมีการลู่เข้าสู่คำตอบที่ดีกว่า

สำหรับผลการทดสอบประสิทธิภาพของวิธีการ CS หลังจากทำการผสมผสานกับวิธีการค้นหาแบบเฉพาะพื้นที่ (Local Search: LS) พบว่า วิธีการ CSLF+EO จะมีประสิทธิภาพในการจัดตารางเรียนตารางสอนโดยมีต้นทุนรวมเฉลี่ยน้อยกว่าวิธีการ CSLF+IO และวิธีการ CSLF ในทุกโจทย์ปัญหาและใช้เวลาในการจัดตารางน้อยที่สุดในทุกโจทย์ปัญหา นอกจากนี้ยังพบว่าวิธีการ CSLF+IO จะมีประสิทธิภาพที่ด้อยกว่าวิธีการ CSLF ในโจทย์ปัญหา 1 ปัญหา 2 และปัญหา 3 เท่านั้น แต่ในปัญหา 4 ถึงปัญหา 11 ซึ่งเป็นปัญหาขนาดกลางไปจนถึงขนาดใหญ่นั้นวิธีการ CSLF+IO จะมีประสิทธิภาพที่ดีกว่าวิธีการ CSLF อย่างชัดเจน ยิ่งไปกว่านั้นผลการเปรียบเทียบประสิทธิภาพในด้านความเร็วการลู่เข้าสู่คำตอบที่ดีตั้งแต่เริ่มต้นการประมวลผล (BSF) ของวิธีการ CSLF วิธีการ CSLF+EO และวิธีการ CSLF+IO ยังพบว่า วิธีการ CSLF+EO มีประสิทธิภาพในการลู่เข้าสู่คำตอบที่ดีได้เร็วกว่าวิธีการอื่นๆ ซึ่งวิธีการ CSLF+EO และ CSLF+IO จะลู่เข้าสู่คำตอบที่ดีได้เร็วกว่าวิธีการ CSLF ตั้งแต่เริ่มต้นการประมวลผลไปจนสิ้นสุดการประมวลผล ดังนั้นสามารถสรุปได้ว่าวิธีการ CSLF ที่ได้รับการผสมผสานกับวิธีการ LS จะมีประสิทธิภาพในการค้นหาคำตอบที่ดีได้เร็วขึ้นเมื่อกำหนดจำนวนการค้นหาคำตอบที่เท่ากัน

เอกสารอ้างอิง

[1] P. Pongcharoen, W. Promtet, P. Yenradee, and C. Hicks, “Stochastic optimisation timetabling tool for university course scheduling,” *International Journal of Production Economics,* vol. 112, no. 2, pp. 903-918, Apr, 2008.

[2] E. K. Burke, and S. Petrovic, “Recent research directions in automated timetabling,” *European Journal of Operational Research,* vol. 140, no. 2, pp. 266-280, 2002.

[3] D. Datta, C. M. Fonseca, and K. Deb, “A multi-objective evolutionary algorithm to exploit the similarities of resource allocation problems,” *Journal of Scheduling,* vol. 11, no. 6, pp. 405-419, Dec, 2008.

[4] S. Daskalaki, T. Birbas, and E. Housos, “An integer programming formulation for a case study in university timetabling,” *European Journal of Operational Research,* vol. 153, no. 1, pp. 117-135, 2004.

[5] R. Lewis, “A survey of metaheuristic-based techniques for University Timetabling problems,” *Or Spectrum,* vol. 30, no. 1, pp. 167-190, Jan, 2008.

[6] H. Cambazard, E. Hebrard, B. O’Sullivan, and A. Papadopoulos, “Local search and constraint programming for the post enrolment-based course timetabling problem,” *Annals of Operations Research,* vol. 194, no. 1, pp. 111-135, 2012/04/01, 2012.

[7] E. K. Burke, B. McCollum, A. Meisels, S. Petrovic, and R. Qu, “A graph-based hyper-heuristic for educational timetabling problems,” *European Journal of Operational Research,* vol. 176, no. 1, pp. 177–192, 2007.

[8] S. A. MirHassani, and F. Habibi, “Solution approaches to the course timetabling problem,” *Artificial Intelligence Review,* vol. 39, no. 2, pp. 133-149, Feb, 2013.

[9] Z. Lü, and J.-K. Hao, “Adaptive Tabu Search for course timetabling,” *European Journal of Operational Research,* vol. 200, no. 1, pp. 235-244, 2010.

[10] A. L. a. Bolaji, A. T. Khader, M. A. Al-Betar, and M. A. Awadallah, "Tackling University Course Timetabling Problem Using Artificial Bee Colony Algorithm," *Frontiers in Information Technology*, A.-D. Ali, ed., Jordan: MASAUM Network, 2012.

[11] R. Lewis, “A time-dependent metaheuristic algorithm for post enrolment-based course timetabling,” *Annals of Operations Research,* vol. 194, no. 1, pp. 273-289, Apr, 2012.

[12] S. N. Jat, and S. X. Yang, “A hybrid genetic algorithm and tabu search approach for post enrolment course timetabling,” *Journal of Scheduling,* vol. 14, no. 6, pp. 617-637, Dec, 2011.

[13] T. Thepphakorn, P. Pongcharoen, and C. Hicks, “An Ant Colony Based Timetabling Tool,” *International Journal of Production Economics,* vol. 149, pp. 131–144, 2014.

[14] M. A. Al-Betar, A. T. Khader, and M. Zaman, “University Course Timetabling Using a Hybrid Harmony Search Metaheuristic Algorithm,” *IEEE Transactions on Systems Man and Cybernetics Part C-Applications and Reviews,* vol. 42, no. 5, pp. 664-681, Sep, 2012.

[15] J. K. Hao, and U. Benlic, “Lower bounds for the ITC-2007 curriculum-based course timetabling problem,” *European Journal of Operational Research,* vol. 212, no. 3, pp. 464-472, Aug, 2011.

[16] D. B. Seo, A. I. L. Paz, and J. Miranda, “Information Systems for Organizational Agility: Action Research on Resource Scheduling at the Universidad de Chile,” *Asia Pacific Journal of Information Systems,* vol. 24, no. 4, pp. 417-441, 2014.

[17] C. Torres-Ovalle, J. R. Montoya-Torres, C. L. Quintero-Araújo, A. Sarmiento-Lepesqueur, and M. Castilla-Luna, “University Course Scheduling and Classroom Assignment,” *Ingeniería y Universidad,* vol. 18, pp. 59-75, 2014.

[18] C. Blum, and A. Roli, “Metaheuristics in combinatorial optimization: Overview and conceptual comparison,” *ACM Computing Surveys,* vol. 35, no. 3, pp. 268-308, 2003.

[19] O. Rossi-Doria, M. Sampels, M. Birattari, M. Chiarandini, M. Dorigo, L. M. Gambardella, J. Knowles, M. Manfrin, M. Mastrolilli, B. Paechter, L. Paquete, and T. Stützle, “A Comparison of the Performance of Different Metaheuristics on the Timetabling Problem,” *Lecture Notes in Computer Science,* vol. 2740, pp. 329-351, 2003/01/01, 2003.

[20] K. Socha, M. Sampels, and M. Manfrin, “Ant Algorithms for the University Course Timetabling Problem with Regard to the State-of-the-Art,” *Lecture Notes in Computer Science,* vol. 2611, pp. 334-345, 2003.

[21] C. Meyers, and J. B. Orlin, “Very large-scale neighborhood search techniques in timetabling problems,” *Lecture Notes in Computer Science,* vol. 3867, pp. 24-39, 2007.

[22] E. G. Talbi, *Metaheuristics: From Design to Implementation*: Wiley, 2009.

[23] X.-S. Yang, and S. Deb, “Engineering optimisation by cuckoo search,” *International Journal of Mathematical Modelling and Numerical Optimisation,* vol. 1, no. 4, pp. 330-343, 2010.

[24] X.-S. Yang, "A New Metaheuristic Bat-Inspired Algorithm," *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, Studies in Computational Intelligence J. González, D. Pelta, C. Cruz, G. Terrazas and N. Krasnogor, eds., pp. 65-74: Springer Berlin Heidelberg, 2010.

[25] X.-S. Yang, “Firefly algorithm, stochastic test functions and design optimisation,” *International Journal of Bio-Inspired Computation,* vol. 2, no. 2, pp. 78 - 84, 2010.

[26] A. H. Gandomi, and A. H. Alavi, “Krill herd: A new bio-inspired optimization algorithm,” *Communications in Nonlinear Science and Numerical Simulation,* vol. 17, no. 12, pp. 4831-4845, 2012.

[27] X.-S. Yang, "Flower Pollination Algorithm for Global Optimization," *Unconventional Computation and Natural Computation*, Lecture Notes in Computer Science J. Durand-Lose and N. Jonoska, eds., pp. 240-249: Springer Berlin Heidelberg, 2012.

[28] A. R. Mehrabian, and C. Lucas, “A novel numerical optimization algorithm inspired from weed colonization,” *Ecological Informatics,* vol. 1, no. 4, pp. 355-366, 2006.

[29] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, “GSA: A Gravitational Search Algorithm,” *Information Sciences,* vol. 179, no. 13, pp. 2232-2248, 2009.

[30] H. Shah-Hosseini, "The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm," *International Journal of Bio-Inspired Computation,* vol. 1, no. 1, pp. 71-79, 2009.

[31] P. Civicioglu, "Backtracking Search Optimization Algorithm for numerical optimization problems," *Applied Mathematics and Computation,* vol. 219, no. 15, pp. 8121-8144, 2013.

[32] A. H. Kashan, "League Championship Algorithm: a new algorithm for numerical function optimization," *Proceedings of Soft Computing and Pattern Recognition, 2009. SOCPAR '09*, pp. 43-48, 2009.

[33] P. Civicioglu, and E. Besdok, "A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms," *Artificial Intelligence Review,* vol. 39, no. 4, pp. 315-346, 2013/04/01, 2013.

[34] X. Ouyang, Y. Zhou, Q. Luo, and H. Chen, "A Novel Discrete Cuckoo Search Algorithm for Spherical Traveling Salesman Problem," *Applied Mathematics & Information Sciences,* vol. 7, no. 2, pp. 777-784, 2013.

[35] L. H. Tein, and R. Ramli, "Recent Advancements of Nurse Scheduling Models and A Potential Path," *Proceedings of the 6th IMT-GT Conference on Mathematics, Statistics and its Applications (ICMSA2010)*, pp. 395-409, 2010.

[36] S. Burnwal, and S. Deb, "Scheduling optimization of flexible manufacturing system using cuckoo search-based approach," *The International Journal of Advanced Manufacturing Technology,* vol. 64, no. 5-8, pp. 951-959, 2013/02/01, 2013.

[37] X.-S. Yang, and S. Deb, "Cuckoo Search via Lévy flights," *Proceedings of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009)*, pp. 210-214, December 2009, 2009.

[38] A. Yildiz, "Cuckoo search algorithm for the selection of optimal machining parameters in milling operations," *The International Journal of Advanced Manufacturing Technology,* vol. 64, no. 1-4, pp. 55-61, 2013/01/01, 2013.

[39] Y. Yang, and S. Petrovic, "A Novel Similarity Measure for Heuristic Selection in Examination Timetabling," *Lecture Notes in Computer Science,* vol. 3616, pp. 247-269, 2005/01/01, 2005.

[40] N. Figlali, C. Ozkale, O. Engin, and A. Figlali, "Investigation of Ant System parameter interactions by using design of experiments for job-shop scheduling problems," *Computers & Industrial Engineering,* vol. 56, no. 2, pp. 538-559, Mar, 2009.

[41] B. Naderi, S. M. T. F. Ghomi, and M. Aminnayeri, "A high performing metaheuristic for job shop scheduling with sequence-dependent setup times," *Applied Soft Computing,* vol. 10, no. 3, pp. 703-710, 2010.

[42] P. Pongcharoen, W. Chainate, and P. Thapatsuwan, "Exploration of genetic parameters and operators through travelling salesman problem," *Science Asia,* vol. 33, pp. 215-222, 2007.

[43] H. Aytug, M. Khouja, and F. E. Vergara, "Use of Genetic Algorithms to solve production and operations management problems: A review," *International Journal of Production Research,* vol. 41, no. 17, pp. 3955-4009, 2003.

[44] D. C. Montgomery, *Design and Analysis of Experiments*, 8th ed., New York: John Wiley & Sons, 2012.

[45] W. Chen, G. Fu, P. Tai, and W. Deng, “Process parameter optimization for MIMO plastic injection molding via soft computing,” *Expert Systems with Applications,* vol. 36, no. 2, Part 1, pp. 1114-1122, 2009.

[46] T. Thepphakorn, and P. Pongcharoen, “Heuristic ordering for ant colony based timetabling tool,” *Journal of Applied Operational Research,* vol. 5, no. 3, pp. 113-123, 2013.

[47] N. R. Sabar, M. Ayob, G. Kendall, and R. Qu, “A honey-bee mating optimization algorithm for educational timetabling problems,” *European Journal of Operational Research,* vol. 216, no. 3, pp. 533-543, 2012.

[48] R.-M. Chen, and H.-F. Shih, “Solving University Course Timetabling Problems Using Constriction Particle Swarm Optimization with Local Search,” *Algorithms,* vol. 6, no. 2, pp. 227-244, 2013.

[49] R. Perzina, “Solving the University Timetabling Problem with Optimized Enrollment of Students by a Self-adaptive Genetic Algorithm,” *Lecture Notes in Computer Science,* vol. 3867, pp. 248-263, 2007.

[50] T. Lutuksin, and P. Pongcharoen, “Best-worst ant colony system parameter investigation by using experimental design and analysis for course timetabling problem,” *Proceedings of the International Conference on Computer and Network Technology*, pp. 467-471, 2010.

[51] C. Blum, J. Puchinger, G. R. Raidl, and A. Roli, “Hybrid metaheuristics in combinatorial optimization: A survey,” *Applied Soft Computing,* vol. 11, no. 6, pp. 4135-4151, 2011.

[52] C. Blum, and A. Roli, “Hybrid Metaheuristics: An Introduction,” *Studies in Computational Intelligence,* vol. 114, pp. 1-30, 2008.

[53] Z. N. Azimi, “Hybrid heuristics for examination timetabling problem,” *Applied Mathematics and Computation,* vol. 163, no. 2, pp. 705-733, Apr, 2005.

[54] S. N. Jat, and S. Yang, “A Memetic Algorithm for the University Course Timetabling Problem,” *Proceedings of the 20th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2008),* vol. 1, pp. 427-433, 3-5 Nov. 2008, 2008.

[55] D. F. Shiau, “A hybrid particle swarm optimization for a university course scheduling problem with flexible preferences,” *Expert Systems with Applications,* vol. 38, no. 1, pp. 235-248, Jan, 2011.

[56] K. Murray, T. Müller, and H. Rudová, "Modeling and Solution of a Complex University Course Timetabling Problem," *Practice and Theory of Automated Timetabling VI*, Lecture Notes in Computer Science E. Burke and H. Rudová, eds., pp. 189-209: Springer Berlin Heidelberg, 2007.

[57] H. Rudova, T. Muller, and K. Murray, “Complex university course timetabling,” *Journal of Scheduling,* vol. 14, no. 2, pp. 187-207, Apr, 2011.

[58] T. Muller, “ITC2007 solver description: a hybrid approach,” *Annals of Operations Research,* vol. 172, no. 1, pp. 429-446, Nov, 2009.

[59] S. Abdullah, and H. Turabieh, “On the use of multi neighbourhood structures within a Tabu-based memetic approach to university timetabling problems,” *Information Sciences,* vol. 191, pp. 146-168, May, 2012.

[60] C. Beyrouthy, E. K. Burke, B. McCollum, P. McMullan, and A. J. Parkes, “University space planning and space-type profiles,” *Journal of Scheduling,* vol. 13, no. 4, pp. 363-374, Aug, 2010.

[61] P. De Causmaecker, P. Demeester, and G. Vanden Berghe, “A decomposed metaheuristic approach for a real-world university timetabling problem,” *European Journal of Operational Research,* vol. 195, no. 1, pp. 307-318, May, 2009.

[62] L. D. Gaspero, B. McCollum, and A. Schaerf, *The second international timetabling competition (ITC-2007): Curriculum-based course timetabling (track 3),* QUB/IEEE 2007/08/01, University of Udine DIEGM, Udine, Italy, 2007.

[63] A. Abbas, and E. P. K. Tsang, “Software engineering aspects of constraint-based timetabling - a case study,” *Information and Software Technology,* vol. 46, no. 6, pp. 359-372, May, 2004.

[64] S. Abdullah, E. K. Burke, and B. McCollum, “An Investigation of Variable Neighbourhood Search for University Course Timetabling,” *Proceedings of the 2nd Multidisciplinary Conference on Scheduling: Theory and Applications (MISTA)*, pp. 413-427, 2005.

[65] S. Abdullah, H. Turabieh, B. McCollum, and P. McMullan, "A multi-objective post enrolment course timetabling problems: A new case study." pp. 1-7.

[66] S. Abdullah, H. Turabieh, B. McCollum, and P. McMullan, “A hybrid metaheuristic approach to the university course timetabling problem,” *Journal of Heuristics,* vol. 18, no. 1, pp. 1-23, Feb, 2012.

[67] A. Abuhamdah, and M. Ayob, “Adaptive randomized descent algorithm for solving course timetabling problems,” *International Journal of the Physical Sciences,* vol. 5, no. 16, pp. 2516-2522, Dec, 2010.

[68] A. Abuhamdah, M. Ayob, G. Kendall, and N. R. Sabar, “Population based Local Search for university course timetabling problems,” *Applied Intelligence,* vol. 40, no. 1, pp. 44-53, Jan, 2014.

[69] R. A. Acha, and R. Nieuwenhuis, “Curriculum-based course timetabling with SAT and MaxSAT,” *Annals of Operations Research,* vol. 218, no. 1, pp. 71-91, Jul, 2014.

[70] L. E. Agustin-Blas, S. Salcedo-Sanz, E. G. Ortiz-Garcia, A. Portilla-Figueras, and A. M. Perez-Bellido, “A hybrid grouping genetic algorithm for assigning students to preferred laboratory groups,” *Expert Systems with Applications,* vol. 36, no. 3, pp. 7234-7241, Apr, 2009.

[71] M. A. Al-Betar, and A. T. Khader, “A harmony search algorithm for university course timetabling,” *Annals of Operations Research,* vol. 194, no. 1, pp. 3-31, 2012/04/01, 2012.

[72] S. M. Al-Yakoob, and H. D. Sherali, “Mathematical programming models and algorithms for a class-faculty assignment problem,” *European Journal of Operational Research,* vol. 173, no. 2, pp. 488-507, Sep, 2006.

[73] S. M. Al-Yakoob, and H. D. Sherali, “A mixed-integer programming approach to a class timetabling problem: A case study with gender policies and traffic considerations,” *European Journal of Operational Research,* vol. 180, no. 3, pp. 1028-1044, Aug, 2007.

[74] C. H. Aladag, and G. Hocaoglu, “A tabu search algorithm to solve a course timetabling problem,” *Hacettepe Journal of Mathematics and Statistics,* vol. 36, no. 1, pp. 53-64, 2007.

[75] C. H. Aladag, G. Hocaoglu, and M. A. Basaran, “The effect of neighborhood structures on tabu search algorithm in solving course timetabling problem,” *Expert Systems with Applications,* vol. 36, no. 10, pp. 12349-12356, Dec, 2009.

[76] R. Alvarez-Valdes, E. Crespo, and J. M. Tamarit, “Design and implementation of a course scheduling system using Tabu Search,” *European Journal of Operational Research,* vol. 137, no. 3, pp. 512-523, Mar, 2002.

[77] M. Amintoosi, and J. Haddadnia, "Feature selection in a fuzzy student sectioning algorithm," *Practice and Theory of Automated Timetabling V*, Lecture Notes in Computer Science E. Burke and M. Trick, eds., pp. 147-160, Berlin: Springer-Verlag Berlin, 2005.

[78] P. Avella, and I. Vasil'Ev, “A Computational Study of a Cutting Plane Algorithm for University Course Timetabling,” *Journal of Scheduling,* vol. 8, no. 6, pp. 497-514, 2005.

[79] R. P. Badoni, D. K. Gupta, and P. Mishra, “A new hybrid algorithm for university course timetabling problem using events based on groupings of students,” *Computers & Industrial Engineering,* vol. 78, pp. 12-25, Dec, 2014.

[80] R. B. Bai, J. Blazewicz, E. K. Burke, G. Kendall, and B. McCollum, “A simulated annealing hyper-heuristic methodology for flexible decision support,” *4or-a Quarterly Journal of Operations Research,* vol. 10, no. 1, pp. 43-66, Mar, 2012.

[81] K. R. Baker, M. J. Magazine, and G. G. Polak, “Optimal block design models for course timetabling,” *Operations Research Letters,* vol. 30, no. 1, pp. 1-8, Feb, 2002.

[82] M. A. Bakir, and C. Aksop, “A 0-1 integer programming approach to a university timetabling problem,” *Hacettepe Journal of Mathematics and Statistics,* vol. 37, no. 1, pp. 41-55, 2008.

[83] M. Banbara, T. Soh, N. Tamura, K. Inoue, and T. Schaub, “Answer set programming as a modeling language for course timetabling,” *Theory and Practice of Logic Programming,* vol. 13, pp. 783-798, Jul, 2013.

[84] R. Bellio, L. Di Gaspero, and A. Schaerf, “Design and statistical analysis of a hybrid local search algorithm for course timetabling,” *Journal of Scheduling,* vol. 15, no. 1, pp. 49-61, Feb, 2012.

[85] R. Bellio, S. Ceschia, L. Di Gaspero, A. Schaerf, and T. Urli, “Feature-based tuning of simulated annealing applied to the curriculum-based course timetabling problem,” *Computers & Operations Research,* vol. 65, pp. 83-92, Jan, 2016.

[86] C. Beyrouthy, E. K. Burke, D. Landa-Silva, B. McCollum, P. McMullan, and A. J. Parkes, “Towards improving the utilization of university teaching space,” *Journal of the Operational Research Society,* vol. 60, no. 1, pp. 130-143, Jan, 2009.

[87] A. L. Bolaji, A. T. Khader, M. A. Al-Betar, and M. A. Awadallah, “University course timetabling using hybridized artificial bee colony with hill climbing optimizer,” *Journal of Computational Science,* vol. 5, no. 5, pp. 809-818, Sep, 2014.

[88] A. Bonutti, F. D. Cesco, L. D. Gaspero, and A. Schaerf, “Benchmarking curriculum-based course timetabling: formulations, data formats, instances, validation, visualization, and results,” *Annals of Operations Research,* vol. 194, no. 1, pp. 59-70, 2012.

[89] E. K. Burke, G. Kendall, and E. Soubeiga, “A Tabu-Search Hyperheuristic for Timetabling and Rostering,” *Journal of Heuristics,* vol. 9, no. 6, pp. 451-470, 2003/12/01, 2003.

[90] E. K. Burke, J. Marecek, A. J. Parkes, and H. Rudova, “Decomposition, reformulation, and diving in university course timetabling,” *Computers & Operations Research,* vol. 37, no. 3, pp. 582-597, Mar, 2010.

[91] E. K. Burke, J. Marecek, A. J. Parkes, and H. Rudova, “A supernodal formulation of vertex colouring with applications in course timetabling,” *Annals of Operations Research,* vol. 179, no. 1, pp. 105-130, Sep, 2010.

[92] E. K. Burke, J. Mareček, A. J. Parkes, and H. Rudová, “A branch-and-cut procedure for the Udine Course Timetabling problem,” *Annals of Operations Research,* vol. 194, no. 1, pp. 71-87, 2012/04/01, 2012.

[93] V. Cacchiani, A. Caprara, R. Roberti, and P. Toth, “A new lower bound for curriculum-based course timetabling,” *Computers & Operations Research,* vol. 40, no. 10, pp. 2466-2477, Oct, 2013.

[94] H. Cambazard, B. O'Sullivan, and H. Simonis, “A Constraint-Based Dental School Timetabling System,” *Ai Magazine,* vol. 35, no. 1, pp. 53-63, Spr, 2014.

[95] M. P. Carrasco, and M. V. Pato, “A comparison of discrete and continuous neural network approaches to solve the class/teacher timetabling problem,” *European Journal of Operational Research,* vol. 153, no. 1, pp. 65-79, Feb, 2004.

[96] M. Chiarandini, M. Birattari, K. Socha, and O. Rossi-Doria, “An effective hybrid algorithm for university course timetabling,” *Journal of Scheduling,* vol. 9, no. 5, pp. 403-432, Oct, 2006.

[97] M. Chiarandini, L. Di Gaspero, S. Gualandi, and A. Schaerf, “The balanced academic curriculum problem revisited,” *Journal of Heuristics,* vol. 18, no. 1, pp. 119-148, Feb, 2012.

[98] S. Ceschia, L. Di Gaspero, and A. Schaerf, “Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem,” *Computers & Operations Research,* vol. 39, no. 7, pp. 1615-1624, Jul, 2012.

[99] A. Dammak, A. Elloumi, H. Kamoun, and J. A. Ferland, “Course Timetabling at a Tunisian University: A case study,” *Journal of Systems Science and Systems Engineering,* vol. 17, no. 3, pp. 334-352, Sep, 2008.

[100] S. Daskalaki, and T. Birbas, “Efficient solutions for a university timetabling problem through integer programming,” *European Journal of Operational Research,* vol. 160, no. 1, pp. 106-120, Jan, 2005.

