รายงานวิจัยฉบับสมบูรณ์

การพัฒนาวิธีคอลัมน์เจนเนอร์เรชันเพื่อใช้ในการแก้ปัญหาในภาคอุตสาหกรรมโลจิสติกส์
ที่มีขนาดใหญ่และซับซ้อน

โดย

นายณกร อินทร์พยุง
คณะโลจิสติกส์ มหาวิทยาลัยบูรพา

กันยายน 2551

สัญญาเลขที่ MRG5080076

รายงานวิจัยฉบับสมบูรณ์

การพัฒนาวิธีคอลัมน์เจนเนอร์เรชันเพื่อใช้ในการแก้ปัญหาในภาคอุตสาหกรรมโลจิสติกส์
ที่มีขนาดใหญ่และซับซ้อน

โดย

นายณกร อินทร์พยุง
คณะโลจิสติกส์ มหาวิทยาลัยบูรพา

สนับสนุนโดยสำนักงานกองทุนสนับสนุนการวิจัย

(ความเห็นในรายงานนี้เป็นของผู้วิจัย สกว.ไม่จำเป็นต้องเห็นด้วยเสมอไป)

กิตติกรรมประกาศ

งานวิจัยฉบับนี้สำเร็จลงได้ด้วยความกรุณาจาก รศ. ดร. วิโรจน์ ศรีสุรภานนท์ นักวิจัยที่ปรึกษา ที่กรุณาให้คำปรึกษาแนะนำแนวทางการทำวิจัย ตลอดจนแก้ไขข้อบกพร่องต่างๆ ด้วยความเอาใจใส่ ทำให้งานวิจัยฉบับนี้เสร็จลุล่วงได้ ผู้วิจัยรู้สึกซาบซึ้ง และระลึกถึงพระคุณเป็นอย่างยิ่ง จึงกราบขอบพระคุณมา ณ ที่นี้ด้วย ขอขอบคุณ นายเกรียงศักดิ์ วณิชชากรพงศ์ นิสิตปริญญาโท ที่ให้ความช่วยในด้านการพัฒนาโปรแกรมคอมพิวเตอร์ ด้วยความมุ่งมั่น ตั้งใจ อย่างสม่ำเสมอ

ขอบขอบคุณสำนักงานคณะกรรมการอุดมศึกษาและสำนักงานกองทุนสนับสนุนการวิจัยที่สนับสนุนทุนพัฒนาศักยภาพในการทำงานวิจัยของอาจารย์รุ่นใหม่ รวมทั้งลูกๆและภรรยา ที่คอยสนับสนุนและให้กำลังใจมาโดยตลอด

นายณกร อินทร์พยุง

Project Code : MRG5080076
(รหัสโครงการ)

Project Title : Improving column generation method for solving large scale logistics problems
(ชื่อโครงการ) การพัฒนาวิธีคอลัมน์เจนเนอร์เรชันเพื่อใช้ในการแก้ปัญหาในภาคอุตสาหกรรมโลจิสติกส์ที่มีขนาดใหญ่และซับซ้อน

Investigator : Assist. Prof. Nakorn Indra-Payoong
(ชื่อนักวิจัย) ผู้ช่วยศาสตราจารย์ ดร. ณกร อินทร์พยุง

E-mail Address : nakorn@buu.ac.th

Project Period : 1 ปี (1 ธันวาคม 2549 – 30 พฤศจิกายน 2550)
(ระยะเวลาโครงการ)

Abstract

Transport and logistics operation is a large and complex decision problem, which cannot simply be formulated into a mathematical form and cannot be solved to optimality within a reasonable computer-runtime; in other words the solving algorithm fails to prove if the optimal solution exists.

This research proposes a constrained local search (CLS) to solve several large and complex logistics problems. The proposed algorithm is relatively flexible and can find the good enough solution very quickly. The problem in a constraint satisfaction problem form can handle additional constraints commonly found in real cases directly without modifying any structures of the algorithm. We apply CLS and column generation method to solve bus and crew scheduling problem as well as the pickup and delivery problem. The hybrid algorithm is non-domain specific and can be applied to obtain a good quality solution within a viable time.

The computational experiments are performed to evaluate the performance of the proposed method compared with the best-known algorithms in the literature. The results have shown that in all test cases, the solution costs from the hybrid algorithm are better than those from CLS alone and are as good as the best-known solutions.

Keywords : Optimization methods, Hybrid column generation, Local search, Large-scale logistics problems

บทคัดย่อ

การจัดการขนส่งและโลจิสติกส์นั้นเป็นปัญหาการตัดสินใจที่มีขนาดใหญ่และมีความซับซ้อน ทำให้ไม่สามารถกำหนดปัญหาให้อยู่ในรูปแบบของคณิตศาสตร์ได้โดยง่ายและยังไม่สามารถหาคำตอบที่ดีที่สุดนั้นได้ในเวลาที่จำกัด หรือในบางครั้งอาจหาคำตอบของปัญหานั้นไม่ได้เลย

งานวิจัยนี้นำเสนอวิธี Constrained local search (CLS) ในการแก้ปัญหาที่สามารถหาคำตอบที่ใกล้เคียงกับคำตอบที่ดีที่สุดได้ในเวลาอันรวดเร็ว และมีความยืดหยุ่นสามารถประยุกต์ใช้กับปัญหาการจัดการโลจิสติกส์ได้หลายปัญหา วิธีดังกล่าวสามารถเพิ่มและลดเงื่อนไขในการหาคำตอบได้ โดยไม่จำเป็นต้องปรับเปลี่ยนขั้นตอนในการหาคำตอบ งานวิจัยประยุกต์ใช้วิธีพันธ์ผสมซึ่งรวมวิธี CLS และวิธีคอลัมน์เจนเนอร์เรชันมาใช้ในการแก้ปัญหา ปัญหาการจัดตารางการเดินรถและพนักงานงานเดินรถ และปัญหาการจัดเส้นทางการรับส่งสินค้าของยานพาหนะ

จากผลการทดลองเพื่อประเมินประสิทธิภาพของวิธีที่พัฒนาขึ้นโดยเปรียบเทียบกับวิธีที่ดีที่สุด (Best known methods) ที่แสดงอยู่ในผลงานวิจัยที่ผ่านมาพบว่า วิธีพันธ์ผสมที่พัฒนาขึ้นนั้นมีประสิทธิภาพมากกว่าวิธี CLS โดยลำพัง และมีประสิทธิภาพดีทัดเทียมกับวิธีที่ดีที่สุดในปัจจุบัน

คำหลัก วิธีแก้ปัญหาการตัดสินใจ วิธีพันธ์ผสม วิธีโลคอลเสริช ปัญหาโลจิสติกส์ขนาดใหญ่