[101] L. Di Gaspero, and A. Schaerf, "Multi-neighbourhood local search with application to course timetabling," *Practice and Theory of Automated Timetabling Iv*, Lecture Notes in Computer Science E. Burke and P. DeCausmaecker, eds., pp. 262-275, Berlin: Springer-Verlag Berlin, 2003.

[102] M. Dimopoulou, and P. Miliotis, “An automated university course timetabling system developed in a distributed environment: A case study,” *European Journal of Operational Research,* vol. 153, no. 1, pp. 136-147, Feb, 2004.

[103] C. W. Fong, H. Asmuni, B. McCollum, P. McMullan, and S. Omatu, “A new hybrid imperialist swarm-based optimization algorithm for university timetabling problems,” *Information Sciences,* vol. 283, pp. 1-21, Nov, 2014.

[104] C. W. Fong, H. Asmuni, and B. McCollum, “A Hybrid Swarm-Based Approach to University Timetabling,” *IEEE Transactions on Evolutionary Computation,* vol. 19, no. 6, pp. 870-884, Dec, 2015.

[105] M. J. Geiger, “Applying the threshold accepting metaheuristic to curriculum based course timetabling A contribution to the second international timetabling competition ITC 2007,” *Annals of Operations Research,* vol. 194, no. 1, pp. 189-202, Apr, 2012.

[106] A. Gunawan, K. M. Ng, and K. L. Poh, “A hybridized Lagrangian relaxation and simulated annealing method for the course timetabling problem,” *Computers & Operations Research,* vol. 39, no. 12, pp. 3074-3088, Dec, 2012.

[107] Y. He, S. Hui, and E.-K. Lai, “Automatic Timetabling Using Artificial Immune System,” *Lecture Notes in Computer Science,* vol. 3521, pp. 55-65, 2005/01/01, 2005.

[108] C. Head, and S. Shaban, “A heuristic approach to simultaneous course/student timetabling,” *Computers & Operations Research,* vol. 34, no. 4, pp. 919-933, Apr, 2007.

[109] G. Jaradat, M. Ayob, and Z. Ahmad, “On the performance of Scatter Search for post-enrolment course timetabling problems,” *Journal of Combinatorial Optimization,* vol. 27, no. 3, pp. 417-439, Apr, 2014.

[110] S. N. Jat, and S. Yang, “A guided search non-dominated sorting genetic algorithm for the multi-objective university course timetabling problem,” *Lecture Notes in Computer Science,* vol. 6622, pp. 1-13, 2011.

[111] D. Junaedi, and N. U. Maulidevi, “Solving Curriculum-Based Course Timetabling Problem with Artificial Bee Colony Algorithm,” *Proceedings of the First International Conference on Informatics and Computational Intelligence (ICI2011)*, pp. 112-117, 2011.

[112] M. Kalender, A. Kheiri, E. Ozcan, and E. K. Burke, “A greedy gradient-simulated annealing selection hyper-heuristic,” *Soft Computing,* vol. 17, no. 12, pp. 2279-2292, Dec, 2013.

[113] A. A. Kardan, H. Sadeghi, S. S. Ghidary, and M. R. F. Sani, “Prediction of student course selection in online higher education institutes using neural network,” *Computers & Education,* vol. 65, pp. 1-11, Jul, 2013.

[114] N. T. T. M. Khang, and T. T. H. Nuong, “The bees algorithm for a practical university timetabling problem in Vietnam,” *Proceedings of IEEE International Conference on Computer Science and Automation Engineering (CSAE),* vol. 4, pp. 42-47 2011.

[115] P. Kostuch, “The University Course Timetabling Problem with a Three-Phase Approach,” *Lecture Notes in Computer Science,* vol. 3616, pp. 109-125, 2005.

[116] P. Kostuch, and K. Socha, "Hardness prediction for the University Course Timetabling Problem," *Evolutionary Computation in Combinatorial Optimization, Proceedings*, Lecture Notes in Computer Science J. Gottlieb and G. R. Raidl, eds., pp. 135-144, Berlin: Springer-Verlag Berlin, 2004.

[117] G. Lach, and M. E. Lubbecke, “Curriculum based course timetabling: new solutions to Udine benchmark instances,” *Annals of Operations Research,* vol. 194, no. 1, pp. 255-272, Apr, 2012.

[118] J. Lee, S. P. Ma, L. F. Lai, N. L. Hsueh, and Y. Y. Fanjiang, “University timetabling through conceptual modeling,” *International Journal of Intelligent Systems,* vol. 20, no. 11, pp. 1137-1160, Nov, 2005.

[119] W. Legierski, "Search strategy for constraint-based class-teacher timetabling," *Practice and Theory of Automated Timetabling Iv*, Lecture Notes in Computer Science E. Burke and P. DeCausmaecker, eds., pp. 247-261, Berlin: Springer-Verlag Berlin, 2003.

[120] R. Lewis, and B. Paechter, "Application of the grouping genetic algorithm to University Course Timetabling," *Evolutionary Computation in Combinatorial Optimization, Proceedings*, Lecture Notes in Computer Science G. R. Raidl and J. Gottlieb, eds., pp. 144-153, Berlin: Springer-Verlag Berlin, 2005.

[121] R. Lewis, and B. Paechter, "Finding feasible timetables using group-based operators," *Ieee Transactions on Evolutionary Computation,* vol. 11, no. 3, pp. 397-413, Jun, 2007.

[122] R. Lewis, and J. Thompson, "Analysing the effects of solution space connectivity with an effective metaheuristic for the course timetabling problem," *European Journal of Operational Research,* vol. 240, no. 3, pp. 637-648, Feb, 2015.

[123] R. Lewis, B. Paechter, and B. McCollum, "Post Enrolment based Course Timetabling: A Description of the Problem Model used for Track Two of the Second International Timetabling Competition," 2007.

[124] Y. K. Liu, D. F. Zhang, and F. Y. L. Chin, "A clique-based algorithm for constructing feasible timetables," *Optimization Methods & Software,* vol. 26, no. 2, pp. 281-294, 2011.

[125] Z. P. Lu, J. K. Hao, and F. Glover, "Neighborhood analysis: a case study on curriculum-based course timetabling," *Journal of Heuristics,* vol. 17, no. 2, pp. 97-118, Apr, 2011.

[126] M. R. Malim, A. T. Khader, and A. Mustafa, "Artificial immune algorithms for university timetabling," *Proceedings of the 6th International Conference on Practice and Theory of Automated Timetabling (PATAT 2006),* pp. 234-245, 2006.

[127] A. L. Marquez, C. Gil, R. Banos, and J. Gomez, "Parallelism on multicore processors using Parallel.FX," *Advances in Engineering Software,* vol. 42, no. 5, pp. 259-265, May, 2011.

[128] B. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, A. J. Parkes, L. Di Gaspero, R. Qu, and E. K. Burke, "Setting the Research Agenda in Automated Timetabling: The Second International Timetabling Competition," *Informs Journal on Computing,* vol. 22, no. 1, pp. 120-130, Win, 2010.

[129] J. Miranda, P. A. Rey, and J. M. Robles, "udpSkeduler: A Web architecture based decision support system for course and classroom scheduling," *Decision Support Systems,* vol. 52, no. 2, pp. 505-513, Jan, 2012.

[130] S. A. MirHassani, "A computational approach to enhancing course timetabling with integer programming," *Applied Mathematics and Computation,* vol. 175, no. 1, pp. 814-822, 2006.

[131] T. Mueller, and H. Rudova, "Real-life curriculum-based timetabling with elective courses and course sections," *Annals of Operations Research,* vol. 239, no. 1, pp. 153-170, Apr, 2016.

[132] T. Muller, H. Rudova, and R. Bartak, "Minimal perturbation problem in course timetabling," *Practice and Theory of Automated Timetabling V*, Lecture Notes in Computer Science E. Burke and M. Trick, eds., pp. 126-146, Berlin: Springer-Verlag Berlin, 2005.

[133] C. Nothegger, A. Mayer, A. Chwatal, and G. R. Raidl, "Solving the post enrolment course timetabling problem by ant colony optimization," *Annals of Operations Research,* vol. 194, no. 1, pp. 325-339, Apr, 2012.

[134] A. H. Ozer, and C. Ozturan, “A direct barter model for course add/drop process,” *Discrete Applied Mathematics,* vol. 159, no. 8, pp. 812-825, Apr, 2011.

[135] V. Pereira, and H. G. Costa, “Linear Integer Model for the Course Timetabling Problem of a Faculty in Rio de Janeiro,” *Advances in Operations Research,* vol. 2016, Article ID 7597062, pp. 9 pages, 2016, 2016.

[136] A. E. Phillips, H. Waterer, M. Ehrgott, and D. M. Ryan, “Integer programming methods for large-scale practical classroom assignment problems,” *Computers & Operations Research,* vol. 53, pp. 42-53, Jan, 2015.

[137] S. Piechowiak, and C. Kolski, “Towards a generic object oriented decision support system for university timetabling: An interactive approach,” *International Journal of Information Technology & Decision Making,* vol. 3, no. 1, pp. 179-208, Mar, 2004.

[138] S. Piechowiak, J. X. Ma, and R. Mandiau, "An open interactive timetabling tool," *Practice and Theory of Automated Timetabling V*, Lecture Notes in Computer Science E. Burke and M. Trick, eds., pp. 34-50, Berlin: Springer-Verlag Berlin, 2005.

[139] D. Qaurooni, and M. R. Akbarzadeh-T, “Course timetabling using evolutionary operators,” *Applied Soft Computing,* vol. 13, no. 5, pp. 2504-2514, May, 2013.

[140] R. Qu, and E. K. Burke, “Hybridizations within a graph-based hyper-heuristic framework for university timetabling problems,” *Journal of the Operational Research Society,* vol. 60, no. 9, pp. 1273-1285, Sep, 2009.

[141] A. Qualizza, and P. Serafini, “A Column Generation Scheme for Faculty Timetabling,” *Lecture Notes in Computer Science,* vol. 3616, pp. 161-173, 2005/01/01, 2005.

[142] O. Rossi-Doria, C. Blum, J. Knowles, M. Samples, K. Socha, and B. Paechter, “A local search for the timetabling problem,” *Proceedings of the 4th International Conference on Practice and Theory of Automated Timetabling (PATAT 2002)*, pp. 124-127, 2002.

[143] H. Rudova, and K. Murray, "University course timetabling with soft constraints," *Practice and Theory of Automated Timetabling Iv*, Lecture Notes in Computer Science E. Burke and P. DeCausmaecker, eds., pp. 310-328, Berlin: Springer-Verlag Berlin, 2003.

[144] A. A. Salman, and S. Hamdan, “Differential evolution-based algorithm for solving the department's course-scheduling problem,” *Kuwait Journal of Science & Engineering,* vol. 39, no. 1B, pp. 175-209, Jun, 2012.

[145] R. Santiago-Mozos, S. Salcedo-Sanz, M. DePrado-Cumplido, and C. Bousono-Calzon, “A two-phase heuristic evolutionary algorithm for personalizing course timetables: a case study in a Spanish university,” *Computers & Operations Research,* vol. 32, no. 7, pp. 1761-1776, Jul, 2005.

[146] H. G. Santos, L. S. Ochi, and M. J. F. Souza, “A Tabu search heuristic with efficient diversification strategies for the class/teacher timetabling problem,” *Journal on Experimental Algorithmics,* vol. 10, pp. 2.9, 2005.

[147] H. G. Santos, E. Uchoa, L. S. Ochi, and N. Maculan, “Strong bounds with cut and column generation for class-teacher timetabling,” *Annals of Operations Research,* vol. 194, no. 1, pp. 399-412, Apr, 2012.

[148] S. C. Sarin, Y. Q. Wang, and A. Varadarajan, “A university-timetabling problem and its solution using Benders' partitioning-a case study,” *Journal of Scheduling,* vol. 13, no. 2, pp. 131-141, Apr, 2010.

[149] K. Schimmelpfeng, and S. Helber, “Application of a real-world university-course timetabling model solved by integer programming,” *Or Spectrum,* vol. 29, no. 4, pp. 783-803, Oct, 2007.

[150] S. T. Shih, C. Y. Chao, and C. M. Hsu, “An Effective and Efficient Class-Course-Faculty Timetabling Assignment for an Educational Institute,” *Life Science Journal-Acta Zhengzhou University Overseas Edition,* vol. 9, no. 1, pp. 47-55, 2012.

[151] S. Shimazaki, K. Sakakibara, and T. Matsumoto, “Iterative optimization techniques using man-machine interaction for university timetabling problems,” *Springerplus,* vol. 4, Jun 12, 2015.

[152] K. Socha, "The influence of run-time limits on choosing ant system parameters," *Genetic and Evolutionary Computation - Gecco 2003, Pt I, Proceedings*, Lecture Notes in Computer Science E. CantuPaz, J. A. Foster, K. Deb, L. D. Davis, R. Roy, U. M. Oreilly, H. G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. A. Dowsland, N. Jonoska and J. Miller, eds., pp. 49-60, Berlin: Springer-Verlag Berlin, 2003.

[153] J. A. Soria-Alcaraz, G. Ochoa, J. Swan, M. Carpio, H. Puga, and E. K. Burke, “Effective learning hyper-heuristics for the course timetabling problem,” *European Journal of Operational Research,* vol. 238, no. 1, pp. 77-86, Oct, 2014.

[154] J. A. Soria-Alcaraz, E. Ozcan, J. Swan, G. Kendall, and M. Carpio, “Iterated local search using an add and delete hyper-heuristic for university course timetabling,” *Applied Soft Computing,* vol. 40, pp. 581-593, Mar, 2016.

[155] C. Soza, R. L. Becerra, M. C. Riff, and C. A. C. Coello, “Solving timetabling problems using a cultural algorithm,” *Applied Soft Computing,* vol. 11, no. 1, pp. 337-344, Jan, 2011.

[156] J. Studenovsky, “Polynomial reduction of time-space scheduling to time scheduling,” *Discrete Applied Mathematics,* vol. 157, no. 7, pp. 1364-1378, Apr, 2009.

[157] V. Tam, J. Ho, and A. Kwan, "Applying an improved heuristic based optimiser to solve a set of challenging university timetabling problems: An experience report," *Pricai 2004: Trends in Artificial Intelligence, Proceedings*, Lecture Notes in Artificial Intelligence C. Zhang, H. W. Guesgen and W. K. Yeap, eds., pp. 164-172, Berlin: Springer-Verlag Berlin, 2004.

[158] T. Thepphakorn, P. Pongcharoen, and C. Hicks, “Modifying Regeneration Mutation and Hybridising Clonal Selection for Evolutionary Algorithms Based Timetabling Tool,” *Mathematical Problems in Engineering,* vol. 2015, pp. 16, 2015.

[159] G. M. Thompson, “Using information on unconstrained student demand to improve university course schedules,” *Journal of Operations Management,* vol. 23, no. 2, pp. 197-208, Feb, 2005.

[160] J. van den Broek, C. Hurkens, and G. Woeginger, “Timetabling problems at the TU Eindhoven,” *European Journal of Operational Research,* vol. 196, no. 3, pp. 877-885, Aug, 2009.

[161] J. J. J. van den Broek, and C. A. J. Hurkens, “An IP-based heuristic for the post enrolment course timetabling problem of the ITC2007,” *Annals of Operations Research,* vol. 194, no. 1, pp. 439-454, Apr, 2012.

[162] Y. Z. Wang, “Using genetic algorithm methods to solve course scheduling problems,” *Expert Systems with Applications,* vol. 25, no. 1, pp. 39-50, Jul, 2003.

[163] A. Wehrer, and J. Yellen, “The design and implementation of an interactive course-timetabling system,” *Annals of Operations Research,* vol. 218, no. 1, pp. 327-345, Jul, 2014.

[164] C. C. Wu, “Parallelizing a CLIPS-based course timetabling expert system,” *Expert Systems with Applications,* vol. 38, no. 6, pp. 7517-7525, Jun, 2011.

[165] L. Wu, “The application of Coarse-Grained Parallel Genetic Algorithm with Hadoop in University Intelligent Course-Timetabling System,” *International Journal of Emerging Technologies in Learning,* vol. 10, no. 8, pp. 11-15, 2015, 2015.

[166] S. X. Yang, and S. N. Jat, “Genetic Algorithms With Guided and Local Search Strategies for University Course Timetabling,” *Ieee Transactions on Systems Man and Cybernetics Part C-Applications and Reviews,* vol. 41, no. 1, pp. 93-106, Jan, 2011.

[167] A. E. Eiben, Z. Michalewicz, M. Schoenauer, and S. J. E., “Parameter Control in Evolutionary Algorithms,” *Studies in Computational Intelligence,* vol. 54, pp. 19-46, 2007.

[168] A. K. Qin, and P. N. Suganthan, “Self-adaptive Differential Evolution Algorithm for Numerical Optimization,” *Proceedings of The 2005 IEEE Congress on Evolutionary Computation,* vol. 2, pp. 1785-1791, 2005.

[169] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, “Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems,” *IEEE Transactions on Evolutionary Computation,* vol. 10, no. 6, pp. 646-657, 2006.

[170] M. S. Arumugam, and M. V. C. Rao, “On the improved performances of the particle swarm optimization algorithms with adaptive parameters, cross-over operators and root mean square (RMS) variants for computing optimal control of a class of hybrid systems,” *Applied Soft Computing,* vol. 8, no. 1, pp. 324-336, 2008.

[171] T. Stützle, M. López-Ibáñez, P. Pellegrini, M. Maur, M. Montes de Oca, M. Birattari, and M. Dorigo, "Parameter Adaptation in Ant Colony Optimization," *Autonomous Search*, Y. Hamadi, E. Monfroy and F. Saubion, eds., pp. 191-215: Springer Berlin Heidelberg, 2012.

[172] P. Avella, B. D'Auria, S. Salerno, and I. Vasil'ev, “A computational study of local search algorithms for Italian high-school timetabling,” *Journal of Heuristics,* vol. 13, no. 6, pp. 543-556, Dec, 2007.

[173] Y.-T. Kao, and E. Zahara, “A hybrid genetic algorithm and particle swarm optimization for multimodal functions,” *Applied Soft Computing,* vol. 8, no. 2, pp. 849-857, 2008.

[174] A. Layeb, “A novel quantum inspired cuckoo search for knapsack problems,” *International Journal of Bio-Inspired Computation,* vol. 3, no. 5, pp. 297-305, 2011.

[175] A. Layeb, and S. R. Boussalia, “A Novel Quantum Inspired Cuckoo Search Algorithm for Bin Packing Problem,” *International Journal of Information Technology and Computer Science (IJITCS),* vol. 4, no. 5, pp. 58-67, 2012.

[176] T. Thepphakorn, P. Pongcharoen, and C. Hicks, “An Ant Colony Based Timetabling Tool,” *International Journal of Production Economics,* vol. In Press, 2013.

[177] T. Lutuksin, and P. Pongcharoen, “Experimental design and analysis on parameter investigation and performance comparison of ant algorithms for course timetabling problem,” *Naresuan University Engineering Journal,* vol. 4, no. 1, pp. 31-38, 2009.

[178] J. K. Ousterhout, *Tcl and the tk toolkit*, 2 ed., Massachusetts: Addison-Wesley, 2009.

ภาคผนวก

Manuscript หรือ Proceeding ที่ได้รับการตีพิมพ์เผยแพร่จากโครงการวิจัยนี้จำนวน 2 บทความ

1. Thepphakorn, T. and Pongcharoen, P. (2019). *Performance Improvement Strategies on Cuckoo Search Algorithms for Solving University Course Timetabling Problem.* Submitted to **Expert Systems with Applications**. (ISI Q1) IF2018: 4.292

2. Thepphakorn, T. and Pongcharoen, P. (2019). *Variants and Parameters Investigations of Particle Swarm Optimisation for Solving Course Timetabling Problems*. **Lecture Notes in Computer Science**, 11655, pp. 177-187. (Scopus Q2)

Elsevier Editorial System(tm) for Expert Systems With Applications or its open access mirror
Manuscript Draft

Manuscript Number: ESWA-D-19-04418

Title: Performance Improvement Strategies on Cuckoo Search Algorithms for Solving University Course Timetabling Problem

Article Type: Full length article

Keywords: course timetabling; cuckoo search; Levy flight; experimental design; self adaptive; metaheuristics.

Corresponding Author: Dr. Pupong Pongcharoen, Ph.D.

Corresponding Author's Institution: Naresuan University

First Author: Thatchai Thepphakorn

Order of Authors: Thatchai Thepphakorn; Pupong Pongcharoen, Ph.D.

Dr. Binshan Lin
Editor-in-Chief
Expert Systems with Applications

September 17, 2019.

Dear Prof. Binshan Lin,

I would be most grateful if the attached manuscript entitled "**Performance Improvement Strategies on Cuckoo Search Algorithms for Solving University Course Timetabling Problem**" could be considered for publication in Expert System with Applications. The manuscript presents three strategies for improving the Cuckoo Search (CS) algorithm's performance applied to solve real-world course timetabling problems. A comprehensive literature survey on metaheuristics applied to solve course timetabling problems has been systematically conducted, summarised and discussed. No previous research has reported on the modifications and hybridisations of CS for solving real-world course timetabling problems. A novel Hybrid Self-adaptive Cuckoo Search based Timetabling (HSCST) program has been developed for minimising the total university operating costs. Sequential computational experiments were designed and conducted using eleven-size real-world timetabling problems. Our proposed methods statistically outperformed the original CS and Particle Swarm Optimisation both in term of solution's quality and their convergence speeds.

We confirm that this manuscript has not been published elsewhere and is not being considered by any other journals.

Thank you for your consideration of our manuscript. I look forward to hearing from you.

Yours sincerely,

Pupong Pongcharoen

Pupong Pongcharoen, Ph.D.

Director of the Research Centre for Operations Research and Industrial Applications (CORIA)
Faculty of Engineering,
Naresuan University,
Muang, Phitsanulok,
65000 Thailand
Email: pupongp@nu.ac.th
Telephone: + 66 55 964201

- Title Page –

Performance Improvement Strategies on Cuckoo Search Algorithms for Solving University Course Timetabling Problem

Thatchai Thepphakorn[1] and Pupong Pongcharoen[2*]

[1] Faculty of Industrial Technology, Pibulsongkram Rajabhat University,
Phitsanulok 65000, Thailand.
Email: thatchai.t@psru.ac.th

[2] Director of the Centre of Operations Research and Industrial Applications (CORIA),
Department of Industrial Engineering, Faculty of Engineering,
Naresuan University, Phitsanulok 65000, Thailand.
Emails: pupongp@nu.ac.th and pupongp@gmail.com

* Corresponding author's email addresses: pupongp@nu.ac.th and pupongp@gmail.com

Performance Improvement Strategies on Cuckoo Search Algorithms for Solving University Course Timetabling Problem

Thatchai Thepphakorn[1] and Pupong Pongcharoen[2*]

[1] Faculty of Industrial Technology, Pibulsongkram Rajabhat University, Phitsanulok 65000, Thailand
[2] Centre of Operations Research and Industrial Applications (CORIA), Department of Industrial Engineering, Faculty of Engineering, Naresuan University, Phitsanulok 65000, Thailand
*Corresponding e-mail address: pupongp@nu.ac.th and pupongp@gmail.com

Abstract: University course timetabling problem (UCTP) arises every academic year and is solved by academic staff with/without course timetabling tool. A feasible timetable must satisfy hard constraints. A Hybrid Self-adaptive Cuckoo Search based Timetabling (HSCST) tool has been developed for minimising the total university operating costs. The HSCST tool was applied to solve eleven-size course timetabling problems of Naresuan University. The performance improvements of Cuckoo Search (CS) algorithm embedded within the proposed tool were demonstrated using three strategies: parameter setting approaches (static and adaptive); movement strategies (Lévy Flights and Gaussian Random Walks); and local search hybridisation techniques. Sequential computational experiments were designed and conducted to investigate the efficiency of three proposed strategies. The statistical analysis on the computational results suggested that the proposed algorithms significantly outperformed the conventional algorithms by nine out of eleven (or 81.8%) datasets. The proposed methods also outperformed Particle Swarm Optimisation in term of solution quality and their convergence speeds.

Keywords: course timetabling; cuckoo search; Levy flight; experimental design; self adaptive; metaheuristics.

1. Introduction

Educational timetabling problem faced by schools, colleges, or universities is involved a set of events (e.g. courses or examinations) that must be appropriately assigned into a certain number of classrooms and timeslots subject to set of constraints (Burke et al., 2007). That problem arises every academic year and is solved by academic staff either with or without timetabling program (Thepphakorn et al., 2015). University course timetabling problem (UCTP) is one of the most challenging scheduling problems due to its complexity and constraints (Jat and Yang, 2011a). The UCTP is well known to be a non-deterministic polynomial (NP) hard problem, which means that the computational time required to find the solution increases exponentially with problem size (Pongcharoen et al., 2008). Automated timetabling tools are more desirable approach especially for solving very large problem size.

There has been a series of comprehensive literature surveys/review on Cuckoo Search (CS) (Abdel-Basset et al., 2018; Fister Jr et al., 2013; Mohamad et al., 2014; Shehab et al., 2017; Yang and Deb, 2014) and its modifications (Chiroma et al., 2017). Fister Jr et al. (2013) have surveyed 147 CS-related articles indexed in international academic databases (e.g. Google Scholar, Scopus and Web of Science). Mohamad et al. (2014) have reported a literature review of 35 articles related to the CS algorithm in terms of its applications and its performance comparison with other methods. Yang and Deb (2014) have reviewed 46 articles related to the fundamental idea of CS, its applications and search mechanisms. Shehab et al. (2017) have studied 194 articles and described comprehensive and exhaustive overview of the CS algorithm, its variants, application area and hybridisation. Chiroma et al. (2017) have

conducted a comprehensive literature survey on CS's relevant papers across eleven international academic databases (e.g. Scopus, IEEE Xplore, SpringerLink, etc.) and found on 76 articles. Abdel-Basset et al. (2018) have researched 115 articles and presented the Cuckoo Optimisation Algorithm main structure, variants, and their applications. Salgotra et al. (2018) have conducted a comprehensive literature survey and concluded that CS has focused only on its application in different domains and very limited work has been done to improve its performance. In this work, an update on the comprehensive literature review focused on the application of metaheuristics for solving course timetabling problems has been newly conducted and reported in the next section. According to the series of comprehensive reviews above, there is no report on the applications of CS considering movement strategies, parameter setting approaches, and its hybridisations for solving real-world university course timetabling problems.

Strategies for improving the metaheuristics performance can be classified as follows: (i) optimal parameter settings, e.g., static parameter approach for GA (Phuc et al., 2011), adaptive parameter approach in local search (Lindahl et al., 2018; Soria-Alcaraz et al., 2017a; Tarawneh and Ayob, 2013); (ii) movement strategies, e.g., random walk in CS (Teoh et al., 2014), low-level heuristic in neighbour search (Pillay and Özcan, 2017), neighbourhood in local search (Kiefer et al., 2017); and (iii) hybridisation, e.g., GA (Pillay and Özcan, 2017), Local Search (Nothegger et al., 2012). Elaborations of these strategies are sequentially presented in the following paragraphs.

According to the difference of problem domains, parameters play a significant role for the algorithm's performance (Chiroma et al., 2017). There are at least two approaches dealing with parameter setting: tuning or control (Talbi, 2009). For tuning approach (static parameter), the parameter values of the algorithms are fixed from the beginning of the computational run to the end (Eiben and Smit, 2011). For seeking the optimal parameter setting, the tuning approach is usually done by experimenting with different values and selecting the ones that give the best results on the test problems. Considering the number of parameters and its possible value settings, this approach is very time-consuming (Eiben et al., 2007). To overcome these difficulties, control approach allows the parameter values to change using adaptive control mechanism during the computational run (Eiben et al., 2007). However, a single work has been done on a particular problem domain to test which set of parameters gives the best performance of CS algorithm (Salgotra et al., 2018). However, there has been no report on the investigation of control approach for the CS parameters using the real-world university course timetabling problem.

Movement strategy has a direct impact on the balance of exploration and exploitation mechanisms (Yang, 2014). Large step-size movement strategy enhances the exploration mechanism whilst the exploitation mechanism would relatively prefer small step-size movement strategy. Step size is commonly achieved via random numbers drawn from distributions e.g. Uniform distribution, Gaussian or Normal distribution, Lévy distribution (Yang, 2014). For example, Lévy flight has been embedded within the CS algorithms to generate step size and to search the large-scale solution space effectively for standard benchmark problems (Salgotra et al. (2018). Zheng and Zhou (2012) have proposed that Gauss distribution in CS algorithm outperforms the conventional CS for solving standard test functions and engineering design optimisation problems. However, there has been no report on the comparative study of step-size movement strategies for solving university course timetabling problems.

Hybridisation strategy between two or more algorithms has been widely accepted in that it eliminates their limitations and increase their strengths for obtaining a better solution (Fong et al., 2015). CS has a simple structure (Salgotra et al., 2018) and a few number of parameters to be tuned (Chiroma et al., 2017; Khoja et al., 2018), but it is slow convergence (Rakhshani et al., 2016; Zhu and Wang, 2017), and poor balance between exploration and exploitation (Rakhshani and Rahati, 2017), and lacks of problem-specific knowledge (Mlakar et al., 2016). To overcome these weaknesses, the CS can be hybridised in order to: (i) improve its performances, i.e., faster convergence to the optimal solution over a shorter period of time (Chiroma et al., 2017); (ii) balance between exploration and exploitation of CS using local exploitation ability because CS is very efficient in global exploration ability (Salgotra et al.,

2018); and (iii) increase problem-specific knowledge to CS incorporating with local search approaches (Mlakar et al., 2016).

The objectives of this paper were to: (i) describe the development of Hybrid Self-adaptive Cuckoo Search based Timetabling (HSCST) tool for minimising total operating costs using eleven datasets obtained from the Faculty of Engineering, Naresuan University; (ii) conduct a comprehensive literature survey on the application of metaheuristics for solving course timetabling problems; (iii) demonstrate the use of experimental design and analysis (EDA) for identifying the appropriate settings of CS parameters; (iv) improve CS performance by implementing three strategies: parameter tuning and control; movement strategies (Lévy flight and Gaussian random walk); and hybridisation with Local Search; and (v) compare the performance of the proposed methods with other methods.

The next section of this paper briefly explains the CS's procedures, CS's terminology, CS's advantages and disadvantages, and a brief literature survey. Section III describes the university course timetabling based on the total operating costs followed by the architectural design of the HSCST tool. Section V presents the experimental design and analysis, comparative results before conclusions. The graphical outline of the article is given in Figure 1.

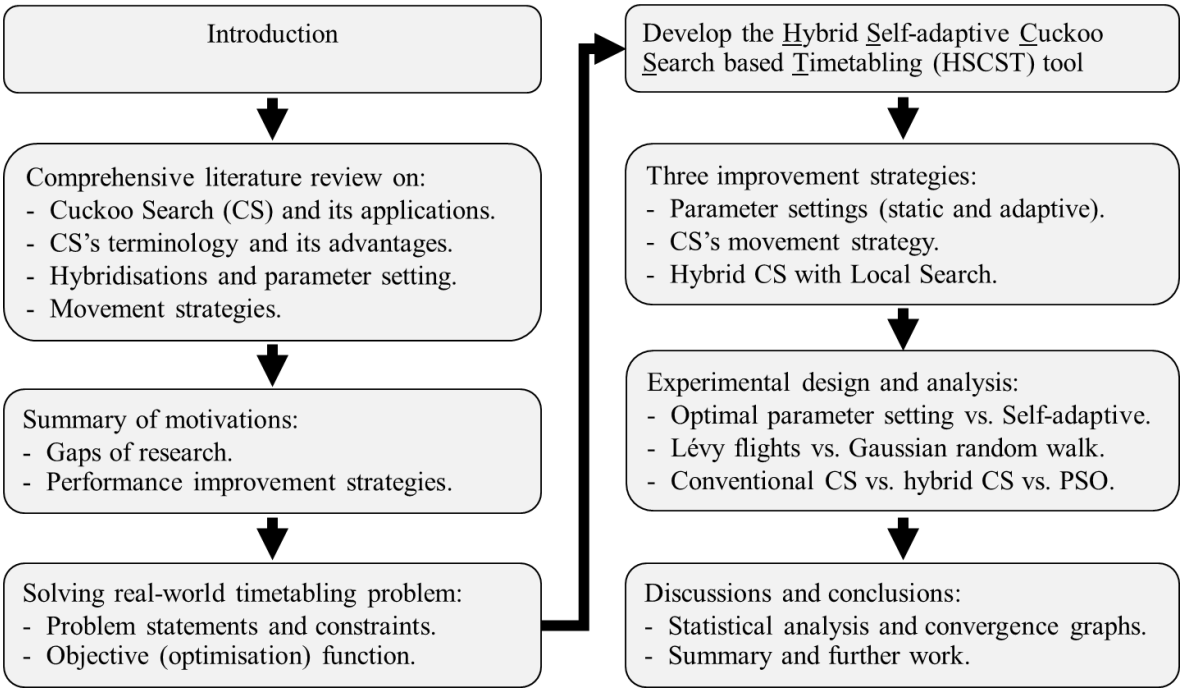


Figure 1 Graphical outline of this article

2. Cuckoo Search algorithm and literature survey

Metaheuristics are a class of approximation methods. They have been widely used to solve large-scale combinatorial optimisation problems within acceptable computational time, but they do not guarantee optimum solutions (Lewis, 2008). Metaheuristics have been investigated and found to be very successful in solving a variety of timetabling problems (Burke et al., 2007) such as Genetic Algorithm (GA) (Pongcharoen et al., 2008), Simulated Annealing (SA) (Kostuch, 2005), Tabu Search (TS) (Lü and Hao, 2010), Variable Neighbourhood Search (VNS) (Abdullah et al., 2005), Particle Swarm Optimisation (PSO) (Chen and Shih, 2013), Artificial Immune System (AIS) (He et al., 2005), Ant Colony Optimisation (ACO) (Thepphakorn et al., 2014), Harmony Search (HS) (Al-Betar and Khader, 2012), Artificial Bee Colony (ABC) algorithm (Junaedi and Maulidevi, 2011), and etc.

Cuckoo Search (CS) is one of the nature-inspired metaheuristic algorithms (Yang, 2010), inspired by the obligate brood parasitism of some cuckoo species that lay their eggs in the nests of other host birds (Li and Yin, 2015). Each egg in a host nest represents a solution, and a cuckoo egg represents a new solution (Yang, 2010). The aim is to create the new and potentially better solutions (cuckoo egg) to replace the worse solutions (host egg) in the host nests (Valian et al., 2013). There are mainly three principle rules for simple CS algorithm: (i) each cuckoo lays one egg at a time, and dumps its egg in a randomly chosen nest (Yang and Deb, 2013); (ii) the best nests with high quality eggs will be maintained to the next generation (Yang and Deb, 2013); and (iii) the number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability of alien egg discovery (P_a), in this case, the host bird can either get rid of the laid egg or abandon the nest and build a new nest (Yang and Deb, 2013). The pseudo code of conventional CS procedure based on these principle rules is shown in Figure 2.

Begin Objective function $F(x)$, $x = (x_1, x_2, \ldots, x_d)$
  Generate initial population x_i (i = 1, 2, …, P)
  While (t < Maximum Iteration: I) or (stop criterion)
    Get a cuckoo (say, x_i) randomly
    Generate a solution x_i by Lévy flights
    Evaluate quality/fitness of new solution x_i or $F(x_i)$
    Randomly select a nest within the population (say, x_j) and evaluate its fitness $F(x_j)$
    If $F(x_i) > F(x_j)$, then replace x_j by the new solution x_i
    If $rand < P_a$ then a current solution (x_i) is abandoned and replaced by a new solution
    Keep best solutions (or nest with quality solutions)
    Rank the solutions and find the current best solution
  End while
  Post-process results and visualisation
End

Figure 2 Pseudo code of the CS via Lévy flights (Yang and Deb, 2010)

In initial procedures, objective function $F(x)$ is specified. Then, each solution x_i (i = 1, 2, …, P) is generated randomly and evaluated for its quality or fitness before identifying the current best solution. When generating new solutions $x_i^{(t+1)}$ for a cuckoo i, the CS via Lévy flight (CSLF) is performed by using equation (1) (Yang and Deb, 2013).

$$x_i^{(t+1)} = x_i^{(t)} + \alpha \oplus Lévy\ (\beta) \tag{1}$$

Where $x_i^{(t)}$ is the current solution at iteration t. The product $\oplus$ means entry wise multiplication that is similar to those used in PSO (Yang, 2010). Where $\alpha > 0$ is the step size which is related to the scales of the problem of interest (Yang and Deb, 2013). In order to accommodate the difference between solution quality, the α can be defined as equation (2) (Yang and Deb, 2013).

$$\alpha = \alpha_0 (x_j^{(t)} - x_i^{(t)}) \tag{2}$$

Where α_0 is a constant value between 0.01-0.001 recommended by Yang (2010), whereas the term in the bracket corresponds to the randomly solution difference between x_j and x_i (Yang and Deb, 2013). The *Lévy* (β) is random walks, in which random step lengths are drawn from a Lévy distribution in equation (3) (Yang and Deb, 2013).

$$Lévy \sim u = t^{-1-\beta} \qquad (0 < \beta \leq 2) \tag{3}$$

The equation has an infinite variance with an infinite mean, in which the consecutive steps of a cuckoo essentially form a random walk process with obeys a power-law step-length distribution with a

heavy tail (Yang and Deb, 2013). Therefore, the generation of step size *s* samples can be summarised by equation (4) (Yang and Deb, 2013).

$$s = \alpha_0 (x_j^{(t)} - x_i^{(t)}) \oplus \textit{Lévy}\,(\beta) \sim 0.01 \frac{u}{|v|^{1/\beta}} (x_j^{(t)} - x_i^{(t)}) \tag{4}$$

Where *u* and *v* are drawn from Normal distributions following in equations (5) and (6) (Yang and Deb, 2013).

$$u \sim N(0, \sigma_u^2), v \sim N(0, \sigma_v^2) \tag{5}$$

$$\sigma_u = \left\{ \frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma[(1+\beta)/2]\beta 2^{(\beta-1)/2}} \right\}^{1/\beta}, \quad \sigma_v = 1 \tag{6}$$

Where Γ is the standard Gamma function (Yang and Deb, 2013). In case of CS via Gaussian random walks (CSGRW), a new solution $x_i^{(t+1)}$ generated from a cuckoo *i* is performed by using equation (7) (Yang and Deb, 2010).

$$x_i^{(t+1)} = x_i^{(t)} + \alpha \oplus \varepsilon_t \tag{7}$$

Where ε_t obeys a Gaussian distribution, this becomes a standard random walk (Yang and Deb, 2010). The α is defined as equation (2) (Yang and Deb, 2013). After preforming the Lévy flight or Gaussian random walks, the fitness value of new solution x_i or $F(x_i)$ is evaluated. A nest or solution among current population is randomly selected (called solution x_j) in order to compare the solution quality with a new solution x_i. The new solution x_i will be accepted and replaced to solution x_j if the $F(x_i)$ is better than the $F(x_i)$.

Next step, the worse nests (alien egg) will be detected and abandoned according to a probability $P_a \in [0, 1]$ (Yang and Deb, 2013). New nests will be built at new locations by using random permutation or random walks according to the similarity/difference to the host eggs (Yang and Deb, 2013). High probability of P_a increases the diversification mechanism whilst the low probability endorses the intensification mechanism (Li and Yin, 2015). After that, the population will be sorted according to the solution quality before identifying the best so far solution. These processes are repeated until getting to the maximum iteration (*I*) or stop criterion.

Terminology always plays an important role for scientists or professionals to express, communicate or transfer their knowledge and specialised texts for avoiding simple misunderstandings or errors. Terminology of CS compared with one of the most classical metaheuristic terminology, called Genetic Algorithm (GA), is shown in Table 1.

Table 1 Terminology comparison between GA and CS

General Terminology	Genetic Algorithm (GA)	Cuckoo Search (CS)
Decision variable	Gens in a Chromosome	Cuckoo eggs in a host nest
Solution	Chromosome	An egg in a host nest
Old solution	Parent Chromosome	An egg in a host nest
New solution	Child Chromosome	Cuckoo's egg laid in a host nest
Best solution	Chromosome having highest fitness value	The nest having highest quality egg
Fitness function	Survival fitness of a chromosome	Survival of cuckoo eggs in the host nest
Initial solution	Initial random chromosome	Initial random host nest
Selection	Roulette Wheel Selection	Ranking from the best to worst nests
Solution improving process	Crossover and Mutation Processes	Random Walks and generating a new nest
Intensification	Crossover Operation	Movement strategies or Random Walks
Diversification	Mutation Operation	Abandon the nest and build a new nest
Proportional selection	Probability of Crossover and Mutation	Probability of alien egg discovery

It can be seen that both GA and CS have something in common but different in terminology. For examples, a candidate solution in GA is called a chromosome whilst a cuckoo's egg is called in CS. GA has four parameters including population size, number of generations, probability of crossover, and probability of mutation (Vitayasak and Pongcharoen, 2018; Vitayasak et al., 2017). CS has three parameters including population size or host nests (P), maximum iteration (I), and probability of alien egg discovery (P_a) (Yang and Deb, 2010). According to Yang (2014), the population size and the probability of alien egg discovery were suggested at 15-40 and 0.25, respectively. The maximum iteration can be reasonably assigned according to the size of the problem considered as well as the availability of computational time and resources.

A review on advantages and disadvantages of both CS and GA is concisely summarised in Table 2. Both methods employ the population-based mechanism for performing multiple directional search (Thepphakorn et al., 2015; Yang, 2010).

Table 2 Advantages and disadvantages of GA and CS

Algorithms	**Advantages**	**Disadvantages**
Genetic Algorithm (GA)	- The performance and final result on time constraints and limited computer power (Karakatič and Podgorelec, 2015) - Population based and perform multiple directional search (Thepphakorn et al., 2015) - Enhance convergence because of crossover operation and elitism (Yang, 2014)	- Tuning the specific parameters (Vitayasak and Pongcharoen, 2018) - Premature convergence (Pandey et al., 2014) - Slow convergence rate and high computational cost (Patel et al., 2017)
Cuckoo Search (CS)	- It can solve both continuous and combinatorial problem domains (Yang et al., 2014). - It is a population-based algorithm (Yang, 2010). - It uses some sort of elitism and/or selection similar to that used in Harmony Search (Yang and Deb, 2010). - The randomisation in CS is more efficient as the step length is heavy-tailed, any large step is possible (Yang, 2010). - Few number of parameters to be tuned (Chiroma et al., 2017; Khoja et al., 2018) - It is potentially more generic to apply to more categories of optimisation problems (Yang, 2010). - Balance between local and global searching (Chiroma et al., 2017) - It has a simple structure (Salgotra et al., 2018)	- Slow convergence (Rakhshani et al., 2016; Zhu and Wang, 2017) - Low precision (Zhao and Niu, 2017; Zhu and Wang, 2017) - Premature convergence (Rakhshani and Rahati, 2017) - Poor balance between exploration and exploitation (Rakhshani and Rahati, 2017; Salgotra et al., 2018) - The performance of algorithm are based on difference parameter setting (Mlakar et al., 2016; Salgotra et al., 2018) - Lack of problem-specific knowledge (Mlakar et al., 2016)

A comprehensive literature survey on the applications of metaheuristics to solve the university course timetabling problem (UCTP) was conducted on the ISI Web of Science and Scopus databases covering the period from the past to September 2019. Using "course timetabl*" and "metaheuristic*" as keywords. The survey found 81 papers on the Scopus database and 34 papers on the ISI Web of Science database. However, some articles were duplicated; some were not research articles e.g. editorial messages; and some papers were not related with university timetabling. After screening and filtering, there were 75 papers that applied metaheuristics to solve UCTP as shown in Table 3. Various conventional metaheuristics have been adopted, e.g., Genetic Algorithm (GA), Memetic Algorithm (MA), Graph Heuristics, Ant Colony Optimisation (ACO), Artificial Bee Colony (ABC), Variable Neighbourhood Search (VNS), Simulated Annealing (SA), Harmony Search (HS), Tabu Search (TS), Scatter Search (SS), and etc.

Table 3 Comprehensive literature review on the applications of metaheuristics for solving university course timetabling problems

Authors	Methods	Course timetabling				Performance Improvement Strategies													In case of hybridisation (hybridised with)
						Parameter setting			Movement Strategies										
		Curriculum	Post enrolment	Benchmarking	Real-world data	Static (Tuning)	Adaptive	***Comparison***	Random walk (RW)	Lévy flight RW	Gaussian RW	Low-level heuristic	Mutation	Crossover	Neighbourhood	Egg Laying Radius	Pheromone	***Comparison***	
Mazlan et al. (2019)	Ant Colony Optimisation (ACO)				✓												✓		
Wahid et al. (2019)	Graph Heuristics	✓																	
Pillay and Özcan (2019)	Genetic Programming (GP), Hyper-heuristics, GA	✓																	
Jafarinejad et al. (2019)	Artificial Neural Networks (ANN)				✓														
Mauritsius et al. (2018)	Local Search (LS)		✓	✓											✓				
Matias et al. (2018)	Genetic Algorithms (GA)				✓								✓	✓					Guided Search
Junn et al. (2018)	Genetic Algorithms (GA)				✓								✓	✓					
Lindahl et al. (2018)	Neighbourhood Search	✓		✓			✓								✓				Integer Programming
Zhang et al. (2017)	Eco-geography Based Optimisation (EBO)				✓				✓										
Soria-Alcaraz et al. (2017b)	Hyper-heuristics, Neighbourhood Search		✓	✓								✓			✓				
Soria-Alcaraz et al. (2017a)	Hyper-heuristics, Iterated Local Search (ILS)		✓	✓		✓	✓					✓			✓				
Junn et al. (2017)	Great Deluge (GD) algorithm, SA				✓										✓				
Ortiz-Aguilar et al. (2017)	Iterated Local Search (ILS), GA			✓									✓	✓	✓				
Obit et al. (2017b)	GD, Constraint Programming (CP) Algorithm				✓										✓				GD, Constraint Programming (CP)
Obit et al. (2017a)	Hyper-heuristics			✓					✓			✓							Particle Swarm Optimisation (PSO)
Pillay and Özcan (2017)	Hyper-heuristics	✓		✓		✓						✓	✓	✓					GA, Genetic Programming (GP)
Kiefer et al. (2017)	Adaptive Large Neighbourhood Search	✓		✓		✓									✓				
Wahid and Hussin (2016b)	Graph Heuristics	✓		✓								✓							
Wahid and Hussin (2016a)	Graph Heuristics	✓		✓								✓							
Bellio et al. (2016)	Simulated Annealing (SA)	✓		✓		✓									✓				
Badoni and Gupta (2016)	Ant Colony Optimisation (ACO)		✓	✓		✓											✓		
Crawford et al. (2015)	Ant Colony Optimisation (ACO)				✓												✓		
Lewis and Thompson (2015)	Neighbourhood Search		✓	✓		✓									✓				
Fong et al. (2015)	Artificial Bee Colony (ABC)		✓	✓		✓									✓				Great Deluge (GD) Algorithm
Dun et al. (2014)	Simulated Annealing (SA)		✓										✓	✓	✓				Genetic Algorithms (GA)
Teoh et al. (2014)	GA, CS, Cuckoo Optimisation Algorithm (COA)				✓	✓			✓	✓			✓	✓		✓			
Soria-Alcaraz et al. (2014)	Hyper-heuristics, ILS		✓	✓		✓						✓			✓				
Jaradat et al. (2014)	Scatter Search (SS)		✓	✓								✓		✓	✓				
Bolaji et al. (2014)	Artificial Bee Colony (ABC)		✓	✓		✓									✓				Hill Climbing
Abuhamdah et al. (2014)	Gravitational Emulation Local Search (GELS)		✓	✓		✓									✓				LS, Collision and Descent Algorithm
Yassin et al. (2013)	Tabu Search (TS)		✓	✓		✓									✓				Great Deluge (GD) Algorithm
Tarawneh and Ayob (2013)	Simulated Annealing (SA)	✓		✓		✓	✓								✓				
Soria-Alcaraz Jorge et al. (2013)	Two-phase Algorithm			✓															
Soria-Alcaraz et al. (2013)	Two-phase Algorithm			✓															
Mansour and El-Jazzar (2013)	SA, Scatter Search (SS), Tuning Heuristic	✓		✓											✓				Local Search (LS)
Jaradat and Ayob (2013)	Big Bang-Big Crunch		✓	✓		✓					✓								
Bolaji et al. (2013)	Artificial Bee Colony (ABC)		✓	✓											✓				
Turabieh and El-Daoud (2012)	Enhanced Great Deluge Algorithm				✓										✓				

Authors	Methods	Course timetabling				Performance Improvement Strategies														In case of hybridisation (hybridised with)
						Parameter setting			Movement Strategies											
		Curriculum	Post enrolment	Benchmarking	Real-world data	Static (Tuning)	Adaptive	***Comparison***	Random walk (RW)	Lévy flight RW	Gaussian RW	Low-level heuristic	Mutation	Crossover	Neighbourhood	Egg Laying Radius	Pheromone	***Comparison***	
Ozcan et al. (2012)	Interleaved Constructive Memetic Algorithm				✓	✓							✓	✓						
Nothegger et al. (2012)	Ant Colony Optimisation (ACO)		✓	✓		✓											✓			LS, Simulated Annealing (SA)
Karami and Hasanzadeh (2012)	Genetic Algorithms (GA)		✓	✓									✓	✓						Hill Climbing
Alirezaei et al. (2012)	Particle Swarm Optimisation (PSO)				✓				✓											Local Search (LS)
Lewis (2012)	Neighbourhood Search		✓	✓											✓					
Geiger (2012)	Threshold Accepting Algorithm	✓		✓		✓						✓								
Ceschia et al. (2012)	Simulated Annealing (SA)		✓	✓		✓									✓					
Al-Betar et al. (2012)	Harmony Search (HS)		✓	✓											✓					Hill Climbing, PSO
Al-Betar and Khader (2012)	Harmony Search (HS)		✓	✓		✓									✓					
Abdullah et al. (2012)	Electromagnetic-like Mechanism	✓	✓	✓											✓					Great Deluge Algorithm, TS
Lu et al. (2011)	LS, ILS,TS, Adaptive Tabu Search	✓		✓											✓					
La'aro Bolaji et al. (2011)	Artificial Bee Colony (ABC)	✓		✓											✓					Neighbourhood Search
Jula and Naseri (2011)	Memetic Algorithm (MA)				✓								✓	✓						
Joudaki et al. (2011)	Memetic Algorithm (MA)		✓	✓									✓	✓						Simulated Annealing (SA)
Jat and Yang (2011b)	Genetic Algorithms (GA)		✓	✓		✓							✓	✓						Guided-search GA, TS
Jaradat and Ayob (2011)	Scatter Search		✓	✓								✓		✓	✓					Iterated Local Search (ILS)
Phuc et al. (2011)	Genetic Algorithms (GA)				✓	✓							✓	✓	✓					Bee Algorithm
Nguyen et al. (2011)	Variable Neighbourhood Search	✓			✓	✓									✓					
Khang et al. (2011)	Bee Algorithm				✓	✓									✓					
Budiono and Wong (2011)	Memetic Algorithm (MA)				✓	✓							✓	✓						
Jaradat and Ayob (2010)	Elitist Ant System (EAS)		✓	✓		✓											✓			Iterated Local Search (ILS)
Nguyen et al. (2010)	Tabu Search (TS)				✓										✓					
Dino Matijaš et al. (2010)	Ant Colony Optimisation (ACO)				✓	✓											✓			
Burke et al. (2010)	Neighbourhood Search	✓		✓		✓									✓					Integer Programming
De Causmaecker et al. (2009)	Local Search (LS)				✓	✓									✓					
Al-Betar et al. (2008)	Harmony Search (HS)		✓	✓																
Geiger (2008)	Local Search (LS)	✓		✓											✓					
Gunawan et al. (2008)	Simulated Annealing (SA), TS				✓										✓					Simulated Annealing (SA), TS
Mayer et al. (2008)	Ant Colony Optimisation (ACO)		✓	✓													✓			
Qarouni-Fard et al. (2008)	Particle Swarm Optimisation (PSO)		✓	✓					✓											
Lewis et al. (2007)	Grouping GA, SA, Heuristic Search		✓	✓		✓							✓	✓	✓					
Aladag and Hocaoglu (2007)	Tabu Search (TS)				✓										✓					
Chiarandini et al. (2006)	Local Search (LS)		✓	✓		✓									✓					SA, TS, Neighbourhood Search
Kostuch (2005)	Simulated Annealing (SA)		✓	✓											✓					
He et al. (2005)	CLONALG, Genetic Algorithms (GA)		✓	✓	✓	✓							✓	✓						
Burke et al. (2003)	Great Deluge (GD) Algorithm		✓	✓											✓					
Rossi-Doria et al. (2003)	Ant Colony Optimisation, GA, ILS, SA, TS		✓	✓		✓							✓	✓	✓		✓			
Socha et al. (2003)	Ant Colony System, Max-Min Ant System		✓	✓													✓			Local Search (LS)
This Work	CSLF, CSGRW, Adaptive Cuckoo Search				✓	✓	✓	✓	✓	✓	✓							✓		LS, Insert and Exchange Operators

From Table 3, there was only single published article from Teoh et al., (2014) that applied conventional Cuckoo Search (CS) with static parameter setting to solve UCTP. No comparative study on CS algorithm using either static or adaptive parameter setting approaches has been reported. There has been no report on the movement strategies and hybridisation of CS for solving UCTP. Moreover, many applications of metaheuristics to solve UCTP were considered and tested using the standard benchmarking datasets, e.g. Curriculum-based course timetabling problems (CB-CTP) (Geiger, 2012; Lu et al., 2011) and Post enrolment-based course timetabling problems (PE-CTP) (Lewis, 2012; Nothegger et al., 2012; Rossi-Doria et al., 2003). However, the considerations on the real-world course timetabling problems were marginal.