1. บทนำ

ระบบโลจิสติกส์นั้นนำมาสู่การจัดการที่มีประสิทธิภาพด้วยผลกำไรที่มากขึ้น หรือต้นทุนในการดำเนินการที่ลดลง โดยทั่วไป ปัญหาการจัดการโลจิสติกส์เป็นปัญหาการตัดสินใจที่มีขนาดใหญ่และซับซ้อน การหาคำตอบที่ดีที่สุดของการตัดสินใจในการแก้ปัญหาดังกล่าวนั้นมีความเป็นไปได้ยากมาก หรืออาจเป็นไปไม่ได้เลย

โดยทั่วไป ปัญหาการตัดสินใจในภาคอุตสาหกรรมโลจิสติกส์ที่มีขนาดใหญ่และมีความซับซ้อน จะอาศัยขั้นตอนวิธีทางคอมพิวเตอร์ที่เรียกว่า"ฮิวริสติกส์" มาใช้ในการหาคำตอบของการตัดสินใจ แม้ว่าวิธีฮิวริสติกส์จะสามารถหาคำตอบที่มีคุณภาพดีได้ภายในระยะเวลาอันรวดเร็ว คำตอบที่ได้ก็ไม่ได้ถูกประกันว่า เป็นคำตอบของตัดสินใจที่ดีที่สุด อีกแนวทางหนึ่งในการแก้ปัญหาการตัดสินใจคือการใช้วิธี Exact (หรือ Optimization method) เพื่อที่จะหาคำตอบที่ดีที่สุดของการตัดสินใจ และประกันการพบคำตอบที่ดีที่สุดในทุกๆครั้งของการรันโปรแกรม อย่างไรก็ตาม ข้อจำกัดที่สำคัญที่สุดของวิธีนี้ คืออัลกอริทึมจะใช้เวลาในการค้นหาคำตอบที่ดีที่สุดเป็นเวลานานมาก (*NP*-hard problem) จึงทำให้ วิธีดังกล่าวไม่สามารถนำไปใช้ในทางปฏิบัติ ในการแก้ปัญหาการตัดสินใจที่มีขนาดใหญ่และซับซ้อนได้

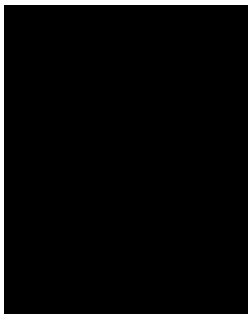
ในการศึกษานี้ ผู้วิจัยจะทำการวิเคราะห์และออกแบบขั้นตอนทางคอมพิวเตอร์ (หรืออัลกอริทึม) โดยประยุกต์ใช้วิธีคอลัมน์เจ็นเนอร์เรชัน (Column generation method) มาทำการพัฒนา ปรับปรุงและสร้างเป็นองค์ความรู้ใหม่ เพื่อให้สามารถหาคำตอบของปัญหาการตัดสินใจได้อย่างมีประสิทธิภาพ โดยในการศึกษานี้จะใช้ปัญหาการจัดตารางเวลาและเส้นทางสำหรับยานพาหนะมาใช้เป็นกรณีศึกษา ผลจากการวิจัย สามารถนำไปประยุกต์ใช้กับปัญหาโลจิสติกส์ที่เกิดขึ้นในภาคอุตสาหกรรมต่างๆ เพื่อช่วยให้บริษัทสามารถลดค่าใช้จ่ายในการดำเนินงาน และเพิ่มศักยภาพในการแข่งขันกับบริษัทต่างประเทศได้

2. วิธีดำเนินงานวิจัย

วิธี CLS เป็นวิธีฮิวริสติกส์วิธีหนึ่งง่าย และมีความยืดหยุ่นในการกำหนดรูปแบบปัญหาและเงื่อนไขของปัญหา ซึ่งสามารถเพิ่มและลดเงื่อนไขของปัญหาโดยไม่จำเป็นต้องกำหนดรูปแบบและขั้นตอนการแก้ปัญหาใหม่ เนื่องจากวิธี CLS นั้นเป็นวิธีการกำหนดค่าความขัดแย้ง (Violation) ให้กับฟังก์ชันวัตถุประสงค์และเงื่อนไขต่าง ๆ เพื่อทำการเปรียบเทียบคุณภาพของคำตอบ ดังนั้นเมื่อมีเงื่อนไขเพิ่มขึ้นก็สามารถกำหนดเขียนเป็นสมการในการกำหนดค่าความขัดแย้งเข้าไปสู่ปัญหาเดิม หรือเมื่อต้องการลดเงื่อนไขบางข้อก็สามารถนำสมการที่คำนวณหาค่าความขัดแย้งของเงื่อนไขดังกล่าวออกได้โดยไม่จำเป็นต้องแก้ไขขั้นตอนการแก้ปัญหา

วิธี CLS นั้นเราจะกำหนดให้ปัญหาอยู่ในรูปแบบของ Set partitioning คือการกำหนดปัญหาให้อยู่ในรูปของตารางเมตริกซ์ โดยตัวแปรภายในตารางจะมีค่าได้คือ 0 หรือ 1 นั้นคือตัวแปรในการตัดสินใจแบบไม่ต่อเนื่อง โดยมีข้อกำหนดคือ ผลรวมของตัวแปรในแถว หรือคอลัมน์ อย่างใดอย่างหนึ่งต้องมีค่าเท่ากับ 1 ดังสมการ (2.1)

$$\begin{aligned} &\min cx \\ &Ax = 1 \\ &x = \{0,1\} \end{aligned} \tag{2.1}$$



ภาพที่ 2.1 การกำหนดรูปแบบปัญหาแบบ set partitioning

จากภาพที่ 2.1 แสดงตัวอย่างการกำหนดปัญหา Job assignment ให้อยู่ในรูปของ set partitioning โดยมีจำนวนงานทั้งหมด 7 งานคือ i1… i7 และมีพนักงานทั้งหมด 3 คนคือ j1… j3 กำหนดตารางเมตริกซ์โดยแทนแถวด้วยงาน และแทนคอลัมน์ด้วยพนักงาน โดยตัวแปรในการตัดสินใจจะมีขนาดเท่ากับจำนวนแถวคูณด้วยจำนวนคอลัมน์ จะได้ $x_{ij} = \{0,1\}$ โดย $i = \{1,...,7\}$ และ $j = \{1,...,3\}$ เมื่อ $x_{ij} = 1$ จะหมายความว่างาน i ถูกปฏิบัติงานด้วยพนักงาน j ซึ่งเงื่อนไขก็คือ งาน i ถูกปฏิบัติโดยพนักงานได้คนเดียวเท่านั้น ซึ่งเป็นไปตามเงื่อนไขของ set partitioning คือ $\sum_{j\in\{1,...,3\}} x_{ij} = 1 \quad \forall i \in \{1,...,7\}$ จากรูปจะกำหนดให้พนักงาน j1 ปฏิบัติงาน i2 และ i5 พนักงาน j2 ปฏิบัติงาน i1, i4 และ i7 และพนักงาน j3 ปฏิบัติงาน i3 และ i6 ซึ่งในตัวอย่างยังไม่ได้คำนึงถึงเงื่อนไขอื่น