3. Real-world university course timetabling problem

In educational institutions, courses and examinations timetabling is a crucial activity, which assigns the appropriate timeslots for students, lecturers, and classrooms according to all constraints (Thepphakorn et al., 2014). Course timetabling constraints can generally be classified into two groups including hard constraints (*HCs*) and soft constraints (*SCs*) (Lewis, 2008). Hard constraints are the most important and must be satisfied to have a feasible timetable whereas soft constraints are more relaxed as some violations are acceptable, although the number of violations should be minimised (Thepphakorn et al., 2014).

There are two types of course timetabling datasets including benchmarking datasets and real-world datasets. The International Timetabling Competition in 2002 (ITC2002) and the International Timetabling Competition in 2007 (ITC2007) are popular benchmarking timetabling datasets, both of which have been regularly solved and reported (Geiger, 2012; Lewis, 2012; Lu et al., 2011; Nothegger et al., 2012). On another hand, many research works have also focused on the timetabling datasets obtained from the real-life course timetabling problems (Junn et al., 2017; Matias et al., 2018; Mazlan et al., 2019; Zhang et al., 2017). The major differences between the benchmarking and the real-world course timetabling problems are the additional complexity imposed by course structures, the variety of constraints, and the distributed responsibility for information needed to solve such problems at a university-wide level (Rudova et al., 2011).

Most universities in the real-world usually have their special set of constraints to practise (Lewis and Paechter, 2007). The complexity and difficulty to find a practical timetable become more relevance when dealing with a large number and variety of constraints related to the large numbers of students, lecturers, classrooms, and course structures as well as the special limitations arisen from university regulations and/or lecturers' requirements. For an example, the problem becomes more complex when students attend courses from multiple academic units and the solutions depend on the availability of students for the classes across multiple problems (Murray et al., 2006). Therefore, the complexity of the course timetabling problems appearing on the benchmarking or artificial datasets was usually decreased (Rudova et al., 2011). Some specific or local constraints will be cut off for more standard and easier comparison, in which the constraints applicable to the complex real-world timetabling problems can be reduced. Therefore, solving simplified problems or artificial datasets has rarely been extended to the solution of actual university problems of any large scale (Rudova et al., 2011).

In this work, the real-world course timetabling data obtained from Naresuan University (NU) was considered. The number of sub-course timetabling problems at NU generally depends on the number of schools/faculties and departments. NU course timetabling becomes more complex when students attend their courses arranged by different faculties or departments. NU course scheduling activities can be considered into three levels including the central or university level, faculty level, and department level. The central level is the most important level because it is related to many mandatory courses (such as General Education subjects, etc.), enrolled by a large number of the first or second year students across faculties or departments. All remaining courses are scheduled by faculties/departments staff.

NU's course structures are not only considered for individual events or courses, but they are also determined for the parts or details for each course. For examples: (i) a course may require either a lecture or laboratory or both, in which there may be similar or difference requirements such as available days and periods, classroom facilities, building locations, and etc.; (ii) mandatory courses for most of student programs in the first or second years across faculties or departments are very hard and complex to share a limited resources; (iii) a course having a number of students attending more than the size of available rooms (e.g. laboratory rooms) will be split into many sections, whereas the conflict constraints for lecturers responsible for multiple sections are increased.

There has been a number of constraints found in the NU course scheduling activities such as: (i) a course having multiple sections and teaching by the same lecturers must be assigned to be only one section at the same time; (ii) some courses having multiple lecturers must be considered, because many practical courses have high dangerous risks to students (e.g. industrial tools, chemical laboratory), the problem complexity in case of multiple lectures is based upon the number of the lecturers in charge; (iii) the specific requirements on a course taught by external or high-administrative-position lecturers must be obeyed; and (iv) all lectures of a course requiring special classrooms (such as drawing room, chemical laboratory, etc.) should be scheduled in double or triple consecutive periods. Therefore, all course timetabling constraints considered in this research can be described into the *HCs* and *SCs* as follows:

Hard constraints (*HCs*) considered were:

- HC_1 - all lectures/laboratories (elements) required for each course must be scheduled and assigned to distinct periods;
- HC_2 - students and lecturers can only attend one lecture at a time;
- HC_3 - only one lecture can take place in a room at a given time;
- HC_4 - lecturers and students must be available for a lecture to be scheduled;
- HC_5 - all courses must be assigned into the classrooms according to their given requirements including building location, room facilities, and room types; and
- HC_6 - all lectures within a course required consecutive periods must be obeyed.

In additions, soft constraints (*SCs*) considered were:

- SC_1 - all courses should be scheduled in the appropriate classroom in order to avoid unnecessary operating or renting costs (per hour);
- SC_2 - the courses taught by the given lecturer(s) should be assigned into their available or preferred day and periods in order to save the lecturing or hiring costs (per hour); and
- SC_3 - the classrooms should be scheduled in consecutive working periods of a day in order to reduce the number of times to clean or setup after using the rooms (per time).

HC_1–HC_6 determine whether potential solutions are feasible. HC_1–HC_3 are the fundamental timetabling constraints (called "event-clash" or binary constraint) that can be found in almost all university timetabling problems (Lewis, 2008). HC_4–HC_6 are individual requirements and timetabling policy found in the NU. SC_1–SC_3 are represented by an objective function in order to minimise the total university operating costs (Z) determined from the constructed course timetables. Moreover, *HCs* and *SC*s considered in this work can be formulated as a simple mathematical model shown in Eq. (8) - Eq. (10).

$$Min \;\; Z = \sum_{r\in R}\sum_{l\in L}\sum_{d\in D}\sum_{p\in Pe} x_{rldp}\cdot R_{rldp} + \sum_{t\in T}\sum_{s\in S}\sum_{d\in D}\sum_{p\in Pe} y_{tsdp}\cdot L_{tsdp} + \sum_{r\in R}\sum_{l\in L}\sum_{d\in D}\sum_{p\in Pe} x_{rldp}\cdot C_{rldp} \tag{8}$$

$$\text{Subject to: } HC_k = 0, \qquad \forall k, \tag{9}$$

$$x_{rldp},\; y_{tsdp} \in \{0,1\}, \qquad \forall r, \forall l, \forall d, \forall p, \forall t, \forall s \tag{10}$$

Eq. (8) is the objective function that evaluated the total university operating costs (currency unit: Thai Baht) consisting of three components. The first one determines the operating costs generated from classrooms used (SC_1). The next one considers the hiring cost calculated from lecturers (SC_2). The last one is to calculate the setup or cleaning costs after classrooms used (SC_3). Where *r*, *l*, *d*, *p*, *t*, and *s* are the index of classrooms, room types (such as lecture room, laboratory), days per week, periods per day, teachers, and curricula whereas *R*, *L*, *D*, *Pe*, *T* and *S* are the number of classrooms, room types, days per week, periods per day, teachers, and curricula, respectively. R_{rldp} is operating cost parameter for classroom *r* with *l* type on period *p* of day *d* (per hour). L_{tsdp} is hiring cost parameter for lecturer *t* taught student *s* on period *p* of day *d* (per hour). C_{rldp} is setup-cleaning cost parameter for classroom *r* having *l* types on period *p* of day *d* (per time). Eq. (9) checks a timetable to be a feasible timetable, in which all hard constraints must be satisfied. Where *k* is an index relating to the k^{th} hard constraint ($k = 1, 2, 3,\ldots, H$), where *H* is the number of hard constraints. The binary decision variables are shown in Eq. (10). Where x_{rldp} is to be 1 if classroom *r* having *l* types is used on period *p* of day *d*; otherwise 0. y_{tsdp} is to be 1 if lecturer *t* taught student *s* on period *p* of day *d*; otherwise 0.

4. Hybrid Self-adaptive Cuckoo Search based Timetabling (HSCST) Program

The Hybrid Self-adaptive Cuckoo Search based Timetabling (HSCST) tool has been coded in modular style with graphic user interface using a general purpose programming language called TCL/TK for front end with C extension for calculation in the back end. The program has been developed for solving course timetabling problems by using Cuckoo Search (CS) algorithm. Three alternatives strategies for improving CS performance, including self-adaptive parameter setting (SPS); movement strategy based on random walk variants; and hybridisation strategy, have been embedded. The proposed program can be implemented and graphically displayed the computational outputs on various platforms of computer operating system.

The procedures of the proposed program shown in Figure 3 can be divided into six main steps: (i) initialisation phase including data uploading, population initialisation and default parameter settings; (ii) strategic movement procedures for Cuckoo Search algorithm either based on Lévy flight or Gaussian random walks; (iii) repair process for rectifying infeasible timetables violated hard constraints; (iv) fitness evaluation and best solution updating procedure; (v) hybridisation strategy with neighbourhood search approaches (Insertion or Exchange Operators); and (vi) updating another CS parameter, which is the probability of alien egg discovery. This probability is used to replace the worse solutions by the new random-generated solutions. The following sub-sections describe these processes in more details.

4.1 Initialisation phase

Data encoding always plays an important role in computer programming. Each cuckoo egg in the nest represents a candidate solution (timetable) that comprises a set of elements. An element can be encoded using either numeric (binary, integer, or real) and/or alphanumeric characters (Pongcharoen et al., 2008). In this work, numeric element was applied and consisted positive and negative integer values. Positive integer values represent the elements/events of coded courses, each of which required a classroom, a day, and a timeslot. Negative value such as -1 was used to indicate empty events/timeslots to be assigned into a timetable. For an example shown in Figure 4, if there are three courses including *Drawing*, *Physics* and *Calculus*; each of which requires 4, 5, and 3 teaching timeslots per week, respectively. It can be seen that the *Drawing* course comprised four zero-coded elements according to the given coding indices of all courses. Likewise, *Physics* and *Calculus* subjects have four one-coded and three two-coded elements, respectively. Therefore, the number of coded elements/events is generated from the summation of lecturing time-periods required for all courses.

Begin Input university course timetabling data Generate a priority list of courses using heuristic orderings Set amount of search including numbers of Population (P) and Maximum Iteration (I) Create initial population of P solutions, x_i (i = 1, 2, 3, ..., P) Generate random keys for each x_i **If** (P_a is adaptive) **do** Initial values of P_a for each x_i	Step 1

While $t < I_{max}$ **do**
For (i = 1, $i <= P$, i++) **do** get a cuckoo (say, x_i) randomly
If (CSLF was selected) **do** generate a new solution x_i' by using Lévy flights
Else if (CSGRW was chosen) **do** generate a new solution using Gaussian random walks
End if
	Step 2

If (x_i' = infeasible timetable) **do**
Repair the x_i' to be a feasible timetable
End if
	Step 3

Evaluate its fitness $f(x_i')$ Choose a nest among n (say, x_j) randomly **If** $f(x_i') > f(x_j)$ **do** replace the x_j by the new solution x_i' **End for**	Step 4

For (i = 1, $i <= P$, i++) **do** get a cuckoo x_i
If (Insertion operator: IO) **do** produce a new solution x_i' using the IO
Else If (Exchange operator: EO) **do** produce a new solution x_i' using the EO
End if
If $f(x_i') > f(x_i)$ **do** replace x_i by a new solution x_i'
End for
	Step 5

If (P_a setting = SPS) **do** update values of P_a for each individual x_i
Rank all solutions (timetables) in current population
If ($rand < P_a$ of worse nests: x_{worse}) **do**
Build/generate new solutions x_{new}
Replace the x_{worse} by the new solution x_{new}
End if
Find the current best solution and keep the best so far solution
End while
Post-process on the results and its visualisation
End
	Step 6

Figure 3 Pseudo code of the HSCST program

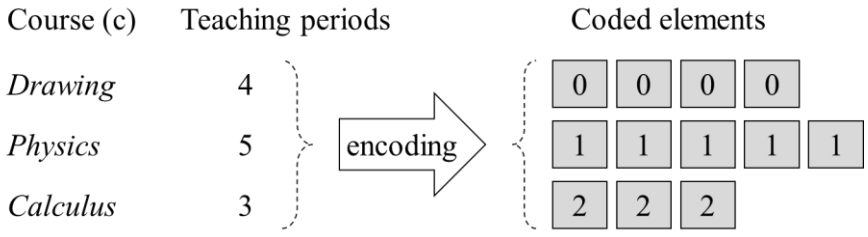


Figure 4 Element encoding for a solution

4.1.1 Population initialisation

Courses may have different priorities, the generation of infeasible solutions can be avoided by scheduling the highest priority activities first (Burke and Newall, 2004). This research, sorting the list of course priority called the Largest Unpermitted Period Degree first (LUPD) (Thepphakorn and Pongcharoen, 2013) is therefore adopted in order to ensure that all candidate solutions are feasible. Next

step is to create an empty solution (timetable) for assigning all coded elements. The length of that solution are calculated by taking into account the multiplications among the number of classrooms (*R*), days per week (*D*), and periods per day (*Pe*) shown in Figure 5.

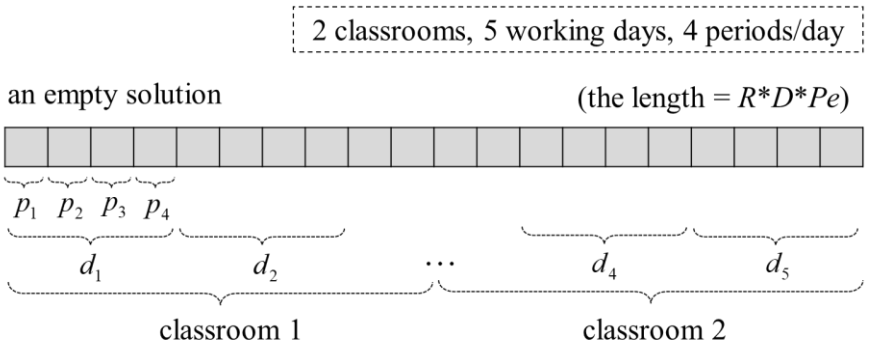


Figure 5 Solution representation on empty-slot timetable

After that, all encoded elements of courses are scheduled into an empty solution according to the list of course priority generated. The encoded elements of a course having the highest priority will be scheduled first. Before assigning each element/event into a timeslot, hard constraint checking is produced to ensure that the considering timeslots are feasible. Otherwise, the algorithm sequentially looks for the next empty timeslots of a solution that do not have any hard constraint violation. These processes are repeated until all courses were scheduled. For example, a completed timetable obtained from initialisation can be shown in Figure 6. It can be seen that all encoded elements/events belonging to *Physics*, *Drawing*, and *Calculus* subjects were completely scheduled, whereas the remaining empty timeslots of that solution were set to be the -1 value.

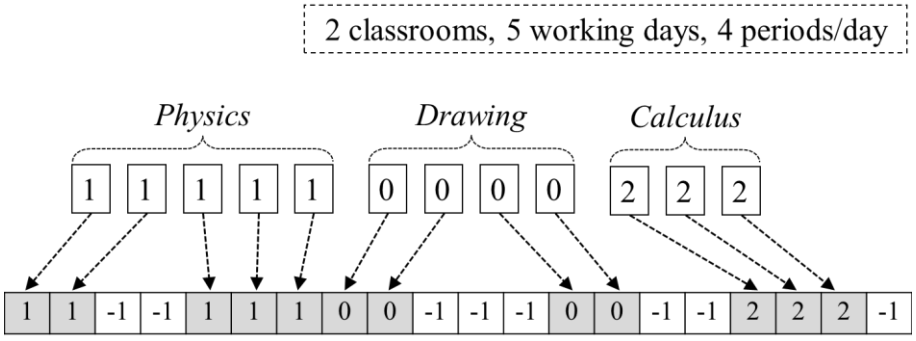


Figure 6 Initialisation of a candidate solution/timetable

4.1.2 Random keys initialisation

Random keys is a technique for adapting the movement strategy used in the continuous optimisation domains to discrete optimisation domains (Khadwilard et al., 2012). For course timetabling, there are four steps to generate random keys for each solution as shown in Figure 7: (i) create an empty-slot associated with all coded elements; (ii) randomly generate the values between 0 and 1 using Uniform distribution for each element; (iii) sort the random keys ascending; and (iv) sequentially assign the sorted random keys to the coded elements.

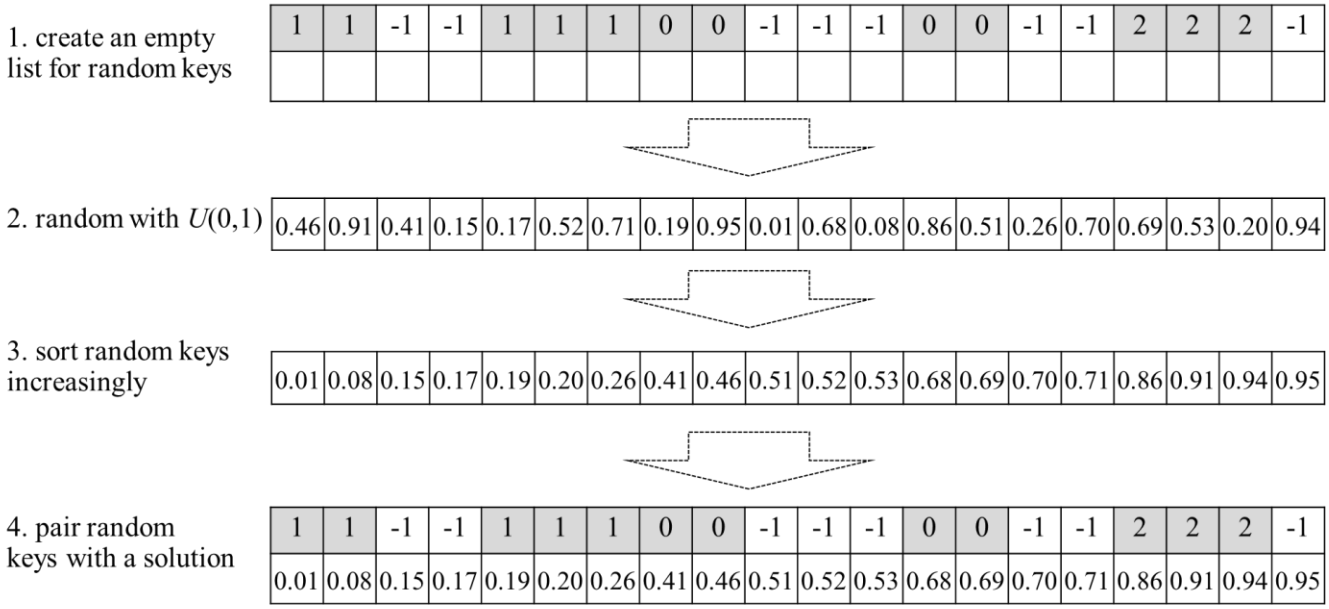


Figure 7 Random-keys generating steps

4.1.3 Adaptive parameter

Parameter setting always plays an important role on the balance between exploration and exploitation mechanisms conducted in metaheuristics. For Cuckoo Search, despite the suggestion on the setting of the probability of alien egg discovery (P_a) with a constant value of 0.25 (Yang, 2014), static parameter setting of 0.10 has been used by Teoh et al. (2014) for solving university timetabling problems. Adopting a constant value of P_a means that the static balance between exploration and exploitation mechanisms is indifferently used during iterative search process conducted within the method. Adaptive parameter setting has therefore been an alternative to employ dynamic balance as well as to avoid the premature convergence resulting in local optimal solutions. The parameter setting of P_a considered in this work is based on the iterative self-adaptive concept using the equation (11) or (12).

$$P_a(i) = 0.05 + 0.15 \times rand \tag{11}$$

$$P_a(i) = 0.85 + 0.05 \times rand \tag{12}$$

Where $P_a(i)$ is the P_a value for solution i whilst *rand* is random values between 0 and 1 based on Uniform distribution. The possibility intensity and slow down the search process will be increased by using equation (11) whilst the diversity and the convergence speed will be increased by using equation (12) (Li and Yin, 2015).

In the first step, an initial $P_a(i)$ value for each solution is generated by using the equation (11). Next step is to create two new integer variables for each solution i, called *success_ratio*(i) and *equation_selection*(i) in order to: (i) store the binary values of the success ratio (0 is unsuccessful, 1 is successful) from selected equations (11) or (12) in the previous periods; and (ii) keep the values of equation used, 1 is to use an equation (11) or 2 is to use an equation (12), respectively. However, the value of the variables is initially set to be 0 for all solutions. Both are crucial variables for self-adaptive parameter updating process, which is described in the next section 4.6.

4.2 Strategic movement procedures

In this research, two movement strategies so called Lévy flights (CSLF) and Gaussian random walks (CSGRW) were embedded within the CS. Both movement strategies can be described using equations (13) and (14);

$$x_i^{(t+1)} = x_i^{(t)} + \alpha_0 (x_{best}^{(t)} - x_i^{(t)}) \, stepsize \oplus randn() \tag{13}$$

$$stepsize = \frac{u}{|v|^{1/\beta}} \tag{14}$$

Where α_0 is a constant value whilst the term in the bracket corresponds to the randomly solution difference between $x_{best}^{(t)}$ and $x_i^{(t)}$. The *stepsize* is the step length in a random walks that can be fixed or varying (Yang, 2010). In case of CSLF, the *stepsize* is a varying random walk that obeyed the Lévy distribution via the Mantegna's algorithm shown in equation (14) (Yang, 2010) whilst the *stepsize* for CSGRW in this work is a fixed random walk and set to be 1. The *randn*() is random values generated from standard normal or Gaussian distribution with 0 mean and 1 standard deviation or $N(0,1)$.

The movement procedure of each element in a solution $x_i^{(t)}$ for CSLF and CSGRW are shown in Figure 8. For example, if the first event of subject number 1 (the third element position) in the $x_i^{(t)}$ solution is firstly selected to move. The first step is to find the position of the first event of subject number 1 (the first element position) in the $x_{best}^{(t)}$ solution. Next step, random key values for that event of subject number 1 belonging to $x_{best}^{(t)}$ and $x_i^{(t)}$ solutions are collected. The third step, a new random key value is calculated by using equation (13) before replacing it into the same position of subject number 1 in the $x_i^{(t)}$ solution. The fourth step is to select the next sequential event in the $x_i^{(t)}$ solution for movement. These four steps are repeated until new random key values for all events in the $x_i^{(t)}$ solution have been updated. The fifth step is to sort the new random keys in ascending order and move all elements of $x_i^{(t)}$ solution to new position according to the ascending order of new random keys. It can be seen that the first event of subject number 1 is moved from the third position of the $x_i^{(t)}$ solution to the second position of the $x_i^{(t+1)}$ solution instead (see in Figure 8).

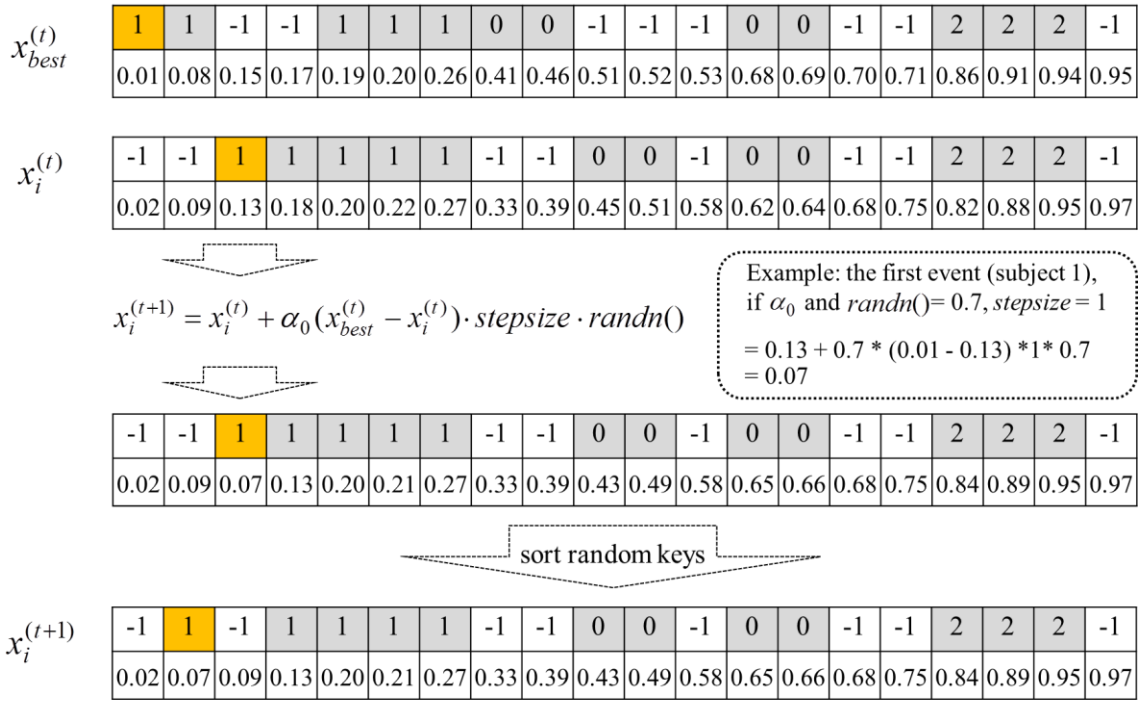


Figure 8 Example of movement procedures for CSLF and CSGRW

4.3 Repair process

After performing the movement procedure, new solutions obtained may be either feasible or infeasible solutions (due to hard constraint violations). In this work, the repair process is introduced for rectifying infeasible solutions. For an example of repair processes shown in Figure 9, *Drawing* subject (element number 0) is required to schedule in double consecutive periods at the drawing room. The first step is to find the possible empty timeslots having no hard constraint (HC) violation in drawing room. If double empty timeslots without any violation of HCs are found, two elements of number 0 are firstly move to that timeslots. Otherwise, a possible empty timeslot (-1 value) in drawing room is randomly selected for swapping with element of number 0. If those elements are not assigned in consecutive periods (see position numbers 3 and 5 in Figure 9), the second step is to the possible empty timeslots near position numbers 3 and 5 (including position numbers 2, 4, and 6). If element number 0 at position 3 can be swapped with the element number 2 at position 4 without any violation of HCs, both positions are then accepted to swap together. These processes are repeated until all coded courses have been satisfied for all hard constraints (HCs).

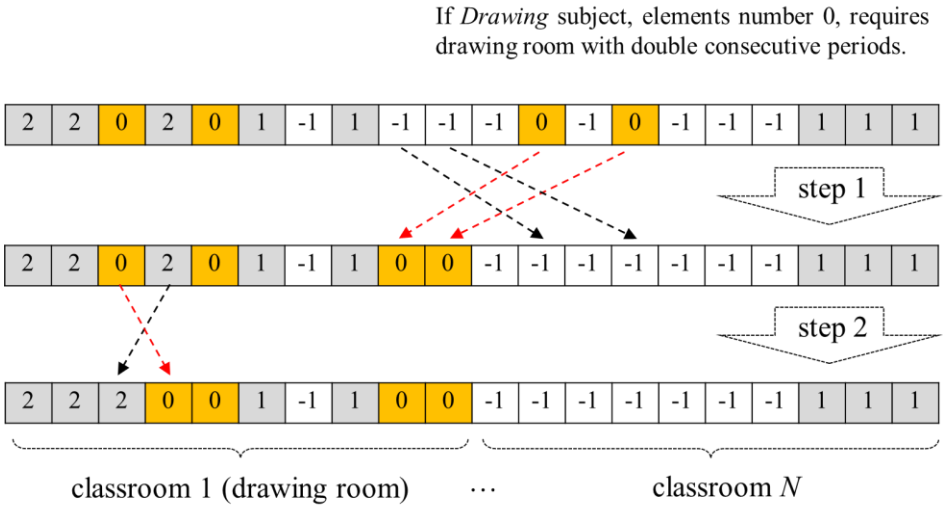


Figure 9 Example of repair process embedded within the HSCST tool

4.4 Fitness evaluation and best solution updating

Once the timetables are complied with hard constraints, the timetables are evaluated for its quality by using the objective function shown in Eq. (8) for calculating the total university operating costs (Z). Due to minimisation problem, a solution or timetable having the lower Z value is more preferable than that having the higher Z value.

4.5 Hybridisation strategy

There are many advantages on conducting hybridisation strategy: (i) improve the solution quality, through increased exploitation search; and (ii) increase the opportunity to quickly discover the global best solution. However, increasing more search usually requires more computational time and resources. In this work, two local search strategies, so called Insertion Operator (IO) and Exchange Operator (EO), were embedded for hybridisation with the Cuckoo Search. Both operators had been demonstrated on its ability to improve the solution quality (Talbi, 2009). The main steps of the IO and EO procedures are illustrated in Figure 10 and Figure 11, respectively. First step starts from specifying two positions within a solution randomly (if the random number of position A is greater than position B). In the case of IO, an

element at position A will be moved to position B whilst the remaining elements between position A-1 and position B will be moved one position to the right hand (see Figure 10). For EO, the element values located at position A and B are swopped (see Figure 11).

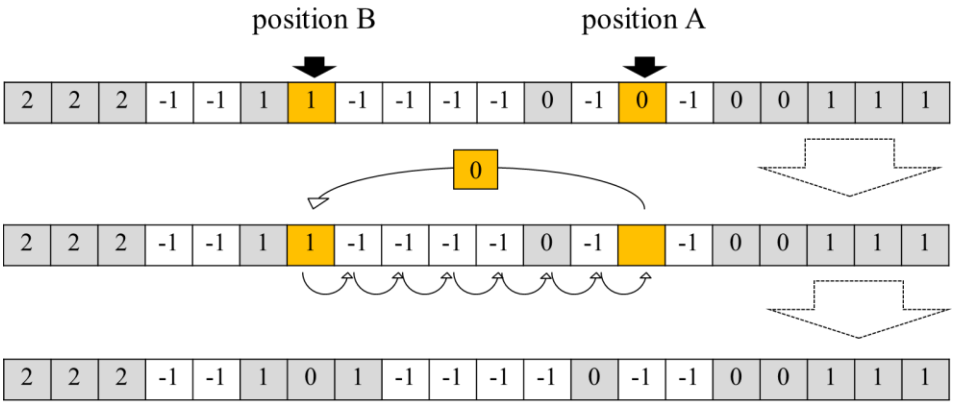


Figure 10 Steps of insertion operator (IO) (Talbi, 2009)

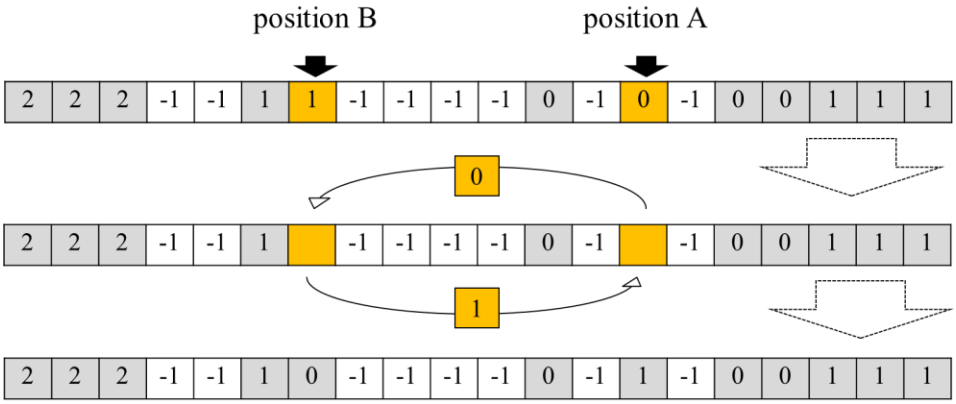


Figure 11 Steps of exchange operator (EO) (Talbi, 2009)

After conducting LS procedures, the repair process is applied if infeasible solution has been found and followed by measuring its fitness value. If fitness value associated with the improved solution is better than the fitness value associated with the old solution, the improved solution is accepted and therefore replaced the old solution. These steps are similarly repeated for the remaining solutions.

4.6 Updating CS's parameter

For updating the CS's parameter in this work, two sequential procedures (which were self-adaptive parameter update and population abandon) are proposed and described in the following sub-sections.

4.6.1 Self-adaptive parameter update

The procedures of self-adaptive P_a setting are described in a pseudo code shown in Figure 12. After fitness measurement, if the solution i using equation (11) (*equation_selection*(i) = 1) for P_a setting has the solution quality better than that in the previous iteration, the *success_ratio*(i) will set to be 1, otherwise it will set be 0. In the same way, the *success_ratio*(i) for solution i using equation (12) (*equation_selection*(i) = 2) will be also updated. In the end of the every iteration, the summations of

success_ratio(*i*) for each selected equations will be calculated and kept at the success1 and success2 variables, respectively.

If the success1 is larger than the success2, it denotes that the parameter P_a obtained from adopting equation (11) perform better than that generated by equation (12). Therefore, the probability (*prob*) of selected equation (11) for P_a setting must be increased. Otherwise, it means that the P_a value obtained from adopting equation (12) is very efficient, in which the *prob* value of selected equation (11) must be reduced. These steps are repeated until all P_a values associated with all solutions are updated.

```
Update success_ratio(i) variable
Calculate success1 and success2 according to success_ratio(i) variable
For (i=1, i <= P, i++) do
    If (iteration = 1) do
        If (rand < 0.5) do
            If (rand <= 0.05) do update Pa(i) using equation (11) and set equation_selection(i) = 1;
        Else
            If (rand <= 0.05) do update Pa(i) using equation (12) and set equation_selection(i) = 2;
        End if
    Else
        If (success1 > success2) do prob = 0.7 + 0.3*rand; else prob = 0.4 - 0.3*rand;
        If (rand < prob) do
            If (rand <= 0.5) do update Pa(i) using equation (11) and set equation_selection(i) = 1;
        Else
            If (rand <= 0.5) do update Pa(i) using equation (12) and set equation_selection(i) = 2;
        End if
    End if
End for
```

Figure 12 Pseudo code of the self-adaptive P_a setting for CS (Li and Yin, 2015).

4.6.2 Population abandon

All solutions (timetables) in the population are decreasingly sorted by determining their fitness values. The solution with the highest fitness is assigned to be the highest ranking or $i = 1$, ($i = 1, 2, 3,\ldots, P$), whereas the solution with the lowest fitness is ranked last ($i = P$). The last-rank solution (x_{worse}) is determined to be abandoned or kept according to its probability $P_a(P) \in [0, 1]$. If a random value based on the Uniform distribution (*rand*) is more than or equal to $P_a(P)$, the x_{worse} solution is kept. Otherwise, the x_{worse} solution will be improved by using the standard random walks in equation (7) (Yang and Deb, 2010). After that, the repair process is applied if the new solution (x_{new}) is infeasible timetable. This follows by measuring the total operating costs (Z) and its fitness value. At this point of the CS method, the iteration is completed and the best-so-far solution is identified and kept. The CS process from step 2 to step 6 shown in Figure 3 are repeated until the maximum iteration (I) is satisfied. Finally, the best-so-far solution and its fitness value are reported.

5. Experimental design and analysis

The computational experiments were aimed to: (i) identify the main factors and their interactions that were statistically significant before concluding the appropriate parameter settings for the CS method; (ii) explore the performance of the CS via Gaussian random walks and the CS via Lévy flights; (iii) investigate the performances of CS using the optimal parameter settings via parameter tuning and parameter control techniques; and (iv) compare the performance of the proposed CS hybridising with the Local Search including Insertion Operator (IO) and Exchange Operator (EO) as well as Particle Swarm

Optimisation (PSO). The characteristics of the eleven real-world course timetabling datasets obtained from the Faculty of Engineering, Naresuan University (NU) are shown in Table 4 (Thepphakorn et al., 2016). The timetables generated by the HSCST program were measured by summation the total university operating costs on the soft constraints mentioned in section 3. Personal computers with Core i7 3.4 GHz CPU and 4 GB RAM was used to determine the simulation time required to execute a computational run.

Table 4 Characteristics of the proposed NU course timetabling problems

Problems	**Characteristics of the NU course timetabling problems**						
	No. Courses	No. Events	No. Classrooms	No. Days/ week	No. Periods/ day	No. Lecturers	No. Curricula
1	56	173	53	5	10	30	19
2	103	323	77	7	10	62	36
3	123	353	86	7	10	49	27
4	124	380	74	7	11	56	35
5	144	452	91	7	10	78	43
6	162	486	99	7	10	71	34
7	163	499	88	7	11	72	38
8	204	639	114	7	10	96	52
9	208	647	99	7	11	102	56
10	221	687	108	7	12	94	44
11	323	1,009	142	7	13	143	66

A. CS's optimal parameter investigations

The first experiment was aimed to demonstrate the use of advance statistical tools called experimental design and analysis (EDA) for investigating the appropriate setting of CS's parameters. The factors included (i) the combination of population sizes (host nests) and the maximum iteration (*PI*), which determines the total number of solutions generated, the amount of search and the execution time required. In this computational experiment, the value was fixed at 24,000 to limit the amounts of search and time taken for the computational search; and (ii) the probability of alien egg discovery (P_a). Yang (2010) also suggested the sufficient values of the population sizes and the P_a parameters for most optimisation problems should be set between 15 and 40, and 0.25, respectively. Table 5 summarises the factors and its level considered.

Table 5 Experimental factors and its levels

Factors	**Levels**	**Values**		
		Low (-1)	*Medium (0)*	*High (+1)*
PI	3	15*1600	25*960	40*600
P_a	3	0.1	0.25	0.4

Due to few parameter numbers, full factorial experiment based on the 3^2 design (Montgomery, 2012) was adopted in this experiment for considering all the combinations of the factors in each replication. The computational experiment was based on eleven problems (shown in Table 4), each of which was repeated thirty times using different random seed numbers. The computational results obtained from 270 (3^2*30) runs per problem were analysed using a general linear model form of analysis of variance (ANOVA). After conducting ANOVA tables for all proposed problems, the 1st, 5th, and 11th problems related to the small, medium, and large sizes were selected for examples to show the results of

ANOVA (shown in Table 6). The ANOVA table consisted of Source of Variation (*Source*), Degrees of Freedom (*DF*), Sum of Square (*SS*), Mean Square (*MS*), *F* value, and *P* value. A factor with value of $P \leq 0.05$ was considered statistically significant with 95% confidence interval. From Table 6, *PI* factor was statistically significant in term of main effect in the 1st and 11th problems.

Table 6 ANOVA on the CS's parameters for selected problems

Problems	*Source*	*DF*	*SS*	*MS*	*F*	*P*
1	*PI*	2	576,316	288,158	43.89	**0.000**
	P_a	2	30,338	15,169	2.31	0.102
	$PI*P_a$	4	32,133	8,033	1.22	0.302
	Seeds	29	247,749	8,543	1.30	0.148
	Error	232	1,523,225	6,566		
	Total	269	2,409,761			
5	*PI*	2	1,101,544	550,772	1.38	0.253
	P_a	2	1,424,859	712,429	1.79	0.169
	$PI*P_a$	4	305,787	76,447	0.19	0.942
	Seeds	29	11,690,452	403,119	1.01	0.452
	Error	232	92,302,925	397,857		
	Total	269	106,825,567			
11	*PI*	2	148,002,875	74,001,438	4.68	**0.010**
	P_a	2	47,204,008	23,602,004	1.49	0.227
	$PI*P_a$	4	119,012,484	29,753,121	1.88	0.115
	Seeds	29	531,110,852	18,314,167	1.16	0.272
	Error	232	3,669,965,471	15,818,817		
	Total	269	4,515,295,691			

In this work, the appropriate parameter settings of the CS's factors can be determined from the lowest points in the main effect plots. For example, Figure 13 to Figure 15 suggested that the CS's factors including *PI* and P_a for the 1st, 5th, and 11th problems should be defined at 15*1,600 and 0.25, 25*960 and 0.1, and 25*960 and 0.4, respectively. Moreover, the best parameter settings of the CS for remaining problems are also concluded and shown in Table 7. There is no generic optimal parameter set that can be efficiently applied to every problem sizes and domains due to the problem specific and the nature of the algorithms (Figlali et al., 2009).

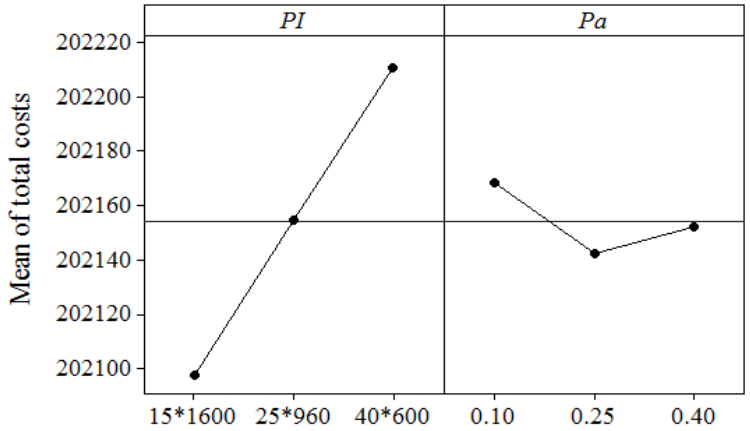


Figure 13 Main effect plots of *PI* and P_a for the 1st problem

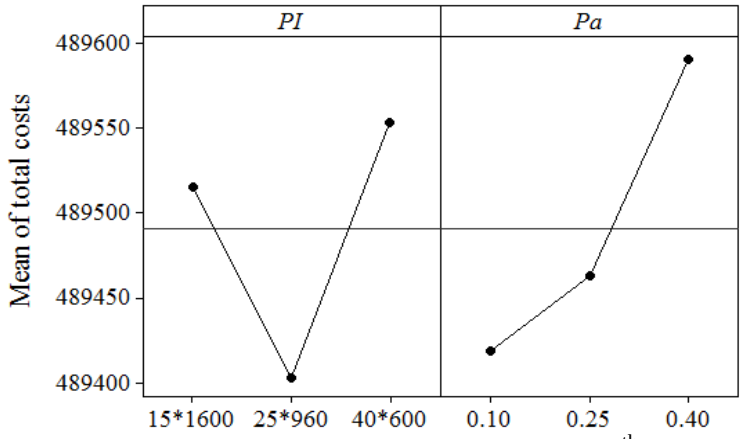


Figure 14 Main effect plots of *PI* and P_a for the 5th problem

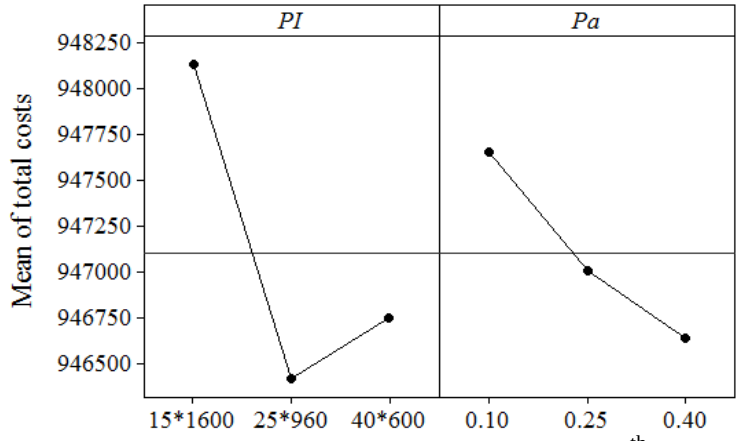


Figure 15 Main effect plots of *PI* and P_a for the 11th problem

Table 7 Appropriate parameter settings of the CS for eleven problems

Problems	**Appropriate parameter settings of the CS**	
	Population size x **Maximum Iteration (*PI*)**	**Probability of alien egg discovery (P_a)**
1	15*1,600	0.25
2	15*1,600	0.10
3	25*960	0.40
4	15*1,600	0.25
5	25*960	0.10
6	25*960	0.25
7	40*600	0.10
8	40*600	0.40
9	40*600	0.25
10	25*960	0.40
11	25*960	0.40

B. Perfomances of CS's random walks

This experiment was designed to compare the performance between CS using Gaussian random walks (CSGRW) and CS using Lévy flights (CSLF) to find the best so far timetables. The appropriate parameter setting of the CS was adopted from the first experiment based on the experimental design and analysis. The performance comparisons for both methods to solve the proposed datasets (detailed in Table 4) were analysed in terms of the minimum, maximum, average, standard deviation (*SD*), and execution time (*Time*: minutes unit) required to find the best so far solutions obtained. The *T* value obtained from the *t*-test method and the *P* value are also shown in Table 8. The amount of search for each method was equally fixed at 24,000 solutions to solve the proposed problems, each of which was also repeated thirty times using different random seed numbers.

Table 8 Computational results obtained from the CSGRW and the CSLF

Prob.	**Methods**	**Best so far solutions (total operating costs)**					***t*-test**	
		Minimum	*Maximum*	*Average*	*SD*	*Time* (*minutes*)	*T* values	*P* values
1	CSGRW	202,054.50	202,272.50	202,147.83	54.10	2.44	3.11	**0.003**
	CSLF	**201,998.00**	**202,225.00**	**202,103.62**	56.02	2.53		
2	CSGRW	378,485.50	379,806.00	379,039.18	341.62	10.05	1.26	0.211
	CSLF	**377,927.25**	**379,754.50**	**378,919.28**	390.96	9.86		
3	CSGRW	**301,028.25**	305,228.75	303,160.19	1,052.36	21.79	0.06	0.956
	CSLF	301,125.25	**305,010.75**	**303,146.30**	866.33	24.49		
4	CSGRW	**299,674.50**	308,693.75	304,171.83	2,322.36	12.16	2.05	**0.045**
	CSLF	300,210.00	**307,450.25**	**302,937.73**	2,337.29	13.25		
5	CSGRW	**488,054.75**	490,183.00	489,362.45	534.46	25.99	0.62	0.540
	CSLF	488,141.50	**490,141.00**	**489,286.09**	416.09	28.24		
6	CSGRW	**406,083.25**	410,164.75	407,872.34	910.48	30.06	0.46	0.645
	CSLF	406,149.75	**409,963.50**	**407,758.56**	991.47	35.00		
7	CSGRW	412,589.00	418,821.25	416,126.10	1,488.51	21.85	2.96	**0.005**
	CSLF	**412,089.00**	**418,062.50**	**414,820.20**	1,901.27	23.14		
8	CSGRW	**581,132.00**	587,924.25	584,719.17	1,569.00	58.34	0.66	0.514
	CSLF	582,654.75	**587,650.00**	**584,485.26**	1,155.05	57.98		
9	CSGRW	604,809.25	**612,667.00**	**609,530.52**	2,146.03	32.66	-0.12	0.902
	CSLF	**602,771.25**	615,903.25	609,619.51	3,317.79	40.79		
10	CSGRW	551,052.50	565,639.25	558,339.95	3,861.85	37.06	1.49	0.143
	CSLF	**546,314.50**	**563,997.25**	**556,667.98**	4,798.57	37.73		
11	CSGRW	**932,438.50**	955,456.00	947,073.63	4,194.05	141.78	1.15	0.256
	CSLF	939,531.25	**955,056.25**	**945,859.18**	4,000.94	145.29		

From Table 8, the CSLF outperformed the CSGRW for finding the best so far solutions (timetables) for most problems because of the lower maximum and average values of total operating costs, whereas the CSGRW outperformed the CSLF for problem number 9. The minimum values of the best so far solutions generated by CSLF were better that the values generated by CSGRW for some problems and vice versa. Moreover, the SD values and the average computational times obtained from both methods were slightly different for all problems.

In term of statistical analysis, the performance difference between the CSLF and the CSGRW was statistically significant with a 95% confidence interval using *t*-test analysis (*P* value ≤ 0.05) for problem

numbers 1, 4 and 7, for small-size and medium-size problems. It can be concluded that the performance of CSGRW was statistically indifference to the CSLF for most problems.