หลังจากการกำหนดรูปแบบของปัญหาให้อยู่ในรูปของ Set partitioning แล้ว จะทำการกำหนดรูปแบบของฟังก์ชันวัตถุประสงค์ และเงื่อนไขต่าง ๆ ให้อยู่ในรูปของค่าความขัดแย้ง ซึ่งจะแบบค่าความขัดแย้งออกเป็นสองแบบหลัก ๆ คือ ค่าความขัดแย้งของเงื่อนไขหลัก (hard violation) และค่าความขัดแย้งของเงื่อนไขรอง (soft violation) โดยจะกำหนดให้ฟังก์ชันวัตถุประสงค์นั้นเป็นค่าความขัดแย้งของเงื่อนไขรอง และกำหนดให้เงื่อนไขต่าง ๆ เป็นค่า ความขัดแย้งของเงื่อนไขหลัก ดังสมการ (2.2)

$$Ax \le b \Rightarrow V = \max(0, Ax - b) \tag{2.2}$$

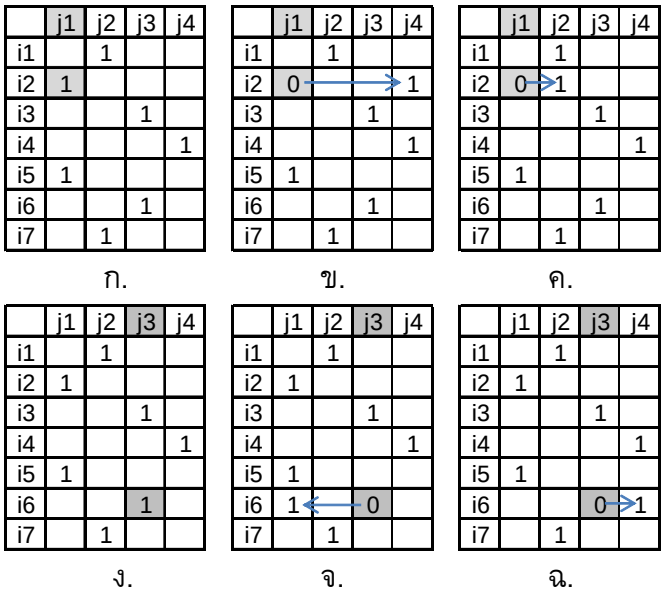
กำหนดให้ V คือ ค่าความขัดแย้ง ในส่วนของค่าความขัดแย้งของเงื่อนไขหลัก นั้นเมื่อคำนวณคำตอบได้แล้วนั้น จะต้องมีค่าเท่ากับ 0 นั้นหมายความว่าคำตอบที่ได้นั้นต้องไม่ผิดเงื่อนไขใดๆ ส่วนค่าความขัดแย้งของเงื่อนไขรอง นั้นเราไม่สามารถรู้ขีดจำกัดล่างของคำตอบได้ว่ามีค่าเท่าใด เราจึงใช้ค่าที่ต่ำที่สุดในอุดมคติในการเปรียบเทียบคือ 0 เมื่อค่าความขัดแย้งของเงื่อนไขรอง เข้าใกล้ศูนย์คำตอบที่ได้นั้นจะเป็นคำตอบที่ดีขึ้น (ในกรณีเราหาค่าที่น้อยที่สุด)

```
// Algorithm: CLS
// INPUT: soft and hard constraints
// OUTPUT: a best feasible solution found
1    A := initial solution assignment
2    A_b := best solution
3    V_b := the best violation
4    WHILE (iteration < stopping criterion) DO
5       nc := Random(numcol)
6       V_g := MAX_INTEGER
7       FOR i ← 1 TO nc
8          C_fst := select-columns (A)
9          P := select-variables(C_fst)
10         V_l := MAX_INTEGER
11         FOR j ← 1 TO Random(numcol)
12            A' := flip(C_fst, C_j, P)
13            H' := total hard violation(A')
14            V' := total violation(A')
15            IF H' = 0 AND V' < V_b
16               A_b ← A'
17               V_b ← V'
18            END IF
19            IF V' < V_l THEN
20               A_l ← A'
21            END IF
22         END FOR
23         IF V_l < V_g THEN
24            A_g ← A_l
25            V_g ← V_l
26         END IF
27      END FOR
28      A ← A_g
29   END WHILE
```

ภาพที่ 2.2 ขั้นตอนการทำงานของวิธี CLS

จากภาพที่ 2.2 แสดงขั้นตอนการทำงาน และการย้ายพื้นที่การหาคำตอบของวิธี CLS จากรูปกำหนดให้เงื่อนไขของ Set partitioning อยู่ในแนวของแถว คือ ผลรวมในแนวแถวเดียวกันจะต้องมีค่าเท่ากับ 1 โดยขึ้นตอนเริ่มจากการกำหนดคำตอบเริ่มต้นที่ไม่ผิดเงื่อนไขของ Set partitioning แต่ไม่จำเป็นต้องตรงตามเงื่อนไขอื่น จากนั้นทำการสุ่มคอลัมน์แรกที่มีค่าตัวแปรในคอลัมน์มีค่าเท่ากับ 1 ขึ้นมา จากนั้นสุ่มตัวแปรที่มีค่าเท่ากับ 1 ภายในคอลัมน์แรก จากนั้นสุ่มจำนวนคอลัมน์ในการสลับค่าไม่เกินจำนวนคอลัมน์ที่เหลือ จากนั้นสุ่มคอลัมน์ที่สองจากคอลัมน์ที่เหลือ ลองสลับค่า (Trial flip) และคำนวณหาค่าความขัดแย้งของแต่ละคำตอบ และสุ่มคอลัมน์ที่สองเท่ากับคำนวณที่สุ่มได้จากนั้นลองสลับค่าตัวแปร คำนวณหาค่าความขัดแย้ง เพื่อเปรียบเทียบคำตอบจะได้ของคอลัมน์ที่สอง จะได้คอลัมน์ที่สองที่สลับค่าแล้วได้คำตอบที่ดีที่สุด จากนั้นลองสุ่มเลือกคอลัมน์แรกใหม่ทำซ้ำวิธีที่กล่าวมาในจำนวนครั้งจากการสุ่มมา จากนั้นเปรียบเทียบว่าคอลัมน์แรก และคอลัมน์ที่สองใดที่สลับค่าแล้วได้คำตอบที่ดีที่สุด จะกำหนดคำตอบนั้นเป็นคำตอบปัจจุบัน ในการหาคำตอบของ CLS นั้นจะถือว่าคำตอบนั้นเป็นคำตอบที่เป็นไปได้ก็ต่อเมื่อค่า ความขัดแย้งของเงื่อนไขหลัก มีค่าเท่ากับ 0 แต่ในกรณีที่ค่าความขัดแย้งของเงื่อนไขหลัก มีค่ามากกว่า 0 หมายถึงคำตอบที่ได้นั้นยังไม่ถูกต้องตาม