A comparison of convergence speeds between CSLF and CSGRW to find the best so far solution is shown in Figure 16. The 11th problem related with large problem size was selected for this experiment. It can be seen that convergence speed belonging to the CSGRW was slightly better than that belonging to the CSLF in early generations. However, the CSLF was able to find the best so far solutions quicker than the CSGRW from the middle to the last generation.

C. Perfomances of CS's parameter investigations

This experiment was designed to compare the performances of CS via Lévy flights (CSLF) using two different ways to get the appropriate parameter settings: (i) parameter tuning via experimental design and analysis (CSLF+EDA) demonstrated in the first experiment; and (ii) parameter control via self-adaptive parameter setting (CSLF+SPS). The comparisons among the proposed methods for solving eleven problems (detailed in Table 4) were analysed in terms of the minimum, maximum, average, standard deviation (*SD*), and the execution time (*Time*: minute unit) required to find a timetable having the minimum total university operating costs. The *T* value obtained by using the *t*-test method and the *P* value are also shown in Table 9. The amount of search for each method was equally fixed at 24,000 solutions to solve the proposed problems (Table 4), each of which was also repeated thirty times using different random seed numbers.

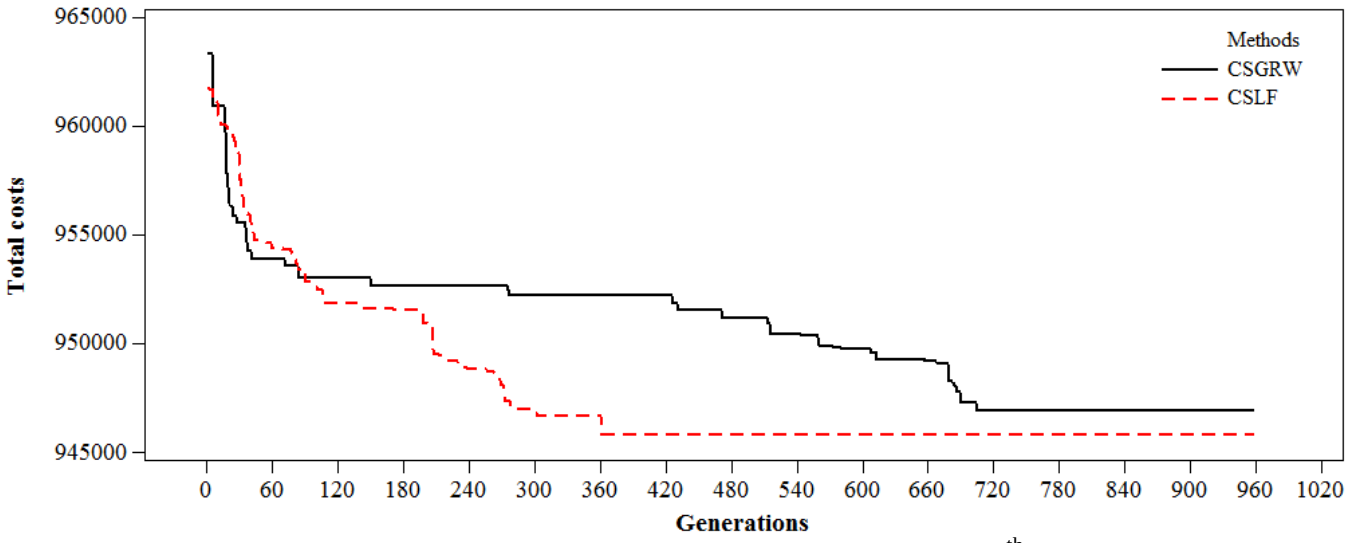


Figure 16 Convergence plots of CSLF and CSGRW for the 11th problem

From Table 9, the CSLF+EDA produced the timetables with the average total operating costs lower than the timetables obtained from the CSLF+SPS for all problems. The minimum and/or maximum values generated from the CSLF+EDA were better than the CSLF+SPS for the first five problems. The *SD* values and the average computational times obtained from both methods were slightly different for all problems, excepted in problem numbers 2 and 4. However, the performance differences achieved by the CSLF+EDA and the CSLF+SPS were not statistically significant with a 95% confidence interval using *t*-test analysis (*P* value $\geq$ 0.05) except for the problem number 5. It means that the performance of the CSLF+SPS was statistically equal to the CSLF+EDA for most problems.

Table 9 Comparative results between the CSLF+EDA and the CSLF+SPS

Prob.	Methods	Best so far solutions					*t*-test	
		Minimum	*Maximum*	*Average*	*SD*	*Time* (*minutes*)	*T* values	*P* values
1	CSLF+SPS	202,016.50	202,229.00	202,105.23	45.34	2.63	0.12	0.903
	CSLF+EDA	**201,998.00**	**202,225.00**	**202,103.62**	56.02	2.53		
2	CSLF+SPS	377,968.75	380,569.00	379,049.68	655.24	12.76	0.94	0.354
	CSLF+EDA	**377,927.25**	**379,754.50**	**378,919.28**	390.96	9.86		
3	CSLF+SPS	301,363.75	306,308.75	303,290.11	983.80	23.99	0.60	0.550
	CSLF+EDA	**301,125.25**	**305,010.75**	**303,146.30**	866.33	24.49		
4	CSLF+SPS	**300,139.25**	307,582.00	303,467.07	2,148.92	19.68	0.91	0.365
	CSLF+EDA	300,210.00	**307,450.25**	**302,937.73**	2,337.29	13.25		
5	CSLF+SPS	488,776.50	491,697.50	489,637.74	599.35	23.91	2.64	**0.011**
	CSLF+EDA	**488,141.50**	**490,141.00**	**489,286.09**	416.09	28.24		
6	CSLF+SPS	**406,088.00**	**409,490.25**	407,992.06	824.44	32.51	0.99	0.326
	CSLF+EDA	406,149.75	409,963.50	**407,758.56**	991.47	35.00		
7	CSLF+SPS	**410,707.75**	419,402.00	415,273.88	2,229.76	20.57	0.85	0.400
	CSLF+EDA	412,089.00	**418,062.50**	**414,820.20**	1,901.27	23.14		
8	CSLF+SPS	**582,162.00**	588,149.50	584,567.33	1,444.74	55.27	0.24	0.809
	CSLF+EDA	582,654.75	**587,650.00**	**584,485.26**	1,155.05	57.98		
9	CSLF+SPS	604,808.25	**615,242.50**	609,723.28	2,902.33	36.02	0.13	0.898
	CSLF+EDA	**602,771.25**	615,903.25	**609,619.51**	3,317.79	40.79		
10	CSLF+SPS	548,167.00	**563,620.50**	557,430.03	4,099.66	39.01	0.66	0.511
	CSLF+EDA	**546,314.50**	563,997.25	**556,667.98**	4,798.57	37.73		
11	CSLF+SPS	**937,061.25**	**954,819.00**	945,879.99	4,234.51	124.22	0.02	0.984
	CSLF+EDA	939,531.25	955,056.25	**945,859.18**	4,000.94	145.29		

A comparison of the convergence speed for the proposed methods to investigate the best so far solution is shown in Figure 17. The problems number 11 that represented a large problem size was selected for example. Although the CSLF+SPS had the performance lower than the CSLF+EDA in middle generations, the averages of best so far solutions obtained from both methods were equal in the last generation. Therefore, it can be concluded that the CSLF's optimal parameter setting investigated by the EDA was able to improve the performance of CSLF better than that investigated by the SPS in terms of solution quality and convergence speed.

Parameter tuning via experimental design and analysis (EDA) is efficient statistical tool to investigate the appropriate parameter setting of the CSLF but it requires a large number of experimental runs and computational resources. For this research, a total run number using experimental design and analysis, full factorial design, obtained from the first experiment is 2,970 (3^2*30*11) runs. Fractional factorial experimental design or reducing number of replications may be suitable if there are enough computational times and resources for all experiments, low number of problems to solve, and no large or very large problems sizes. Otherwise, self-adaptive parameter setting (SPS) is a choice to deal with CSLF's parameter settings so that the amount of computational runs associated with the experimental design for each problem can be dismissed. Moreover, the experimental results indicated that the performances of the CSLF+EDA and the CSLF+SPS were statistically insignificant with a 95% confidence interval.

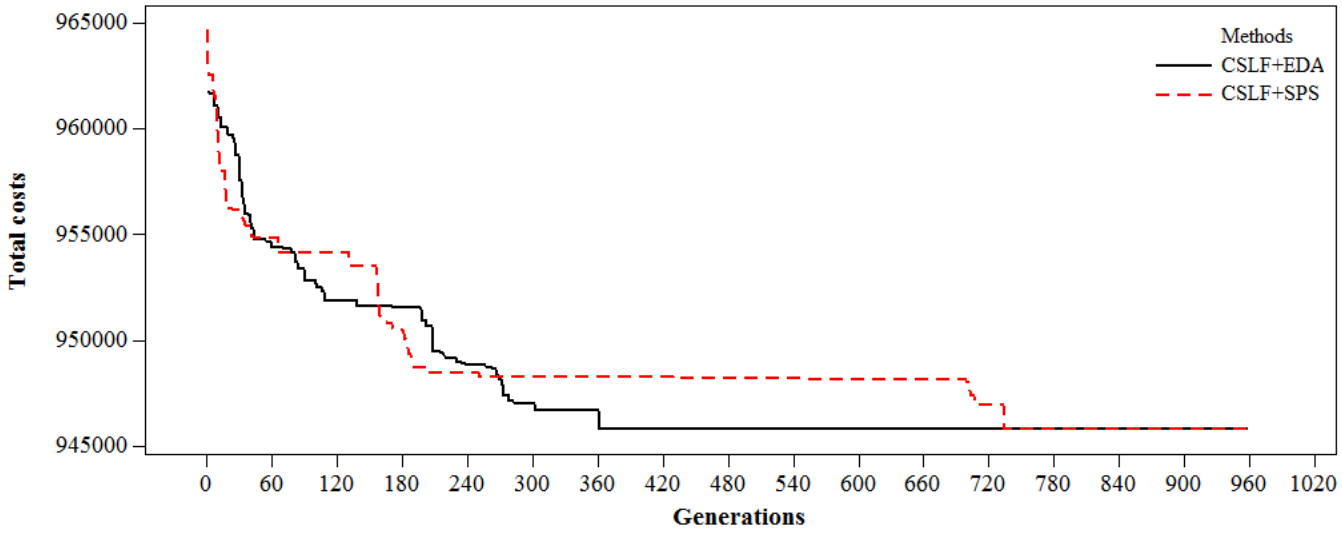


Figure 17 Convergence plots of CSLF+EDA and CSLF+SPS for the 11th problem

D. Perfomances of CSLF with/without LS

This experiment was designed to compare the performance of the CSLF with/without the LS strategies. Two types of the LS including Insertion Operator (IO) and Exchange Operator (EO) were hybridised with the CSLF, called the CSLF+IO and the CSLF+EO, respectively. The performance of the CSLF with/without the LSs was also compared with the well-known metaheuristic method, called Maurice Clerc Particle Swarm Optimisation (MCPSO), which was the best performance of the PSO variants to solve the university course timetabling problems (Thepphakorn and Pongcharoen, 2019). The appropriate parameter setting of the proposed methods was adopted from the first experiment based on the reliable statistical investigations. Because the amount of search for each method was equally fixed at 24,000 solutions, the maximum iteration (*I*) for the CSLF+IO and CSLF+EO were reduced to half compared with the CSLF. Since the total number of improvements within population was double after hybridising with the IO or EO. Solving each dataset shown in Table 4 via CSLF, CSLF+IO, and CSLF+EO was repeated thirty times using different random seed numbers. The comparisons among the proposed hybrid methods for solving eleven datasets were analysed in terms of the minimum, maximum, average, standard deviation (*SD*), and the computational time (*Time*) required to find the best so far timetables. Moreover, the *T* value obtained by using the Tukey's method and the *P* value are also shown in Table 10.

From Table 10, the CSLF with/without the hybrid LSs outperformed the MCPSO in terms of the lower average total operating costs and computational times for all problem datasets. The CSLF hybridised with the LSs can produce the timetables with lower average total university operating costs than the CSLF without the LSs for all problems. The results obtained from CSLF+EO outperformed those obtained from CSLF+IO and CSLF for all datasets. CSLF+IO outperformed CSLF for most datasets, except in problem numbers 1, 2, and 3 related with small problem sizes. Moreover, the best so far solutions constructed by the CSLF+EO were lower than those obtained from other methods in terms of minimum and maximum total operating costs, excepted in problem numbers 2 and 3. The *SD* values obtained from the CSLF hybridised with the LSs were lower than those values obtained from the individual CSLF for many datasets, most of them are related with medium and large problem sizes.

Table 10 Comparative results obtained from the CSLF with/without the LSs and MCPSO

Prob.	**Methods**	**Best so far solutions (total operating costs)**					***Tukey's method***	
		Minimum	*Maximum*	*Average*	*SD*	*Time* (*minutes*)	*T* values	*P* values
1	CSLF	201,998.00	202,225.00	202,103.62	56.02	2.38	-	-
	CSLF+IO	202,032.50	202,202.00	202,111.42	41.21	1.86	-0.520	0.955
	CSLF+EO	**201,878.50**	**201,999.50**	**201,933.75**	**28.73**	1.33	11.280	**0.000**
	MCPSO	202,668.50	203,135.50	203,030.80	134.34	6.44	-43.530	**0.000**
2	CSLF	377,927.25	379,754.50	378,919.28	390.96	13.34	-	-
	CSLF+IO	**374,599.50**	379,625.25	378,993.15	869.67	7.93	-0.528	0.952
	CSLF+EO	378,097.25	**378,759.75**	**378,578.39**	**128.49**	4.70	2.435	0.077
	MCPSO	382,239.00	383,409.50	382,744.20	391.59	22.82	-19.318	**0.000**
3	CSLF	**301,125.25**	305,010.75	303,146.30	866.33	20.62	-	-
	CSLF+IO	301,457.25	305,675.75	303,555.73	875.72	16.05	-2.050	0.177
	CSLF+EO	301,621.50	**304,104.00**	**302,957.78**	**594.69**	9.64	0.944	0.781
	MCPSO	304,349.25	306,025.75	305,400.33	596.58	36.55	-7.979	**0.000**
4	CSLF	300,210.00	307,450.25	302,937.73	2,337.29	14.07	-	-
	CSLF+IO	300,121.50	303,155.00	301,810.26	602.98	8.90	3.239	**0.009**
	CSLF+EO	**299,378.25**	**300,493.00**	**299,818.81**	**283.40**	5.71	8.959	**0.000**
	MCPSO	307,855.75	309,819.75	308,821.35	597.81	26.75	-11.951	**0.000**
5	CSLF	488,141.50	490,141.00	489,286.09	416.09	19.76	-	-
	CSLF+IO	488,741.00	490,562.50	489,143.93	**402.93**	16.28	1.277	0.580
	CSLF+EO	**487,315.00**	**489,495.25**	**488,342.48**	450.17	9.23	8.473	**0.000**
	MCPSO	492,248.50	493,737.00	493,002.15	500.36	47.81	-23.595	**0.000**
6	CSLF	406,149.75	409,963.50	407,758.56	991.47	35.00	-	-
	CSLF+IO	406,073.00	409,059.75	407,526.06	692.80	25.65	1.174	0.645
	CSLF+EO	**405,637.75**	**408,083.75**	**407,048.07**	657.90	15.39	3.587	**0.003**
	MCPSO	409,361.50	410,730.00	410,328.68	**409.93**	59.83	-9.175	**0.000**
7	CSLF	412,089.00	418,062.50	414,820.20	1,901.27	23.14	-	-
	CSLF+IO	410,564.75	413,794.75	412,277.72	799.64	15.75	8.380	**0.000**
	CSLF+EO	**408,279.50**	**410,263.25**	**409,320.88**	**409.69**	9.07	18.130	**0.000**
	MCPSO	418,731.50	420,752.00	419,722.18	688.56	31.97	-11.430	**0.000**
8	CSLF	582,654.75	587,650.00	584,485.26	1,155.05	57.98	-	-
	CSLF+IO	581,817.00	585,534.50	583,827.22	765.29	31.10	2.994	**0.018**
	CSLF+EO	**581,335.25**	**583,897.00**	**582,670.56**	**597.82**	18.75	8.258	**0.000**
	MCPSO	587,955.00	589,903.75	589,177.85	623.70	92.22	-15.099	**0.000**
9	CSLF	602,771.25	615,903.25	609,619.51	3,317.79	40.79	-	-
	CSLF+IO	602,841.25	607,679.50	605,291.43	1,161.05	24.98	8.320	**0.000**
	CSLF+EO	**598,125.75**	**602,827.50**	**600,343.48**	988.88	16.07	17.820	**0.000**
	MCPSO	616,349.00	618,259.00	617,236.60	**612.54**	54.29	-10.350	**0.000**
10	CSLF	546,314.50	563,997.25	556,667.98	4,798.57	37.73	-	-
	CSLF+IO	547,759.50	553,854.50	550,632.50	1,495.93	33.62	8.270	**0.000**
	CSLF+EO	**540,221.00**	**543,150.50**	**541,621.70**	**913.53**	24.64	20.620	**0.000**
	MCPSO	566,033.25	568,658.00	567,615.93	1,051.13	105.42	-10.610	**0.000**
11	CSLF	939,531.25	955,056.25	945,859.18	4,000.94	145.29	-	-
	CSLF+IO	926,254.75	933,329.00	930,400.41	1,874.13	61.96	22.110	**0.000**
	CSLF+EO	**909,172.75**	**917,453.00**	**914,026.72**	1,954.70	36.86	45.530	**0.000**
	MCPSO	956,528.50	961,157.00	958,843.10	**1,734.03**	212.91	-13.130	**0.000**

In term of statistical analysis, the performance differences achieved by the CSLF with/without the LSs and MCPSO were statistically significant with a 95% confidence interval using Tukey's method (*P* value $\leq$ 0.05) for all problems. The results obtained from CSLF were significantly better than those obtained from MCPSO for all problems. The CSLF+EO method was the best configuration of hybrid LSs for most problems because of the highest *T* value. Therefore, it can be concluded that the performances of the CSLF based upon the concept of individual diversification can dramatically be improved by hybridising with the LSs related with intensification concept.

The average execution times required for the CSLF+IO and CSLF+EO were obviously lower than the CSLF whereas the MCPSO required the longest execution times to solve the problems. For example, the 11th problem related with large problem sizes, CSLF+EO and CSLF+IO were able to reduce the computational times (comparing with the CSLF) up to 75% and 57%, respectively. Since the number of improved solutions within population were double after producing the LS improvements (both IO and EO). Therefore, the number of iterations (*I*) for the proposed hybrid methods was reduced to half in order to control the amount of search at the 24,000 solutions. Moreover, the IO and EO are also simpler compared with the other improvement processes.

The 11th problem related with large problem sizes was selected as an example to: (i) demonstrate a comparison of the convergence speeds for the MCPSO and CSLF with/without the LSs; and (ii) investigate the best so far solution shown in Figure 18. It can be seen that the convergence line belonging to the CSLF+EO can converge to the minimum total operating costs quicker than that lines generated from MCPSO, CSLF, and CSLF+IO methods. The CSLF+IO was the second ranking to find the average of best so far solution. Therefore, the LS strategies were able to improve the performances of the CSLF both in terms of the solution quality and its convergence speed.

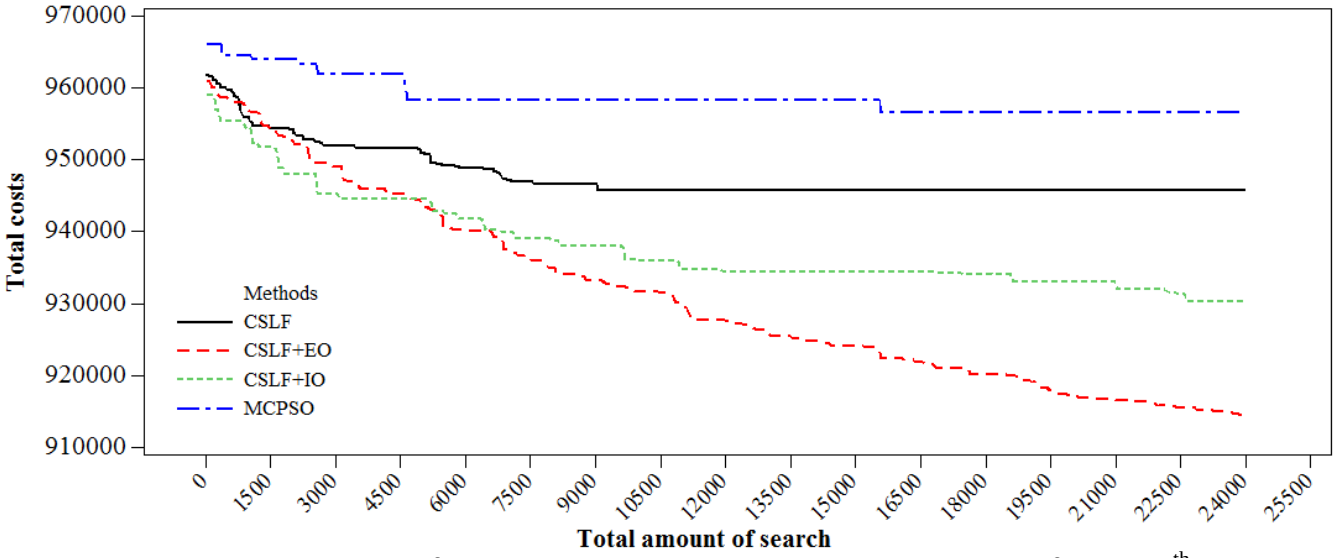


Figure 18 Convergence plots of CSLF with/without LS strategies and MCPSO for the 11th problem

6. Conclusions

A novel Hybrid Self-adaptive Cuckoo Search based Timetabling (HSCST) tool has been developed in order to solve the real world university course timetabling problems (UCTP). The solutions or timetables generated by the HSCST program were measured by minimising the total university operation costs: (i) lecturing cost; (ii) classroom operating cost; and (iii) classroom setup/cleaning cost. Two random walks strategies for the Cuckoo Search (CS) algorithm, called CS via Lévy flights (CSLF) and CS via Gaussian random walks (CSGRW), were embedded in the HSCST program. Two local search (LS) heuristics including Insertion Operator (IO) and Exchange Operator (EO) were hybridised with the

proposed algorithm (CSLF+IO and CSLF+EO). Moreover, the CSLF integrated with self-adaptive parameter setting (CSLF+SPS) technique was developed and also embedded in the HSCST program before comparing its performance with the CSLF using optimal parameters obtained from the experimental design and analysis (CSLF+EDA). Eleven datasets obtained from Naresuan University in Thailand were computationally conducted using the proposed program. This paper also demonstrated the use of the experimental design and analysis for investigating the appropriate parameters setting of the proposed method before sequentially conducting a comparative study on its performance.

The experimental results indicated that the optimised parameter settings of the CS algorithm investigated via experimental design and analysis were different in various problem sizes. In addition, the P_a factor was a robust parameter with no statistical significance with 95% confidence interval for all problems. The performance of CSGRW was statistically equal in the performances of CSLF for most problems, whereas the average computational times required by both algorithms were slightly different. Finding the timetable with the lowest total operating costs using the CSLF+SPS was statistically equal to that using the CSLF+EDA for most problems. Moreover, the CSLF+SPS will be suitable for the cases of limited computational resources and times to solve a lot of UCTPs, each of which has very large problem sizes. The performances of CSLF can be dramatically improved by using the LS strategies in terms of solution quality, convergence speeds, and execution times for all problems. The CSLF+EO was the best performances followed by the CSLF+IO. The average execution times required for both hybridised algorithms were obviously lower than the CSLF because of controlled amount of search and the simple processes of the LSs. For example, the 11^{th} dataset related with large problem sizes, CSLF+EO and CSLF+IO were able to reduce the computational times up to 75% and 57%, respectively. Moreover, the CSLF with/without LS hybridisations outperformed the MCPSO in terms of the lower average total operating costs, convergence speeds, and computational times.

Further work could be focused on the dissemination of these improvement strategies on other optimisation methods or the applications of these strategies on the different problem domains.

Acknowledgements

This work was part of research project co-funded between the Thailand Science Research and Innovation (TSRI) and the Office of the Higher Education Commission (OHEC) of Thailand under grant number MRG6080066.

References

Abdel-Basset, M., Hessin, A.N., Abdel-Fatah, L., 2018. A comprehensive study of cuckoo-inspired algorithms. Neural Computing & Applications 29, 345-361.

Abdullah, S., Burke, E.K., McCollum, B., 2005. An Investigation of Variable Neighbourhood Search for University Course Timetabling, in: Kendall, G., Lei, L., Pinedo, M. (Eds.), The 2nd Multidisciplinary Conference on Scheduling: Theory and Applications (MISTA 2005), New York, USA, pp. 413-427.

Abdullah, S., Turabieh, H., McCollum, B., McMullan, P., 2012. A hybrid metaheuristic approach to the university course timetabling problem. Journal of Heuristics 18, 1-23.

Abuhamdah, A., Ayob, M., Kendall, G., Sabar, N.R., 2014. Population based Local Search for university course timetabling problems. Applied Intelligence 40, 44-53.

Al-Betar, M.A., Khader, A.T., 2012. A harmony search algorithm for university course timetabling. Annals of Operations Research 194, 3-31.

Al-Betar, M.A., Khader, A.T., Gani, T.A., 2008. A harmony search algorithm for university course timetabling, 7th International Conference on the Practice and Theory of Automated Timetabling, PATAT 2008.

Al-Betar, M.A., Khader, A.T., Zaman, M., 2012. University Course Timetabling Using a Hybrid Harmony Search Metaheuristic Algorithm. Ieee Transactions on Systems Man and Cybernetics Part C-Applications and Reviews 42, 664-681.

Aladag, C.H., Hocaoglu, G., 2007. A TABU SEARCH ALGORITHM TO SOLVE A COURSE TIMETABLING PROBLEM. Hacettepe Journal of Mathematics and Statistics 36, 53-64.

Alirezaei, E., Vahedi, Z., Ghaznavi-Ghoushchi, M., 2012. Parallel hybrid meta heuristic algorithm for university course timetabling problem (PHACT), pp. 673-678.

Badoni, R.P., Gupta, D.K., 2016. A new algorithm based on students groupings for university course timetabling problem.

Bellio, R., Ceschia, S., Di Gaspero, L., Schaerf, A., Urli, T., 2016. Feature-based tuning of simulated annealing applied to the curriculum-based course timetabling problem. Computers & Operations Research 65, 83-92.

Bolaji, A.L., Khader, A.T., Al-Betar, M.A., Awadallah, M.A., 2013. A modified artificial bee colony algorithm for post-enrolment course timetabling, pp. 377-386.

Bolaji, A.L., Khader, A.T., Al-Betar, M.A., Awadallah, M.A., 2014. University course timetabling using hybridized artificial bee colony with hill climbing optimizer. Journal of Computational Science 5, 809-818.

Budiono, T.A., Wong, K.W., 2011. Memetic algorithm behavior on timetabling infeasibility, pp. 93-97.

Burke, E., Bykov, Y., Newall, J., Petrovic, S., 2003. A time-predefined approach to course timetabling. Yugoslav Journal of Operations Research 13, 139-151.

Burke, E.K., Marecek, J., Parkes, A.J., Rudova, H., 2010. Decomposition, reformulation, and diving in university course timetabling. Computers & Operations Research 37, 582-597.

Burke, E.K., McCollum, B., Meisels, A., Petrovic, S., Qu, R., 2007. A graph-based hyper-heuristic for educational timetabling problems. European Journal of Operational Research 176, 177-192.

Burke, E.K., Newall, J.P., 2004. Solving examination timetabling problems through adaption of heuristic orderings. Annals of Operations Research 129, 107-134.

Ceschia, S., Di Gaspero, L., Schaerf, A., 2012. Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem. Computers & Operations Research 39, 1615-1624.

Chen, R.-M., Shih, H.-F., 2013. Solving University Course Timetabling Problems Using Constriction Particle Swarm Optimization with Local Search. Algorithms 6, 227-244.

Chiarandini, M., Birattari, M., Socha, K., Rossi-Doria, O., 2006. An effective hybrid algorithm for university course timetabling. Journal of Scheduling 9, 403-432.

Chiroma, H., Herawan, T., Fister, I., Fister, I., Abdulkareem, S., Shuib, L., Hamza, M.F., Saadi, Y., Abubakar, A., 2017. Bio-inspired computation: Recent development on the modifications of the cuckoo search algorithm. Applied Soft Computing 61, 149-173.

Crawford, B., Soto, R., Johnson, F., Paredes, F., 2015. A timetabling applied case solved with ant colony optimization, Advances in Intelligent Systems and Computing, pp. 267-276.

De Causmaecker, P., Demeester, P., Vanden Berghe, G., 2009. A decomposed metaheuristic approach for a real-world university timetabling problem. European Journal of Operational Research 195, 307-318.

Dino Matijaš, V., Molnar, G., Čupić, M., Jakobović, D., Dalbelo Bašić, B., 2010. University course timetabling using ACO: A case study on laboratory exercises, pp. 100-110.

Dun, Y.J., Wang, Q., Shao, Y.B., 2014. A simulated annealing genetic algorithm for solving timetable problems, Advances in Intelligent Systems and Computing, pp. 365-374.

Eiben, A.E., Michalewicz, Z., Schoenauer, M., Smith, J.E., 2007. Parameter Control in Evolutionary Algorithms. Studies in Computational Intelligence 54, 19-46.

Eiben, A.E., Smit, S.K., 2011. Parameter tuning for configuring and analyzing evolutionary algorithms. Swarm and Evolutionary Computation 1, 19-31.

Figlali, N., Ozkale, C., Engin, O., Figlali, A., 2009. Investigation of Ant System parameter interactions by using design of experiments for job-shop scheduling problems. Computers & Industrial Engineering 56, 538-559.

Fister Jr, I., Fister, D., Fistar, I., 2013. A comprehensive review of Cuckoo search: Variants and hybrids. International Journal of Mathematical Modelling and Numerical Optimisation 4, 387-409.

Fong, C.W., Asmuni, H., McCollum, B., 2015. A hybrid swarm-based approach to university timetabling. IEEE Transactions on Evolutionary Computation 19, 870-884.

Geiger, M.J., 2008. An application of the threshold accepting metaheuristic for curriculum based course timetabling, 7th International Conference on the Practice and Theory of Automated Timetabling, PATAT 2008.

Geiger, M.J., 2012. Applying the threshold accepting metaheuristic to curriculum based course timetabling A contribution to the second international timetabling competition ITC 2007. Annals of Operations Research 194, 189-202.

Gunawan, A., Ming, N.K., Leng, P.K., 2008. A hybrid algorithm for the university course timetabling problem, 7th International Conference on the Practice and Theory of Automated Timetabling, PATAT 2008.

He, Y., Hui, S., Lai, E.-K., 2005. Automatic Timetabling Using Artificial Immune System. Lecture Notes in Computer Science 3521, 55-65.

Jafarinejad, T., Erfani, A., Fathi, A., Shafii, M.B., 2019. Bi-level energy-efficient occupancy profile optimization integrated with demand-driven control strategy: University building energy saving. Sustainable Cities and Society 48.

Jaradat, G., Ayob, M., Ahmad, Z., 2014. On the performance of Scatter Search for post-enrolment course timetabling problems. Journal of Combinatorial Optimization 27, 417-439.

Jaradat, G.M., Ayob, M., 2010. An elitist-ant system for solving the post-enrolment course timetabling problem, pp. 167-176.

Jaradat, G.M., Ayob, M., 2011. Scatter search for solving the course timetabling problem, pp. 213-218.

Jaradat, G.M., Ayob, M., 2013. Effect of Elite Pool and Euclidean Distance in Big Bang-Big Crunch Metaheuristic for Post-Enrolment Course TimetablingProblems. International Journal of Soft Computing 8, 96-107.

Jat, S.N., Yang, S., 2011a. A guided search non-dominated sorting genetic algorithm for the multi-objective university course timetabling problem. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 6622 LNCS, 1-13.

Jat, S.N., Yang, S.X., 2011b. A hybrid genetic algorithm and tabu search approach for post enrolment course timetabling. Journal of Scheduling 14, 617-637.

Joudaki, M., Imani, M., Mazhari, N., 2011. Using improved memetic algorithm and local search to solve university Course Timetabling problem (UCTP), pp. 501-506.

Jula, A., Naseri, N.K., 2011. Using CMAC to obtain dynamic mutation rate in a metaheuristic memetic algorithm to solve university timetabling problem. European Journal of Scientific Research 63, 172-181.

Junaedi, D., Maulidevi, N.U., 2011. Solving Curriculum-Based Course Timetabling Problem with Artificial Bee Colony Algorithm, The 1st International Conference on Informatics and Computational Intelligence (ICI 2011), Bandung, Indonesia, pp. 112-117.

Junn, K.Y., Obit, J.H., Alfred, R., 2017. Comparison of simulated annealing and great deluge algorithms for university course timetabling problems (UCTP). Advanced Science Letters 23, 11413-11417.

Junn, K.Y., Obit, J.H., Alfred, R., 2018. The Study of Genetic Algorithm Approach to Solving University Course Timetabling Problem, Lecture Notes in Electrical Engineering, pp. 454-463.

Karakatič, S., Podgorelec, V., 2015. A survey of genetic algorithms for solving multi depot vehicle routing problem. Applied Soft Computing 27, 519-532.

Karami, A.H., Hasanzadeh, M., 2012. University course timetabling using a new hybrid genetic algorithm, pp. 144-149.

Khadwilard, A., Chansombat, S., Thepphakorn, T., Thapatsuwan, P., Chainate, W., Pongcharoen, P., 2012. Application of Firefly Algorithm and Its Parameter Setting for Job Shop Scheduling. The Journal of Industrial Technology 8, 49-58.

Khang, N.T.T.M., Phuc, N.B., Nuong, T.T.H., 2011. The bees algorithm for a practical university timetabling problem in Vietnam, pp. 42-47.

Khoja, I., Ladhari, T., M'Sahli, F., Sakly, A., 2018. Cuckoo Search Approach for Parameter Identification of an Activated Sludge Process. Computational Intelligence and Neuroscience.

Kiefer, A., Hartl, R.F., Schnell, A., 2017. Adaptive large neighborhood search for the curriculum-based course timetabling problem. Annals of Operations Research 252, 255-282.

Kostuch, P., 2005. The University Course Timetabling Problem with a Three-Phase Approach. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 3616 LNCS, 109-125.

La'aro Bolaji, A., Tajudin Khader, A., Azmi Al-Betar, M., Awadallah, M.A., 2011. An improved artificial bee colony for Course Timetabling, pp. 9-14.

Lewis, R., 2008. A survey of metaheuristic-based techniques for University Timetabling problems. OR Spectrum 30, 167-190.

Lewis, R., 2012. A time-dependent metaheuristic algorithm for post enrolment-based course timetabling. Annals of Operations Research 194, 273-289.

Lewis, R., Paechter, B., 2007. Finding feasible timetables using group-based operators. Ieee Transactions on Evolutionary Computation 11, 397-413.

Lewis, R., Paechter, B., Rossi-Doria, O., 2007. Metaheuristics for university course timetabling, Studies in Computational Intelligence, pp. 237-272.

Lewis, R., Thompson, J., 2015. Analysing the effects of solution space connectivity with an effective metaheuristic for the course timetabling problem. European Journal of Operational Research 240, 637-648.

Li, X., Yin, M., 2015. Modified cuckoo search algorithm with self adaptive parameter method. Information Sciences 298, 80-97.

Lindahl, M., Sorensen, M., Stidsen, T.R., 2018. A fix-and-optimize matheuristic for university timetabling. Journal of Heuristics 24, 645-665.

Lü, Z., Hao, J.-K., 2010. Adaptive Tabu Search for course timetabling. European Journal of Operational Research 200, 235-244.

Lu, Z.P., Hao, J.K., Glover, F., 2011. Neighborhood analysis: a case study on curriculum-based course timetabling. Journal of Heuristics 17, 97-118.

Mansour, N., El-Jazzar, H., 2013. Curriculum based course timetabling, pp. 787-792.
Matias, J.B., Fajardo, A.C., Medina, R.M., 2018. A fair course timetabling using genetic algorithm with guided search technique, pp. 77-82.
Mauritsius, T., Fajar, A.N., Harisno, John, P., 2018. Novel Local Searches for Finding Feasible Solutions in Educational Timetabling Problem, pp. 270-275.
Mayer, A., Nothegger, C., Chwatal, A., Raidl, G.R., 2008. Solving the post enrolment course timetabling problem by ant colony optimization, 7th International Conference on the Practice and Theory of Automated Timetabling, PATAT 2008.
Mazlan, M., Makhtar, M., Ahmad Khairi, A.F.K., Mohamed, M.A., 2019. University course timetabling model using ant colony optimization algorithm approach. Indonesian Journal of Electrical Engineering and Computer Science 13, 72-76.
Mlakar, U., Fister Jr., I., Fister, I., 2016. Hybrid self-adaptive cuckoo search for global optimization. Swarm and Evolutionary Computation 29, 47-72.
Mohamad, A.B., Zain, A.M., Bazin, N.E.N., 2014. Cuckoo Search Algorithm for Optimization Problems - A Literature Review and its Applications. Applied Artificial Intelligence 28, 419-448.
Montgomery, D.C., 2012. Design and Analysis of Experiments, 8 ed. John Wiley & Sons, Incorporated.
Murray, K., Müller, T., Rudová, H., 2006. Modeling and Solution of a Complex University Course Timetabling Problem. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 3867 LNCS, 189-209.
Nguyen, K., Dang, N., Trieu, K., Tran, N., 2010. Automating a real-world university timetabling problem with Tabu search algorithm, 2010 IEEE-RIVF International Conference on Computing and Communication Technologies: Research, Innovation and Vision for the Future, RIVF 2010.
Nguyen, K., Nguyen, Q., Tran, H., Nguyen, P., Tran, N., 2011. Variable neighborhood search for a real-world curriculum-based university timetabling problem, pp. 157-162.
Nothegger, C., Mayer, A., Chwatal, A., Raidl, G.R., 2012. Solving the post enrolment course timetabling problem by ant colony optimization. Annals of Operations Research 194, 325-339.
Obit, J.H., Alfred, R., Abdalla, M.H., 2017a. A PSO inspired asynchronous cooperative distributed hyper-heuristic for course timetabling problems. Advanced Science Letters 23, 11016-11022.
Obit, J.H., Junn, K.Y., Alfred, R., 2017b. A performance comparison of metaheuristics search for university course timetabling problems. Advanced Science Letters 23, 11012-11015.
Ortiz-Aguilar, L.M., Carpio, M., Puga, H., Soria-Alcaraz, J.A., Ornelas-Rodríguez, M., Lino, C., 2017. Increase methodology of design of course timetabling problem for students, classrooms, and teachers, Studies in Computational Intelligence, pp. 713-728.
Ozcan, E., Parkes, A.J., Alkan, A., 2012. The Interleaved Constructive Memetic Algorithm and its application to timetabling. Computers & Operations Research 39, 2310-2322.
Pandey, H.M., Chaudhary, A., Mehrotra, D., 2014. A comparative review of approaches to prevent premature convergence in GA. Applied Soft Computing 24, 1047-1077.
Patel, J., Savsani, V., Patel, V., Patel, R., 2017. Layout optimization of a wind farm to maximize the power output using enhanced teaching learning based optimization technique. Journal of Cleaner Production 158, 81-94.
Phuc, N.B., Khang, N.T.T.M., Nuong, T.T.H., 2011. A new hybrid GA-Bees algorithm for a real-world university timetabling problem, pp. 321-326.
Pillay, N., Özcan, E., 2017. Automated generation of constructive ordering heuristics for educational timetabling. Annals of Operations Research, 1-28.
Pillay, N., Özcan, E., 2019. Automated generation of constructive ordering heuristics for educational timetabling. Annals of Operations Research 275, 181-208.
Pongcharoen, P., Promtet, W., Yenradee, P., Hicks, C., 2008. Stochastic Optimisation Timetabling Tool for university course scheduling. International Journal of Production Economics 112, 903-918.
Qarouni-Fard, D., Najafi-Ardabifi, A., Moeinzadeh, M.H., Sharifian-R, S., Asgarian, E., Mohammadzadeh, J., 2008. Finding feasible timetables with particle swarm optimization, pp. 387-391.
Rakhshani, H., Dehghanian, E., Rahati, A., 2016. Hierarchy cuckoo search algorithm for parameter estimation in biological systems. Chemometrics and Intelligent Laboratory Systems 159, 97-107.
Rakhshani, H., Rahati, A., 2017. Snap-drift cuckoo search: A novel cuckoo search optimization algorithm. Applied Soft Computing 52, 771-794.
Rossi-Doria, O., Sampels, M., Birattari, M., Chiarandini, M., Dorigo, M., Gambardella, L.M., Knowles, J., Manfrin, M., Mastrolilli, M., Paechter, B., Paquete, L., Stutzle, T., 2003. A comparison of the performance of different

metaheuristics on the timetabling problem, in: Burke, E., DeCausmaecker, P. (Eds.), Practice and Theory of Automated Timetabling Iv, pp. 329-351.
Rudova, H., Muller, T., Murray, K., 2011. Complex university course timetabling. Journal of Scheduling 14, 187-207.
Salgotra, R., Singh, U., Saha, S., 2018. New cuckoo search algorithms with enhanced exploration and exploitation properties. Expert Systems with Applications 95, 384-420.
Shehab, M., Khader, A.T., Al-Betar, M.A., 2017. A survey on applications and variants of the cuckoo search algorithm. Applied Soft Computing 61, 1041-1059.
Socha, K., Sampels, M., Manfrin, M., 2003. Ant algorithms for the university course timetabling problem with regard to the state-of-the-art, in: Raidl, G., Cagnoni, S., Cardalda, J.J.R., Corne, D.W., Gottlieb, J., Guillot, A., Hart, E., Johnson, C.G., Marchiori, E., Meyer, J.A., Middendorf, M. (Eds.), Applications of Evolutionary Computing, pp. 334-345.
Soria-Alcaraz, J.A., Martin, C., Héctor, P., Hugo, T.M., Laura, C.R., Sotelo-Figueroa, M.A., 2013. Methodology of design: A novel generic approach applied to the course timetabling problem, Studies in Fuzziness and Soft Computing, pp. 287-319.
Soria-Alcaraz, J.A., Ochoa, G., Sotelo-Figeroa, M.A., Burke, E.K., 2017a. A methodology for determining an effective subset of heuristics in selection hyper-heuristics. European Journal of Operational Research 260, 972-983.
Soria-Alcaraz, J.A., Ochoa, G., Sotelo-Figueroa, M.A., Carpio, M., Puga, H., 2017b. Iterated VND versus hyper-heuristics: Effective and general approaches to course timetabling, Studies in Computational Intelligence, pp. 687-700.
Soria-Alcaraz, J.A., Ochoa, G., Swan, J., Carpio, M., Puga, H., Burke, E.K., 2014. Effective learning hyper-heuristics for the course timetabling problem. European Journal of Operational Research 238, 77-86.
Soria-Alcaraz Jorge, A., Martín, C., Héctor, P., Sotelo-Figueroa, M.A., 2013. Comparison of metaheuristic algorithms with a methodology of design for the evaluation of hard constraints over the course timetabling problem, Studies in Computational Intelligence, pp. 289-302.
Talbi, E.-G., 2009. Metaheuristics: From Design to Implementation. John Wiley & Sons.
Tarawneh, H.Y., Ayob, M., 2013. Adaptive neighbourhoods structure selection mechanism in simulated annealing for solving university course timetabling problems. Journal of Applied Sciences 13, 1087-1093.
Teoh, C.K., Wibowo, A., Ngadiman, M.S., 2014. An adapted cuckoo optimization algorithm and genetic algorithm approach to the university course timetabling problem. International Journal of Computational Intelligence and Applications 13.
Thepphakorn, T., Pongcharoen, P., 2013. Heuristic ordering for ant colony based timetabling tool. Journal of Applied Operational Research 5, 113-123.
Thepphakorn, T., Pongcharoen, P., 2019. Variants and Parameters Investigations of Particle Swarm Optimisation for Solving Course Timetabling Problems. Lecture Notes in Computer Science 11655, 177-187.
Thepphakorn, T., Pongcharoen, P., Hicks, C., 2014. An Ant Colony Based Timetabling Tool. International Journal of Production Economics 149, 131-144.
Thepphakorn, T., Pongcharoen, P., Hicks, C., 2015. Modifying Regeneration Mutation and Hybridising Clonal Selection for Evolutionary Algorithms Based Timetabling Tool. Mathematical Problems in Engineering 2015, Article Number 841748, 16.
Thepphakorn, T., Pongcharoen, P., Vitayasak, S., 2016. A new multiple objective cuckoo search for university course timetabling problem. Lecture Notes in Computer Science 10053 LNAI, 196-207.
Turabieh, H., El-Daoud, E., 2012. University course timetabling problem at Zarqa University.
Valian, E., Tavakoli, S., Mohanna, S., Haghi, A., 2013. Improved cuckoo search for reliability optimization problems. Computers & Industrial Engineering 64, 459-468.
Vitayasak, S., Pongcharoen, P., 2018. Performance improvement of Teaching-Learning-Based Optimisation for robust machine layout design. Expert Systems with Applications 98, 129-152.
Vitayasak, S., Pongcharoen, P., Hicks, C., 2017. A tool for solving stochastic dynamic facility layout problems with stochastic demand using either a Genetic Algorithm or modified Backtracking Search Algorithm. International Journal of Production Economics 190, 146-157.
Wahid, J., Abdul-Rahman, S., Mohamed Din, A., Mohd-Hussin, N., 2019. Constructing population of initial university timetable: Design and analysis. Indonesian Journal of Electrical Engineering and Computer Science 15, 1109-1118.
Wahid, J., Hussin, N.M., 2016a. Combination of graph heuristics in producing initial solution of curriculum based course timetabling problem.
Wahid, J., Hussin, N.M., 2016b. Construction of initial solution population for curriculum-based course timetabling using combination of graph heuristics. Journal of Telecommunication, Electronic and Computer Engineering 8, 91-95.
Yang, X.-S., 2010. Nature-Inspired Metaheuristic Algorithms, 2 ed. Luniver Press, University of Cambrige, United Kingdom.
Yang, X.-S., 2014. Nature-Inspired optimization algorithms. Elsevier.
Yang, X.-S., Chien, S.F., Ting, T.O., 2014. Computational Intelligence and Metaheuristic Algorithms with Applications. The Scientific World Journal 2014, 4.

Yang, X.-S., Deb, S., 2010. Engineering Optimisation by Cuckoo Search. International Journal of Mathematical Modelling and Numerical Optimisation 1, 330-343.

Yang, X.-S., Deb, S., 2013. Multiobjective cuckoo search for design optimization. Computers & Operations Research 40, 1616-1624.

Yang, X.S., Deb, S., 2014. Cuckoo search: recent advances and applications. Neural Computing & Applications 24, 169-174.

Yassin, R.M., Nazri, M.Z.A., Abdullah, S., 2013. Hybrid approach: Tabu-based non-linear great deluge for the course timetabling problem. Research Journal of Applied Sciences 8, 131-138.

Zhang, M.X., Zhang, B., Qian, N., 2017. University course timetabling using a new ecogeography-based optimization algorithm. Natural Computing 16, 61-74.

Zhao, W.B., Niu, D.X., 2017. Prediction of CO2 Emission in China's Power Generation Industry with Gauss Optimized Cuckoo Search Algorithm and Wavelet Neural Network Based on STIRPAT model with Ridge Regression. Sustainability 9.

Zheng, H., Zhou, Y., 2012. A novel cuckoo search optimization algorithm based on Gauss distribution. Journal of Computational Information Systems 8, 4193–4200.

Zhu, X.H., Wang, N., 2017. Cuckoo search algorithm with membrane communication mechanism for modeling overhead crane systems using RBF neural networks. Applied Soft Computing 56, 458-471.

Highlights

- ✓ First report on modified and hybrid Cuckoo Search for real-world course timetabling
- ✓ Comprehensive review on metaheuristics applied to solve course timetabling problem
- ✓ Describe the Hybrid Self-adaptive Cuckoo Search based Timetabling (HSCST) tool
- ✓ Proposed three improvement strategies: parameter setting, movement and hybridisation
- ✓ The proposed algorithms outperformed other conventional algorithms by 81.8%

***Conflict of Interest**

Declaration of interests

☒ The authors declare that they have no conflict of interests on the work reported in this paper.

☐ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Variants and Parameters Investigations of Particle Swarm Optimisation for Solving Course Timetabling Problems

Thatchai Thepphakorn[1] and Pupong Pongcharoen[2(✉)]

[1] Faculty of Industrial Technology, Pibulsongkram Rajabhat University, Phitsanulok 65000, Thailand
[2] Centre of Operations Research and Industrial Applications (CORIA), Department of Industrial Engineering, Faculty of Engineering, Naresuan University, Phitsanulok 65000, Thailand
pupongp@nu.ac.th

Abstract. University course timetabling problem (UCTP) is well known to be Non-deterministic Polynomial (NP)-hard problem, in which the amount of computational time required to find the optimal solutions increases exponentially with problem size. Solving the UCTP manually with/without course timetabling tool is extremely difficult and time consuming. A particle swarm optimisation based timetabling (PSOT) tool has been developed in order to solve the real-world datasets of the UCTP. The conventional particle swarm optimisation (PSO), the standard particle swarm optimisation (SPSO), and the Maurice Clerc particle swarm optimisation (MCPSO) were embedded in the PSOT program for optimising the desirable objective function. The analysis of variance on the computational results indicated that both main effect and interactions were statistically significant with a 95% confidence interval. The MCPSO outperformed the other variants of PSO for most datasets whilst the computational times required by all variants were moderately difference.

Keywords: Course timetabling · Particle swarm · Metaheuristic · Parameter setting

1 Introduction

University course timetabling problem (UCTP) is one of the most challenging scheduling problems and also classified into combinatorial optimisation problems due to its complexity and constraints [1]. This problem arises every semester and is solved either manually by academic staff or using automatic course timetabling tool [2, 3]. Solving large course timetabling problems without efficient timetabling program is extremely difficult and may require a group of experts to work for several days [4].

Swarm intelligence (SI) has received great attention in the communities of optimisation, computer science, computational intelligence, bio-inspired algorithms, and SI-based algorithms [5]. SI-based algorithms such as ant colony optimisation (ACO), artificial bee colony (ABC) algorithm, firefly algorithm (FA), cuckoo search (CS), and particle swarm optimisation (PSO) have become very popular to solve large-scale

Y. Tan et al. (Eds.): ICSI 2019, LNCS 11655, pp. 177–187, 2019.
https://doi.org/10.1007/978-3-030-26369-0_17

combinatorial optimisation problems [5]. These algorithms have been widely adopted to solve NP hard problems within acceptable computational time, but they do not guarantee optimum solutions [6]. Among the intelligent algorithms, PSO has been successfully applied to solve problems in several domains such as clustering problem, image processing, function optimisation, etc. This is because of PSO algorithm has a few parameters to adjust and requires little memory for computation, easy to understand and implement [7].