เงื่อนไข และเมื่อค่าความขัดแย้งของเงื่อนไขหลัก มีค่ามาขึ้นเท่ากับว่าคำตอบนั้นจะผิดเงื่อนไขมากตามไปด้วย ในส่วนของค่าความขัดแย้งของเงื่อนไขรอง นั้นจะมีค่าน้อยมากเท่าไรยิ่งได้คำตอบที่ดี (กรณีที่หาคำตอบที่ต่ำที่สุด) อย่างไรก็ตามแม้ค่าความขัดแย้งของเงื่อนไขรอง จะมีค่าต่ำมาก แต่ค่าความขัดแย้งของเงื่อนไขหลัก มีค่ามากกว่า 0 คำตอบดังกล่าวนั้นก็เป็นคำตอบที่ผิดเงื่อนไขและเป็นคำตอบที่เป็นไปไม่ได้ (Infeasible solution) ดังนั้นในการลองสลับค่าหรือย้ายคำตอบทุกครั้ง จะทำการตรวจสอบว่าค่าความขัดแย้งที่ได้จากคำตอบนั้น โดยถ้าค่าความขัดแย้งของเงื่อนไขหลักมีค่าเท่ากับ 0 และความขัดแย้งของเงื่อนไขรองมีค่าน้อยกว่าค่าความขัดแย้งของเงื่อนไขรอง ที่น้อยที่สุดของคำตอบที่พบมา จะแทนที่คำตอบที่ดีที่สุดด้วยคำตอบปัจจุบัน และเมื่อการสลับค่าครั้งใดมีค่าต่ำกว่าก็จะแทนที่ไปเรื่อยๆ



ภาพที่ 2.3 ตัวอย่างการย้ายคำตอบของวิธี CLS

จากภาพที่ 2.3 เป็นตัวอย่างของปัญหา job assignment กำหนดให้มีพนักงาน 4 คน คือ j1,..., j4 และมีจำนวน 7 งานคือ i1,..., i7 กำหนดรูปแบบปัญหาให้อยู่ในรูปของ set partitioning จากนั้นกำหนดคำตอบเริ่มต้น โดยการสุ่มค่าของตัวแปรที่ตรงตามเงื่อนไขของ set partitioning ดังภาพที่ 3.3 ก. จากนั้นสุ่มเลือกคอลัมน์แรกในการลองสลับได้คอลัมน์ j1 จากนั้นสุ่มเลือกค่าตัวแปรที่เป็น 1 ในคอลัมน์แรกได้ (j1, i2) ดังแถบสีในภาพที่ 3.3 ก. จากนั้นสุ่มจำนวนในการลองสลับกับคอลัมน์ที่สอง ว่าจะลองสลับกี่ครั้ง จากตัวอย่างกำหนดให้สุ่มได้ 2 ครั้ง โดยครั้งสุ่มเลือกคอลัมน์ที่เหลืออยู่ ได้คอลัมน์ j4 เป็นคอลัมน์ที่สอง จากนั้นลองสลับดังภาพที่ 2.3 ข. และสุ่มเลือกคอลัมน์ที่สองครั้งที่สองได้คอลัมน์ j2 และลองสลับ จากนั้นเลือกคอลัมน์แรกใหม่เท่าจำนวนครั้งที่สุ่มขึ้น ดังภาพที่ 2.3 ง. เป็นการสุ่มคอลัมน์แรกครั้งที่สอง ได้คอลัมน์ j3 และตัวแปร (j3, i6) สุ่มจำนวนการเลือกคอลัมน์ที่สอง และสุ่มคอลัมน์ที่สองและสลับตามจำนวนที่สุ่มได้ ดังภาพที่ 2.3 จ. และ 2.3 ฉ. จากนั้นเปรียบเทียบค่าความขัดแย้งของคำตอบจาก 2.3 ข. ค. จ. และ ฉ. ว่าคำตอบใดมีค่าความขัดแย้ง

น้อยสุด จะสลับคำตอบหรือย้ายคำตอบไปยังคำตอบดังกล่าว ทั้งนี้สังเกตว่าเราจะเปรียบเทียบค่าความขัดแย้งที่ได้จากการลองสลับเท่านั้น จะไม่เปรียบเทียบกับคำตอบเดิมก่อนการลองสลับคือภาพที่ 2.3 ก. ทั้งนี้เพื่อการย้ายคำตอบเพื่อแก้ปัญหา Local optimal

วิธีคอลัมน์เจนเนอร์เรชั่นสำหรับปัญหาการตัดสินใจที่มีขนาดใหญ่และซับซ้อน

วิธี CG นั้นเป็นวิธีการแก้ปัญหาที่หาคำตอบที่ดีที่สุด ทั้งนี้ในกรณีที่ปัญหาย่อยนั้นต้องหาคำตอบที่ดีที่สุดได้เช่นกัน โดยวิธี CG นั้นอาศัยวิธีโปรแกรมเชิงเส้น (Linear programming: LP) ในการหาคำตอบ ซึ่งวิธี LP นั้นมีการคำนวณที่รวดเร็ว แต่อย่างไรก็ตามเมื่อตัวแปรมีจำนวนมากๆ จะทำให้การคำนวณใช้เวลานานเช่นกัน ข้อสำคัญอีกสิ่งหนึ่งคือในบางปัญหาไม่สามารถหาค่าสัมประสิทธิ์ของตัวแปรทั้งหมดได้ ดังนั้นทำให้ไม่สามารถใช้วิธีโปรแกรมเชิงเส้นในการหาคำตอบได้ วิธี CG เป็นวิธีหนึ่งในการแก้ปัญหาดังกล่าว กล่าวคือจะคำนวณหาคำตอบโดยใช้วิธี LP ของคอลัมน์จำนวนหนึ่งจากจำนวนคอลัมน์ทั้งหมด จากนั้นคำนวณหาคอลัมน์ใหม่เพื่อเพิ่มเข้าไปจนกระทั่งไม่สามารถหาคอลัมน์มาเพิ่มแล้วทำให้คำตอบที่ได้ดีขึ้นอีก จะถือว่าคำตอบที่ได้จากคอลัมน์กลุ่มนั้น เป็นคำตอบที่ดีที่สุดของคอลัมน์ทั้งหมด