We have conducted a comprehensive literature survey on articles indexed in Scopus databases covering the period from the past to February 2019 using "course timetabl*" and "particle swarm*" as keywords, several variants of PSO were found to be applied to solve the UCTP. For examples, the conventional PSO (called PSO) has been applied to generate the optimal course timetables for 16 lecturers, 10 classrooms, and 10 classes [8]. The standard PSO using inertia weight factor (called SPSO) has been also developed to solve the UCTP both real world datasets [9] and benchmarking datasets [10]. Another variant of PSO introduced by Maurice Clerc (called MCPSO) has been wildly applied to deal with the UCTP [11–14]. However, there is no report related with the performance comparison among three variants of PSO to solve the UCTP. Moreover, parameter values of PSOs found on all articles have been set by using ad hoc fashion approach [10–14] or one factor at a time experimental strategy [8, 9]. The factorial experiment is one of the best statistical approaches for identifying optimal parameter setting especially when considering several factors [15].

The objectives of this paper were to: (i) develop a particle swarm optimisation based timetabling (PSOT) tool for solving real-world UCTP in Thailand; (ii) investigate the appropriate parameter settings of PSOs using statistical experimental design and analysis; and (iii) compare the performances of the conventional particle swarm optimisation (CPSO), the standard particle swarm optimisation (SPSO), and the Maurice Clerc particle swarm optimisation (MCPSO) in terms of the solution quality and computational time. The next section of this paper briefly explains the PSO algorithm. Section 3 describes the UCTP followed by the procedures of the PSOT tool in Sect. 4. Section 5 presents the experimental results and analysis followed by conclusions.

2 Particle Swarm Optimisation (PSO)

Particle Swarm Optimisation (PSO) was inspired by swarm behaviour in nature, such as bird flocking, fish schooling, and proposed by Kennedy and Eberhart in 1995 [16]. PSO has become one of the most widely used swarm-intelligence-based algorithms to solve every area in optimisation, computational intelligence, and design applications due to its simplicity and flexibility [17].

According to the conventional PSO procedures, the objective function $F(x)$ at an initial process is specified. Each particle x_i (i = 1, 2,…, P) is generated randomly and evaluated its fitness. The iteration best solution (P_{best}) and the global best solution (G_{best}) are identified by following Eq. (1) [18].

$$P_{best}(i,t) = \underset{k=1,2,\ldots,t}{\arg\min}[F(X_i^k)], i \in (1,2,3,\ldots,P),$$
$$G_{best}(t) = \underset{\substack{i=1,2,\ldots,P \\ k=1,2,\ldots,t}}{\arg\min}[F(X_i^k)], \quad . \tag{1}$$

Where i is index of particles, P is population (particle) size, t is current iteration, and $F(x)$ is objective function. For each generation of conventional PSO, generating new solutions x_i^{t+1} for each particle i is updated by using velocity and position vectors according to Eqs. (2) and (3), respectively [19].

$$V_i^{t+1} = V_i^t + c_1 r_1(P_{best}(i,t) - X_i^k) + c_2 r_2(G_{best}(i,t) - X_i^k). \tag{2}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1}. \tag{3}$$

Where V_i denotes the velocity, c_1 and c_2 are positive constant parameters called acceleration coefficients, and r_1 and r_2 are uniformly distributed random variables within range from 0 to 1 [18]. For standard PSO (called SPSO), generating new solutions x_i^{t+1} is produced by using velocity and position vectors according to Eqs. (4) and (3) [18, 19]. Another variant of PSO introduced by Maurice Clerc is called MCPSO, in which applies Eqs. (5)–(6), and (3) for velocity and position updates [19].

$$V_i^{t+1} = \omega V_i^t + c_1 r_1(P_{best}(i,t) - X_i^k) + c_2 r_2(G_{best}(i,t) - X_i^k). \tag{4}$$

$$V_i^{t+1} = K(V_i^t + c_1 r_1(P_{best}(i,t) - X_i^k) + c_2 r_2(G_{best}(i,t) - X_i^k)). \tag{5}$$

$$K = \frac{2}{\left|2 - \phi - \sqrt{\phi^2 - 4\phi}\right|}, \quad \phi = c_1 + c_2, \quad \phi > 4. \tag{6}$$

Where ω is the inertia weight used to balance the global exploration and local exploitation [18], K is constriction factor to control the velocity of particles [19], and φ is a positive parameter depending on the acceleration coefficients. After preforming the movement strategies of PSO, the fitness value of new solution x_i or $F(x_i)$ is evaluated. The new x_i will be replaced to the P_{best} if the $F(x_i)$ is better than the $F(P_{best})$. Moreover, if the $F(x_i)$ is also better than the $F(G_{best})$, The new x_i will be replaced to the G_{best}. These processes are repeated until getting to the maximum iteration (G) or stop criterion.

Parameters required for any metaheuristic algorithm play a significant role for the algorithm's performance [20]. Parameters have to be tuned due to the optimal values for the parameters depend on the problem domain, the instance, and the computational time to solve [21]. A comprehensive literature survey on Scopus database covering the period from the past to February 2019 focused on the application of PSO on UCTP has been conducted and summarised in Table 1. There are many parameters to be assigned before computational executions including: (i) the number of population (particle) sizes (P); (ii) the acceleration coefficients (c_1 and c_2); and (iii) the inertia weight (ω). Due to

an ad hoc fashion, most of research articles have not reported on the investigation of the best parameter setting of PSO via the appropriate statistical design and analysis.

Table 1. Comprehensive literature review of PSO's parameter settings to solve the UCTP

Authors	PSO variants	PSO's parameter settings to solve the UCTP			
		No. of Particles	*c1*	*c2*	ω
Oswald and Anand Deva Durai [11]	MCPSO	10	2.5	1.5	1/(2 * log(2))
Ahandani and Vakil Baghmisheh [10]	SPSO	60	0.8	0.8	0.3
Chen and Shih [8]	SPSO, PSO	30	2	2	0.8
Kanoh and Chen [9]	SPSO	200	5	2	0.05
Irene, Safaai, Mohd and Zaiton [12]	MCPSO	10	2.8	1.3	1/(2 * log(2))
Irene, Deris and Mohd Hashim [13]	MCPSO	10	2.8	1.3	1/(2 * log(2))
Sheau Fen Ho, Safaai and Siti Zaiton [14]	MCPSO	10	2.8	1.3	1/(2 * log(2))
Range		10–200	0.8–5	0.8–2	0.05–1.66

3 University Course Timetabling Problem (UCTP)

Timetabling courses and examinations in educational institutions is a crucial activity, which assigns appropriate timeslots for students, lecturers, and classrooms [22]. In this research, the real-world university course timetabling data obtained from previous research was considered [23]. Generally, the constraints found in course timetabling can be classified into two types: hard constraints (*HC*) and soft constraints (*SC*) [6]. Hard constraints are the most important and must be satisfied to have a feasible timetable whereas soft constraints are more relaxed as some violations are acceptable. However, the number of SC violations should be minimised [22]. Both *HC* and *SC* constraints considered in this research can be described as following [23].

The considered *HC* were: (i) all lectures within a course must be scheduled and assigned to distinct periods (HC_1); (ii) students and lecturers can only attend one lecture at a time (HC_2); (iii) only one lecture can take place in a room at a given time (HC_3); (iv) lecturers and students must be available for a lecture to be scheduled (HC_4); (v) all courses must be assigned into the classrooms according to their given requirements including building location, room facilities, and room types (HC_5); and (vi) all lectures within a course required consecutive periods must be obeyed (HC_6).

In additions, The considered *SC* were: (i) all courses should be scheduled in the appropriate classroom in order to avoid unnecessary operating or renting costs (hour)

(SC_1); (ii) the courses taught by the given lecturer(s) should be assigned into their available or preferred day and periods in order to save the hiring costs (hour) (SC_2); and (iii) the classrooms should be scheduled in consecutive working periods of a day in order to reduce the number of times to clean or setup after using the rooms (times) (SC_3).

HC_1–HC_6 determine whether potential solutions are feasible. HC_1–HC_3 are the fundamental timetabling constraints (called "event-clash") that can be found in almost all university timetabling problems [6] whilst HC_4–HC_6 are individual requirements and timetabling policy found in many universities in Thailand. This research, SC_1–SC_3 are considered as the objective function, which aim to minimise the total university operating costs considered from the candidate timetables following (7);

$$Minimise \quad F(X_i^k) = W_1SC_1 + W_2SC_2 + W_3SC_3. \tag{7}$$

$$\text{Subject to} : HC_h = 0, \quad \forall h, \tag{8}$$

Equation (7) is the objective functions that evaluate the total university operating costs of the SC_1–SC_3, called $F(X_i^k)$. The weightings (W_1–W_3) for each SC are not restricted and depend upon the user preferences for each institution. In this work, W_1–W_3 were specified at 50 (currency units per hour), 300 (currency units per hour), and 2.5 (currency units per times), respectively. Equation (8) checks a timetable to be a feasible timetable, in which all HCs must be satisfied. Where h is an index relating to the h^{th} hard constraint (h = 1, 2, 3,…, H), where H is the number of hard constraints.

4 Particle Swarm Optimisation Based Timetabling (PSOT) Tool

The PSOT program has been coded in modular style using a general purpose programming language called TCL/TK with C extension [24]. It was developed in order to solve the real world UCTP by using three variants of particle swarm optimisation (PSO) including: (i) conventional PSO (PSO); (ii) standard PSO (SPSO), and (iii) Maurice Clerc PSO (MCPSO). The main procedures within the PSOT program are included in five steps and shown in Fig. 1.

Step 1: after uploading course timetable data and assigning PSO's parameters, the total number of events (n) is determined from the number of teaching periods required for all modules (courses). Then, an event list containing a set of n events was initialised. The event sequence in the list was sorted by using the Largest unpermitted period degree (LUPD) first heuristic [25]. This rule reduces the probability of getting infeasible timetables that generally occur in the process of solution initialisation. Next process is to create an empty timetable or solution. The length of that is calculated taking into account the number of timeslots per day, working day per week, and given classrooms. Then, all events according to the sorted list were inserted into an empty timetable in order to produce an initial population x_i (i = 1, 2, 3, …, P) that represents a set of possible timetables. Next step is to create a new list having the same length of

```
Begin                                                    /*Step 1*/
   Input data and Set PSO's parameters
   Sort a list of courses using heuristic orderings
   Create initial population, x_i (i = 1,2,…,P)
   Generate random keys for each x_i
   While t < Max_Iteration(I) do                         /*Step 2*/
       For (i=1, i<= Max_Pop(P), i++) do
          Pick random numbers: r1, r2 ~ U(0,1)
          If PSO do Update particle's velocity x_i' using Eq.(2)
          If SPSO do Update particle's velocity x_i' using Eq.(4)
          If MCPSO do Update particle's velocity x_i' using Eq.(5)
          Update particle's position x_i' using Eq.(3)
          If (x_i' = an infeasible timetable) do
             Repair x_i' to be a feasible timetable      /*Step 3*/
          Evaluate objective functions F(x_i')           /*Step 4*/
          If F(x_i') > F(P_best),do Replace P_best by the new solution x_i
          If F(x_i') > F(G_best),do Replace G_best by the new solution x_i
   Output results and visualisation of G_best            /*Step 5*/
End
```

Fig. 1. Pseudo code of the PSOT tool

solution, in which each timeslot of a new list is assigned random numbers uniformly distributed between 0 and 1. This process is called a random key technique [26].

Step 2: this is the evolution process of the PSO algorithm. Each particle x_i is selected to update particle's velocity based on the variants of PSO. Particle's velocity of the CPSO, SPSO, and MCPSO are produced by using Eqs. (2), (3), and (4), respectively. Next process, particle's position of x_i' for all variants of PSO are updated by using Eq. (6). Step 3: after evolution process, a new solution (x_i') may be either feasible or infeasible timetable. The repair process was therefore design and embedded in the PSOT program in order to rectify infeasible solutions. Step 4: the solution quality of the x_i' can be measured by using Eq. (7). If $F(x_i')$ is better than $F(P_{best})$, a particle P_{best} is replaced by the x_i' whereas a particle P_{gest} is replaced by the x_i' if its solution quality is better. This processes will be repeated until all particles in the population are improved. Step 5: These processes (Step 2 to Step 4) will be repeated until reach the maximum iterations before showing the best so far results.

5 Experimental Results and Analysis

The objective of the PSOT program is to construct course timetables with the lowest total operating costs (Z). The aims of the computational experiments were to: (i) identify which main factors and their interactions were statistically significant for three variants of PSO; and (ii) explore and compare the performance of the PSO with difference movement strategies including the conventional PSO, the standard PSO (called SPSO), and the Maurice Clerc PSO (called MCPSO). Personal computer with Core 2 Quad 3.00 GHz CPU and 4 GB RAM was used to determine the computational time required to execute experimental runs. Five real-world university course timetabling datasets obtained from the previous research [23] were used in the computational experiment.

5.1 PSO Parameters Investigation

The experiment was aimed to investigate which factors and first level interactions were statistically significant; and to identify the best settings for these factors. The main factors of the PSO, SPSO, and MCPSO included (i) the combination of population (particle) sizes and the number of generation (*PG*), which determines the total number of solutions generated (or amount of search) and the execution time, this computational experiments the value was fixed at 24,000 to limit the time taken for computational search; (ii) the acceleration coefficients (c_1 and c_2); and (iii) the inertia weight (ω) for the SPSO and the MCPSO, excepted the conventional PSO. The experimental design for all PSO's variants, shown in Table 2 was used together with data from dataset number 1. The range of available values for each parameter of PSO were considered from comprehensive literature reviews (shown in Table 1).

Table 2. Experimental factors and levels for the PSO variants

Factors	Levels	SPSO Factor values			PSO Factor values			MCPSO Factor values		
		−1	*0*	*+1*	*−1*	*0*	*+1*	*−1*	*0*	*+1*
PG	3	10 * 2400	60 * 400	200 *120	10 * 2400	60 * 400	200 * 120	10 * 2400	60 * 400	200 * 120
c1	3	0.8	2.8	5	0.8	2.8	5	2.8	4	5
c2	3	0.8	1.3	2	0.8	1.3	2	1.3	1.5	2
ω	3	0.05	0.8	1.66	–	–	–	–	–	–

A full factorial experiment based on the design in Table 2 was considered for this experiment. Thus, the total number of runs required for the PSO and MCPSO would be $3^3 = 27$ runs per replication whereas the total runs for the SPSO would be $3^4 = 81$ runs per replication. The first instant problem was selected and replicated ten times using different random seeds for all PSO's variants. The computational results obtained from the SPSO ($3^4 * 10 = 810$ runs), the PSO ($3^3 * 10 = 270$ runs), and MCPSO ($3^3 * 10 = 270$ runs) were analysed by using a general linear model form of analysis of variance (ANOVA). Table 3 shows the ANOVA table, which shows the source of variation (*Source*), degrees of freedom (*DF*), *F*-value, and *P*-value.

Table 3 shows the PSO parameters in terms of the main effect and first level interactions. *PG*, *PG* * *c2*, and *PG* * ω were statistically significant with a 95% confidence interval. The random seed number (*Seeds*) did not statistically affect the PSO performance. Moreover, the most influential factor in this experiment was *PG* because it had the highest *F*-value. After ANOVA analysis, the appropriate parameter settings for each variant of PSO were determined by using the lowest mean obtained from main effect and interaction plots. For example shown in Fig. 2, the best parameter settings for SPSO are: *PG* = 200 * 120, *c1* = 0.8–5.0, *c2* = 0.8, and ω = 0.8. Moreover, the best settings for PSO parameters are: *PG* = 200 * 120, *c1* = 0.8, and *c2* = 0.8. However, the best settings for MCPSO are: *PG* = 10 * 2,400, *c1* = 2.8, and *c2* = 1.5.

Table 3. ANOVA analysis of PSOs parameters

Source	DF	SPSO		PSO		MCPSO	
		F-value	*P*-value	*F*-value	P-value	*F*-value	*P*-value
PG	2	92.15	0.000	19.36	0.000	1.490	0.228
c1	2	0.00	1.000	1.17	0.311	0.600	0.548
c2	2	2.50	0.083	0.80	0.449	0.580	0.563
ω	2	0.90	0.407	–	–	–	–
Seeds	9	1.65	0.097	1.02	0.426	1.150	0.326
PG * *c1*	4	0.00	1.000	0.27	0.899	0.730	0.574
PG * *c2*	4	8.04	**0.000**	0.24	0.917	1.410	0.230
c1 * *c2*	4	0.00	1.000	0.53	0.714	0.490	0.740
PG * ω	4	5.23	**0.000**	–	–	–	–
c1 * ω	4	0.00	1.000	–	–	–	–
c2 * ω	4	1.47	0.210	–	–	–	–
Error	768						
Total	809						

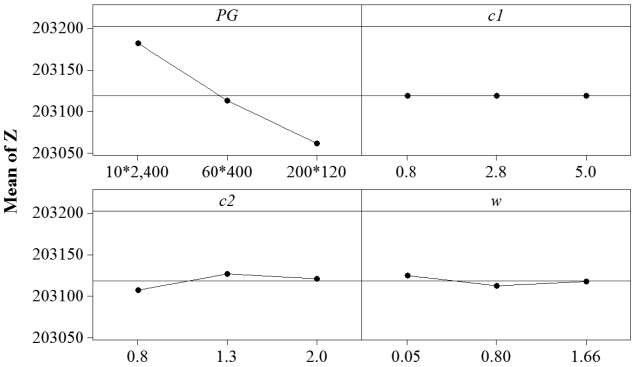


Fig. 2. Example of SPSO's main effect plots of *PG*, *c1*, *c2*, and ω factors

5.2 Performance of PSO's Variants

The objective of this experiment was to explore and compare the performance of the PSO with difference movement strategies including PSO, SPSO, and MCPSO in term of the quality of the solutions. The appropriate parameter settings for three variants of PSO were adopted from previous experiment. Five course timetabling datasets were used to test and compare the performance of these algorithms to find the course timetable with the lowest penalty *Z*. The computational run for each instance was repeated ten times by using different random seeds. The computational results were analysed in terms of *Avg* (currency unit), *SD*, and *Time* (minute unit) as shown in Table 4.

Table 4. Performance comparisons between three variants of PSO

Dataset No.	SPSO			PSO			MCPSO		
	Avg	*SD*	*T*	*Avg*	*SD*	*T*	*Avg*	*SD*	*T*
1	203,098.75	41.28	7.37	**202,944.90**	410.98	6.60	203,030.80	134.34	6.44
2	382,899.98	364.81	24.23	382,918.73	218.27	21.61	**382,744.20**	391.59	22.82
3	306,721.83	213.26	41.03	306,854.75	163.04	35.26	**305,400.33**	596.58	36.55
4	310,214.98	382.26	31.10	310,000.63	464.64	30.05	**308,821.35**	597.81	26.75
5	**492,891.90**	409.66	50.04	492,910.05	197.49	41.72	493,002.15	500.36	47.81

From Table 4, it can be seen that the average values of the best so far solutions (timetables) generated by MCPSO were better than those values generated by both PSO and SPSO for most problems. The PSO outperformed the other methods for problem number 1 whereas the SPSO outperformed both PSO and MCPSO for problem number 5. Moreover, the SD values and the averages of the computational times obtained from both methods were moderately different for all problems.

6 Conclusions

A particle swarm optimisation based timetabling (PSOT) tool has been developed in order to solve the real-world university course timetabling problems. The conventional PSO, the SPSO, and the MCPSO were embedded in the PSOT program for constructing the desirable timetables with minimal objective function. Full factorial experimental designs and ANOVA were adopted to investigate the statistically influential factors for each variant of PSO before identifying its best parameter settings. It was found that the PSOs' parameters in terms of the main effect and interactions including *PG*, *PG* * *c2*, and *PG* * ω were statistically significant with a 95% confidence interval. The most influential factor in this experiment was *PG* because it had the highest *F*-value. Moreover, the MCPSO outperformed the other variants of PSO for most datasets whereas the SPSO and PSO outperformed the other variants only one dataset. However, the computational times required by the proposed PSO variants were moderately difference.

Acknowledgements. This work was part of research project supported by the Thailand Research Fund (TRF) and Office of the Higher Education Commission (OHEC) under grant number MRG6080066.

References

1. Jat, S.N., Yang, S.: A guided search non-dominated sorting genetic algorithm for the multi-objective university course timetabling problem. In: Merz, P., Hao, J.-K. (eds.) EvoCOP 2011. LNCS, vol. 6622, pp. 1–13. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20364-0_1
2. Thepphakorn, T., Pongcharoen, P., Hicks, C.: Modifying regeneration mutation and hybridising clonal selection for evolutionary algorithms based timetabling tool. Math. Probl. Eng. **2015**, 16 (2015). Article Number 841748
3. Lutuksin, T., Pongcharoen, P.: Best-worst ant colony system parameter investigation by using experimental design and analysis for course timetabling problem. In: 2nd International Conference on Computer and Network Technology, ICCNT 2010, pp. 467–471 (2010)
4. MirHassani, S.A.: A computational approach to enhancing course timetabling with integer programming. Appl. Math. Comput. **175**, 814–822 (2006)
5. Yang, X.-S.: Swarm intelligence based algorithms: a critical analysis. Evol. Intel. **7**, 17–28 (2014)
6. Lewis, R.: A survey of metaheuristic-based techniques for university timetabling problems. OR Spectrum **30**, 167–190 (2008)
7. Rana, S., Jasola, S., Kumar, R.: A review on particle swarm optimization algorithms and their applications to data clustering. Artif. Intell. Rev. **35**, 211–222 (2011)
8. Chen, R.M., Shih, H.F.: Solving university course timetabling problems using constriction particle swarm optimization with local search. Algorithms **6**, 227–244 (2013)
9. Kanoh, H., Chen, S.: Particle Swarm Optimization with Transition Probability for Timetabling Problems. In: Tomassini, M., Antonioni, A., Daolio, F., Buesser, P. (eds.) ICANNGA 2013. LNCS, vol. 7824, pp. 256–265. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37213-1_27
10. Ahandani, M.A., Vakil Baghmisheh, M.T.: Hybridizing genetic algorithms and particle swarm optimization transplanted into a hyper-heuristic system for solving university course timetabling problem. WSEAS Trans. Comput. **12**, 128–143 (2013)
11. Oswald, C., Anand Deva Durai, C.: Novel hybrid PSO algorithms with search optimization strategies for a university course timetabling problem. In: Proceedings of the 5th International Conference on Advanced Computing, ICoAC 2013, pp. 77–85 (2014)
12. Irene, H.S.F., Safaai, D., Mohd, H., Zaiton, S.: University course timetable planning using hybrid particle swarm optimization. In: Proceedings of the 1st ACM/SIGEVO Summit on Genetic and Evolutionary Computation, GEC 2009, pp. 239–245 (2009)
13. Irene, S.F.H., Deris, S., Mohd Hashim, S.Z.: A combination of PSO and local search in university course timetabling problem. In: Proceedings - 2009 International Conference on Computer Engineering and Technology, ICCET 2009, pp. 492–495 (2009)
14. Sheau Fen Ho, I., Safaai, D., Siti Zaiton, M.H.: A study on PSO-based university course timetabling problem, pp. 648–651 (2009)
15. Montgomery, D.C.: Design and Analysis of Experiments. Wiley, Hoboken (2012)
16. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
17. Yang, X.-S.: Nature-Inspired Optimization Algorithms. Elsevier, Amsterdam (2014)
18. Zhang, Y., Wang, S., Ji, G.: A comprehensive survey on particle swarm optimization algorithm and its applications. Math. Prob. Eng. **2015**, 38 (2015)
19. Thangaraj, R., Pant, M., Abraham, A., Bouvry, P.: Particle swarm optimization: hybridization perspectives and experimental illustrations. Appl. Math. Comput. **217**, 5208–5226 (2011)

20. Chiroma, H., Herawan, T., Fister, I., Fister, I., Abdulkareem, S., Shuib, L., Hamza, M.F., Saadi, Y., Abubakar, A.: Bio-inspired computation: recent development on the modifications of the cuckoo search algorithm. Appl. Soft Comput. **61**, 149–173 (2017)
21. Talbi, E.-G.: Metaheuristics: From Design to Implementation. Wiley, Hoboken (2009)
22. Thepphakorn, T., Pongcharoen, P., Hicks, C.: An ant colony based timetabling tool. Int. J. Prod. Econ. **149**, 131–144 (2014)
23. Thepphakorn, T., Pongcharoen, P., Vitayasak, S.: A New Multiple Objective Cuckoo Search for University Course Timetabling Problem. In: Sombattheera, C., Stolzenburg, F., Lin, F., Nayak, A. (eds.) MIWAI 2016. LNCS (LNAI), vol. 10053, pp. 196–207. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49397-8_17
24. Ousterhout, J.K., Jones, K.: Tcl and the Tk Toolkit, 2nd edn. Addison-Wesley, Boston (2009)
25. Thepphakorn, T., Pongcharoen, P.: Heuristic ordering for ant colony based timetabling tool. J. Appl. Oper. Res. **5**, 113–123 (2013)
26. Khadwilard, A., Chansombat, S., Thepphakorn, T., Thapatsuwan, P., Chainate, W., Pongcharoen, P.: Application of firefly algorithm and its parameter setting for job shop scheduling. J. Ind. Technol. **8**, 49–58 (2012)