ขั้นตอนการทำงานของคอลัมน์เจนเนอร์เรชัน

วิธี CG เป็นวิธีที่เพิ่มประสิทธิภาพการคำนวณของวิธี LP กับปัญหาที่มีขนาดใหญ่และมีขนาดของตัวแปรจำนวนมาก โดยวิธี CG นั้นจะเป็นการหาคำตอบจากกลุ่มตัวแปร หรือคอลัมน์ จากนั้นจะคำนวณหาตัวแปร หรือคอลัมน์ที่ความเป็นไปได้ที่จะได้คำตอบของฟังก์ชันวัตถุประสงค์ที่ดีขึ้นจากการหาค่า Negative reduced cost โดยจะแบ่งปัญหาออกเป็นสองส่วน คือ Master problem (MP) และ Subproblem (SP) โดยการคำนวณหาคำตอบนั้นจะคำนวณจาก Restricted master problem (RMP) ซึ่งปัญหายังเป็นรูปแบบเดียวกับ MP แต่คำนวณเฉพาะซับเซตของคอลัมน์ทั้งหมดเท่านั้น และในส่วนของ SP จะเป็นปัญหาเพื่อหาคอลัมน์ให้กับ RMP โดยฟังก์ชันวัตถุประสงค์ของ SP คือ ค่า reduced cost ของคอลัมน์ใหม่ ซึ่งได้มาจากการนำค่าดูอัลไพซ์ (dual price) ของ RMP มาปรับเปลี่ยนค่าใช้จ่ายในการหาคำตอบของ SP

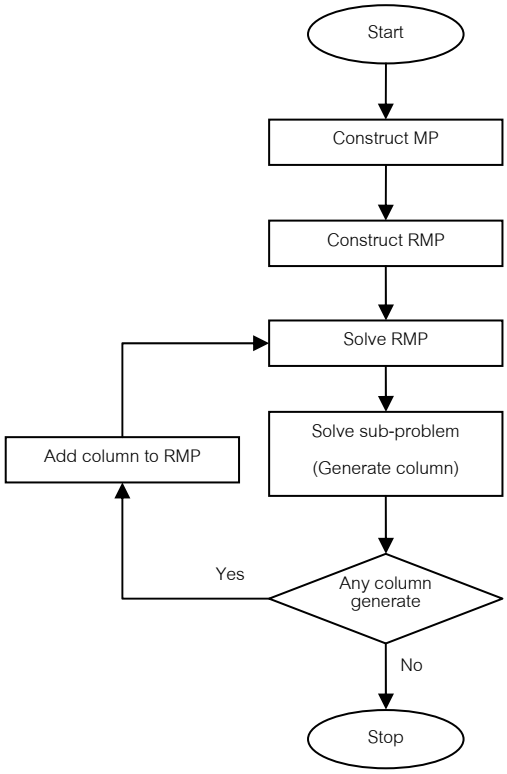
$$
\begin{aligned}
\square^{*}_{MP} &:= \min \sum_{j \in J} c_j \lambda_j \\
\sum_{j \in J} a_j \lambda_j &\geq b \qquad (2.3) \\
\lambda_j &\geq 0,\ j \in J
\end{aligned}
$$

จากสมการ (2.3) นั้นคือการกำหนด MP ในแต่ละรอบการหาคำตอบ LP โดยวิธี Simplex เราจะนำตัวแปร Non-basic ออกและใส่ตัวแปร Basic เข้าไปแทน ดังนั้นกำหนดให้ π คือเวคเตอร์ของตัวแปรดูอัล เราสามารถหา j ซึ่ง $j \in J$ ได้จาก $\min \bar{c}_j := c_j - \pi^t a_j$ เมื่อ $|J|$ มีขนาดใหญ่ จะกำหนดซับเซตของ J คือ $J' \subseteq J$ ของคอลัมน์ทั้งหมด และปัญหาของซับเซตดังกล่าวคือ RMP กำหนดให้ λ

คือคำตอบ และ π คือค่าดูอัลของคำตอบ RMP ปัจจุบัน เมื่อ a_j เป็นสมาชิกของเซต A และค่าสัมประสิทธิ์ของ c_j สามารถคำนวณได้จาก a_j โดยฟังก์ชัน c ซึ่งเป็นฟังก์ชันของ SP

$$\bar{c}^* := \min\left\{c(a) - \pi^t a \mid a \in A\right\} \tag{2.4}$$

จากสมการ (2.4) ถ้า $\bar{c}^* \geq 0$ นั้นหมายถึงไม่มี $\bar{c}_j, j \in J$ ใดมีค่าติดลบ จะทำให้คำตอบ หรือค่า $\bar{c}^*$ เป็นคำตอบที่ดีที่สุดของ RMP นั้นเป็นคำตอบที่ดีที่สุดของ MP ด้วย แต่ในกรณีที่ $\bar{c}^*$ มีค่าติดลบ จะทำการเพิ่มคอลัมน์ที่ได้จาก SP เข้าสู่ RMP และทำการคำนวณ RMP ซ้ำไปเรื่อย ๆ จน กระทั่ง $\bar{c}^* \geq 0$ ในขณะที่เราสามารถหาคำตอบจาก SP ได้คำตอบที่ดีที่สุดได้ จะทำให้คำตอบของ MP นั้นเป็นคำตอบที่ดีที่สุดเช่นกัน



ภาพที่ 2.4 ขั้นตอนการทำงานของวิธี CG

จากภาพที่ 2.4 อธิบายการทำงานของวิธี CGได้ดังนี้

1. กำหนดรูปแบบปัญหาอยู่ในสมการเชิงเส้น จากนั้นละเว้นเงื่อนไขว่าตัวแปรในการหาคำตอบมีการเป็นแบบไม่ต่อเนื่องให้ตัวแปรสามารถมีค่าต่อเนื่องได้ (Construct MP)

2. สร้างกลุ่มคอลัมน์เริ่มต้น หรือซับเซตของคอลัมน์ทั้งหมด (Construct RMP)

3. แก้ปัญหา RMP โดยใช้โปรแกรมเชิงเส้น จะได้ค่า Dual price (π^t) จากแต่ละเงื่อนไข

4. นำค่า Dual price จากปัญหาหลักมาช่วยในการหาคำตอบของ SP ทำให้ได้คอลัมน์ใหม่และคำนวณหาค่า Negative reduced cost ($\bar{c}^*$) ถ้ามีค่าน้อยกว่าศูนย์ จะถือว่าคอลัมน์ใหม่สามารถ

ปรับปรุงคำตอบให้กับปัญหาหลักได้ ดังนั้นจะเพิ่มคอลัมน์ใหม่เข้าไปยังซับเซตของคอลัมน์ และทำซ้ำขั้นตอนที่ 3 ถ้าค่า Negative reduced cost มีค่ามากกว่าหรือเท่ากับศูนย์หมายความว่าไม่มีคอลัมน์ที่จะสามารถปรับปรุงคำตอบให้กับปัญหาหลักได้ ดังนั้นคำตอบที่ได้จะเป็นคำตอบที่ดีที่สุด

Restricted Master problem และ Sub-problem

วิธี CG นั้นเป็นวิธีการหาคำตอบที่แบ่งปัญหาเป็นสองส่วนคือ RMP และ SP โดยฟังก์ชันวัตถุประสงค์ของ RMP นั้นคือฟังก์ชันวัตถุประสงค์ของปัญหาดั้งเดิม (MP) ส่วนฟังก์ชันวัตถุประสงค์ของ SP นั้นคือการหาคอลัมน์ที่มี Reduced cost ที่ดีที่สุดเพื่อเพิ่มเข้าสู่ซับเซตของคอลัมน์ใน RMP ซึ่งสามารถแยกเงื่อนไขของปัญหาดั้งเดิมออกเป็นสองส่วนกำหนดให้อยู่ใน RMP และ SP ก็ได้ อย่างไรก็ตามเมื่อรวมเงื่อนไขทั้งหมดใน RMP และ SP แล้วต้องมีเงื่อนไขของปัญหาดั้งเดิมทุกเงื่อนไข การแบ่งเงื่อนไขต่าง ๆ ว่าเงื่อนไขใดอยู่ในปัญหาส่วนใดนั้นเป็นเรื่องสำคัญเรื่องหนึ่ง กล่าวคือถ้ากำหนดเงื่อนไขใดเงื่อนไขหนึ่งใน RMP แล้วนั้น ใน SP เราสามารถละเว้น (Relax) เงื่อนไขดังกล่าวได้ และทั้งนี้เมื่อนำเงื่อนไขใดกำหนดใน SP เงื่อนไขดังกล่าวก็ไม่จำเป็นต้องกำหนดใน RMP แต่ก็สามารถกำหนดเงื่อนไขเดียวกันทั้ง RMP และ SP ก็ได้

การกำหนดเงื่อนไขนั้นดังที่กล่าวไปข้างต้นว่า เงื่อนไขที่กำหนดภายใน RMP แล้วใน SP เราสามารถจะละเว้นเงื่อนไขดังกล่าวได้ การละเว้นเงื่อนไขบางประการของ SP นั้นจะทำให้การหาคำตอบของคอลัมน์ที่ได้นั้นเป็นคำตอบที่ดีกว่าปกติ ซึ่งในบางครั้งนั้นเป็นคำตอบที่เป็นไปไม่ได้ในปัญหาดังเดิมเนื่องจากละเว้นเงื่อนไขบางส่วนแล้ว ทั้งนี้การคำนวณหาคำตอบของ SP นั้นจะนำค่า dual price ของแต่ละเงื่อนไขที่ละเว้นไปนั้นมาปรับเปลี่ยนค่าใช้จ่ายทำให้คอลัมน์ที่ได้นั้น เมื่อนำเข้าสู่ RMP จะทำให้คำตอบของ RMP นั้นได้คำตอบที่ดีขึ้น จากนั้นจะคำนวณ RMP โดยวิธี LP เพื่อได้ค่า Dual price มาปรับค่าใช้จ่ายของ SP ในการหาคอลัมน์ใหม่อีกครั้ง

การกำหนด SP นั้นดังที่กล่าวมาข้างต้นแล้วว่า เป็นการหาคอลัมน์ให้กับ RMP และสามารถละเว้นเงื่อนไขบางประการที่อยู่ใน RMP ได้ ดังนั้นจากปัญหาดั้งเดิมที่มีความซับซ้อน จะทำให้ SP นั้นมีความซับซ้อนน้อยลง ทำให้ในบางปัญหานั้นจากเดิมปัญหาดั้งเดิมไม่สามารถหาคำตอบที่ดีที่สุดได้ด้วยเหตุผลเรื่องเวลาในการคำนวณ เมื่อละเว้นเงื่อนไขบางประการแล้ว SP สามารถหาคำตอบที่ดีที่สุดได้ให้กับ RMP เมื่อคำนวณ RMP จนไม่สามารถหาคอลัมน์ที่ปรับปรุงคำตอบได้แล้ว ก็จะถือว่าหาคำตอบที่ดีที่สุดของปัญหาได้เช่นกัน ทั้งนี้ขึ้นอยู่การละเว้นเงื่อนไขในข้อใดของปัญหาดั้งเดิมด้วย เพราะมีบางปัญหาแม้จะละเว้นปัญหาบางประการแล้วความซับซ้อนของปัญหาจะน้อยลงเพียงเล็กน้อยเท่านั้น และยังไม่สามารถหาวิธีการแก้ปัญหาที่หาคำตอบที่ดีที่สุดใน SP ได้ สามารถนำวิธีฮิวริสติกส์ หรือเมต้าฮิวริสติกส์มาแก้ปัญหาได้ แต่ทั้งนี้จะไม่สามารถรับประกันคำตอบสุดท้ายของ RMP ได้ว่าเป็นคำตอบที่ดีที่สุด อย่างไรก็ตาม คำตอบที่ได้จากวิธี CG ดังกล่าวนั้นจะเป็นคำตอบที่ดีกว่า และมีความเสถียรภาพกว่าการแก้ปัญหาโดยวิธีฮิวริสติกส์ และเมต้าฮิวริสติกส์เท่านั้น

การประยุกต์ใช้วิธีคอลัมน์เจนเนอร์เรชั่นและวิธีโลคอลเสิร์จ

การแก้ปัญหาโดยใช้ CLS นั้นเป็นวิธีหนึ่งที่มีความยืดหยุ่นและสามารถหาคำตอบที่ดีได้ในระดับหนึ่ง แต่อย่างไรก็ตามคำตอบที่ได้นั้นในบางครั้งอาจจะไม่มีความเสถียรภาพเท่าที่ควร ดังนั้นเราจึงนำวิธี CG เข้ามาช่วยเพิ่มประสิทธิภาพ และความเสถียรภาพของคำตอบ โดยกำหนดรูปแบบปัญหาให้อยู่ในรูปของ RMP และนำวิธี CLS มาใช้แก้ SP ทั้งนี้เนื่องจากปัญหาการจัดเส้นทางและตารางเวลายานพาหนะนั้น แม้ละเว้นเงื่อนไขบางประการแล้ว ก็เป็นการยากที่จะหาคำตอบที่ดีที่สุดภายในเวลาที่จำกัด ดั้งนั้นเราจึงนำเสนอวิธี CLS ในการหาคำตอบของ SP เนื่องจากมีความยืดหยุ่นในการแก้ปัญหา สามารถปรับเปลี่ยนเงื่อนไขได้สะดวก รวดเร็ว และเมื่อนำ CLS มาแก้ SP นั้นก็สามารถนำค่า Dual price ที่ได้จาก RMP มาสมการเพื่อหาค่าความขัดแย้งในเงื่อนไขรองได้ทันทีโดยไม่จำเป็นต้องเปลี่ยนขั้นตอนการหาคำตอบ และเมื่อมีเงื่อนไขเพิ่มเติมของปัญหาสามารถนำเงื่อนไขดังกล่าวเพิ่มเข้าสู่ CLSได้ เนื่องจาก CLS สามารถหาคำตอบที่ถูกต้องตามเงื่อนไขของปัญหาดั้งเดิมได้ในการหาคอลัมน์เริ่มต้นนั้นสามารถนำ CLS มาหาได้ทันทีและมีประสิทธิภาพ เนื่องจากคอลัมน์เริ่มต้นที่ได้นั้นถูกต้องตามเงื่อนไขของ MP ด้วย

3. ผลการทดลอง

3.1 ปัญหาการจัดตารางยานพาหนะ

ปัญหาการจัดตารางยานพาหนะที่เรานำมาพิจารณาคือปัญหาการจัดตารางเวลารถประจำทาง ซึ่งเป็นปัญหาที่สำคัญของบริษัทผู้ให้บริการขนส่งสาธารณะ โดยทั่วไป การวางแผนการให้บริการรถประจำทางประกอบด้วย 4 ขั้นตอนได้แก่ 1) การกำหนดเส้นทางและความถี่ในการให้บริการ 2) การกำหนดตารางเวลาเดินรถ 3) การจัดตารางเวลาเดินรถประจำทาง และ 4) การจัดตารางเวลาการทำงานของพนักงาน เริ่มจากขั้นตอนการวางแผนเส้นทางเดินรถประจำทาง (Bus route หรือ Line) รวมทั้งการกำหนดความถี่ในการให้บริการ (Frequency, Headway) ต่อจากนั้น จะเป็นขั้นตอนการสร้างตารางเวลาเดินรถประจำทาง (Bus timetabling) โดยกำหนดจำนวนเที่ยวรถประจำทางทั้งหมดที่ต้องให้บริการต่อวัน หลังจากนั้น จะเป็นขั้นตอนการจัดตารางเดินรถประจำทาง (Bus scheduling) และจัดตารางเวลาการทำงานสำหรับพนักงาน (Crew scheduling) เพื่อให้บริการรถประจำทางตามตารางเวลาที่กำหนด

ในงานวิจัยนี้เสนอแนวคิดและวิธีแก้ปัญหาการจัดตารางเวลาเดินรถประจำทาง เพื่อให้ต้นทุนในการดำเนินงานน้อยที่สุด เราประเมินประสิทธิภาพของแบบจำลองและขั้นตอนวิธีทางคอมพิวเตอร์ที่พัฒนาขึ้นโดยใช้ข้อมูลจากองค์การขนส่งมวลชนกรุงเทพ

เราประเมินประสิทธิภาพของระบบจัดตารางเวลาเดินรถประจำทางที่พัฒนาขึ้นโดยใช้ข้อมูลจากองค์การขนส่งมวลชนกรุงเทพ (The Bangkok mass transit authority: BMTA), BMTA เป็นรัฐวิสาหกิจสังกัดกระทรวงคมนาคม ให้บริการรถประจำทางจำนวน 3,535 คัน 108 สายทั่วกรุงเทพมหานคร BMTA แบ่งเขตการให้บริการรถประจำทางออกเป็น 8 เขต (Zone) แต่ละเขตแบ่งออกเป็นกองการเดินรถประมาณ 3 กอง (Depot หรือ Division) แต่ละกองประกอบด้วยเส้นทางประมาณ 5-10 สาย ปัจจุบัน BMTA สร้างตารางเวลาเดินรถและจัดรถประจำทางเป็นรายเดือน โดยอาศัยนักวางแผนที่มีประสบการณ์ ซึ่งจำนวนเที่ยวให้บริการทั้งหมดในแต่ละวันจะถูกกำหนดในเชิง

นโยบายจากสำนักงานใหญ่อีกทีหนึ่ง ขั้นตอนการจัดรถประจำทางของ BMTA จะขึ้นอยู่กับตารางการทำงานของพนักงานประจำรถ (พนักงานขับรถ และพนักงานเก็บสตางค์) เป็นหลัก ซึ่งโดยทั่วไปพนักงานขับรถและพนักงานเก็บสตางค์จะทำงานคู่กันและใช้รถประจำทางคันเดิมในการให้บริการ

เนื่องจากในปัจจุบัน BMTA ยังไม่มีระบบคอมพิวเตอร์ช่วยในการจัดตารางเวลาเดินรถ รวมทั้งเวลาการเดินทางของรถประจำทางที่ไม่แน่นอนในกรุงเทพฯ ในแต่ละวัน ผู้จัดการสายจะจัดรถประจำทางตามหน้างาน โดยอาศัยแผนจำนวนเที่ยวของรถประจำทางในแต่ละชั่วโมง (Hourly timeslot) ที่กำหนดไว้ล่วงหน้าเป็นแนวทางในการปฏิบัติ นั่นคือ เมื่อรถประจำทางกลับเข้าถึงกองการเดินรถในทุก ๆ Round trip พนักงานจะทำการบันทึกเวลา จำนวนเที่ยวการทำงาน และรายได้จากการให้บริการ ต่อจากนั้น ผู้จัดการสายจะพิจารณาทรัพยากร (รถประจำทางและพนักงาน) ที่มีอยู่ในขณะนั้น และทำการกำหนดเวลาการให้บริการของรถประจำทางในสายนั้น ๆ ใหม่ตามความต้องการในการเดินทาง สภาพการจราจร และทรัพยากรที่มีอยู่

Potential cost savings with line changes and multi-depot

จากข้อมูลการทำงานในปัจจุบัน แม้ว่าเงื่อนไขในเชิงนโยบายบางประการอาจจะไม่อนุญาตให้รถประจำทาง วิ่งสลับสาย อย่างไรก็ตาม หลาย ๆ ครั้งในทางปฏิบัติ BMTA มีการวิ่งสลับสายอยู่บ้าง ซึ่งจะกระทำขึ้นที่หน้างานโดยทันที โดยการสลับสายนั้น ผู้จัดการสายจะพิจารณาจากช่วงเวลา และบาง Section ของ Line ในแต่ละวัน (ในบางสายช่วงเวลาที่เป็น Peak demand อาจไม่เท่ากัน โดยเฉพาะอย่างยิ่งในช่วงเย็นหลังเลิกงาน ผู้โดยสารอาจจะเลิกงานพร้อมกัน แต่อาจจะทยอยกันกลับบ้านในช่วงเวลาที่แตกต่างกัน ในทางตรงกันข้าม ช่วง Morning peak hour รถประจำทางบางสายอาจจะติดอยู่ในกระแสการจราจรทั้งหมด) ดังนั้น จึงมีโอกาสในการจัดสรร (Re-allocate) รถประจำทางจากสายอื่นมาใช้ สังเกตอีกด้วยว่า ในปัจจุบัน BMTA ยังไม่มีการใช้รถประจำทางร่วมกันระหว่างกองการเดินรถ นั่นอาจเป็นเพราะว่า BMTA แบ่งโครงสร้างการบริหารในระดับกองการเดินรถเป็นอิสระต่อกัน

เราทดสอบแบบจำลองที่พัฒนาขึ้นโดยกำหนดค่า Headway ของรถประจำทางแต่ละสายให้แปรผันตามช่วงเวลา และกำหนด Trip time (โดยใช้ค่าเฉลี่ยของเวลาในการเดินทางในอดีตในแต่ละช่วงเวลาของวันต่าง ๆ) ของรถประจำทางปรับอากาศจำนวน 9 สาย 125 คัน สังกัดเขตการเดินรถที่ 4 ซึ่งประกอบด้วย 3 กองการเดินรถ เพื่อแสดงให้เห็นถึงศักยภาพในการลดต้นทุนในการดำเนินงานเมื่อกองการเดินรถทั้ง 3 ใช้ทรัพยากรของรถประจำทางร่วมกัน เรากำหนดค่าพารามิเตอร์ที่ใช้ในแบบจำลองดังนี้ ค่าใช้จ่ายคงที่ของรถประจำทาง/ คัน/ วัน เท่ากับ 5,000 บาท (ซึ่งอาจจะประมาณจากค่าเช่า หรือ Asset cost ที่เกิดขึ้นจริง) ค่าใช้จ่ายในการสลับสายต่อครั้งเท่ากับ 100 บาท (× 2 คิดเปลี่ยนไปและเปลี่ยนกลับ) ค่าใช้จ่ายในการใช้รถประจำทางจากกองการเดินรถอื่นต่อครั้ง (โดยคิดแยกออกจากค่าใช้จ่ายที่เกิดจากระยะทางในการเดินทาง) เท่ากับ 500 บาท (× 2 คิดเรียกออกและเรียกเข้า) และค่าใช้จ่ายในการใช้งานรถประจำทางต่อกิโลเมตรเท่ากับ 30 บาท ผลการทดลองแสดงในตารางที่ 3.1

ตารางที่ 3.1 Multiple-depot with line change operations

จากตารางที่ 3.1 จะเห็นได้ว่า เมื่อใช้ระบบที่พัฒนาขึ้น (DSS) จัดตารางเดินรถประจำทางแบบสลับสาย (Line change) ทำให้ BMTA สามารถลดจำนวนรถประจำทางที่ใช้ทั้งหมดลงมากถึง 27 คัน และเมื่อมีการใช้รถประจำทางร่วมกันระหว่างกองการเดินรถ (Multi-depot) ด้วย สามารถลดจำนวนรถประจำทางลงไปได้อีก 17 คัน สังเกตด้วยว่า การใช้รถประจำทางระหว่างกองการเดินรถสามารถลดค่าใช้จ่ายในการดำเนินงานลงเพียงเล็กน้อยเมื่อเปรียบเทียบกับการวิ่งสลับสาย เนื่องจากต้องเสียค่าใช้จ่ายในการ Transfer รถระหว่างกอง ดังนั้น ในตารางที่ 3.1 ระบบ DSS สามารถช่วย BMTA ลดค่าใช้จ่ายในการดำเนินงานได้ 91,000 บาทต่อวัน (หรือ 2.73 ล้านบาทต่อเดือน)

เราตระหนักด้วยว่า จำนวนรถประจำทางที่ใช้งานทั้งหมดนั้นแปรผันโดยตรงกับความแตกต่างของการให้บริการรถประจำทางในแต่ละสาย (นั่นคือ ระยะเวลาของ Trip และ Headway ที่กำหนดในแต่ละช่วงเวลา) ยิ่งมีความแตกต่างกันมาก ก็จะทำให้มีโอกาสในการใช้ทรัพยากรร่วมกันได้มาก ซึ่งทำให้ BMTA สามารถใช้จำนวนรถประจำทางที่ลดลงได้ เราอาจจะตั้งข้อสังเกตด้วยว่า ผลที่ได้จากตารางที่ 3.1 อาจจะไม่เป็นจริงทั้งหมดในทางปฏิบัติ ในลักษณะการทำงานที่เป็นอยู่ในปัจจุบัน อย่างไรก็ตาม เมื่อมีสายรถประจำทางจำนวนมาก วิ่งสลับสาย และใช้รถร่วมกันระหว่างกอง ระบบ DSS จะยังคงสามารถช่วยนักวางแผนในการจัดตารางเวลาเดินรถได้โดยง่ายและมีประสิทธิภาพ

3.2 ปัญหาตารางทำงานของพนักงานเดินรถ

ปัญหาการจัดตารางเวลาการทำงานของพนักงานของระบบขนส่งมวลชน (Crew scheduling problem: CSP) เป็นปัญหาการตัดสินใจที่มีขนาดใหญ่และมีความซับซ้อนมากกว่าปัญหาการตัดสินใจในภาคอุตสาหกรรมขนส่งและโลจิสติกส์ทั่วๆ ไป เนื่องจากพนักงานแต่ละคนถูกพิจารณาเป็นทรัพยากร 1 หน่วย และมีเป็นจำนวนมาก ยกตัวอย่างเช่น เมื่อเปรียบเทียบกับปัญหาการจัดเส้นทางและตารางเวลาของยานพาหนะ ซึ่งทรัพยากรคือจำนวนยานพาหนะซึ่งอาจมีจำนวนไม่กี่สิบคันเท่านั้น นอกจากนี้ ปัญหา CSP ยังมีเงื่อนไขที่เกี่ยวกับความพึงพอใจของพนักงาน (Human factors) เข้ามาเกี่ยวข้องด้วย อาทิเช่น จำนวนชั่วโมงการทำงาน เวลาเริ่มงาน (กะการทำงาน) เวลาการหยุดพักระหว่างการทำงาน วันหยุด การจัดสรรค่าล่วงเวลา ทักษะและความเหมาะสมของงานที่

ทำ ฯลฯ ซึ่งเงื่อนไขในการทำงานต่าง ๆ เหล่านี้จะต้องพิจารณาอย่างละเอียดรอบคอบ ไม่เพียงต้องการจัดสรรให้พนักงานทำงานให้เกิดประสิทธิภาพมากที่สุด ด้วยต้นทุนในการดำเนินการน้อยที่สุด แต่เรายังต้องคำนึงถึงระดับความพึงพอใจของพนักงานและความเสมอภาคในการทำงานให้มากที่สุดด้วยเช่นกัน

เราทดสอบแบบจำลองและอัลกอริทึมที่พัฒนาขึ้นโดยประยุกต์ใช้กับปัญหา CSP ขององค์การขนส่งมวลชนกรุงเทพ (BMTA) เพื่อความสะดวกในการอธิบาย เราแบ่งเงื่อนไขรอง (หรือฟังก์ชันวัตถุประสงค์) ออกเป็น 3 กลุ่มคือ 1) Pre-specified 2) Crew preference และ 3) Schedule cost ดังแสดงในตารางที่ 3.2

ตารางที่ 3.2 Soft violation scheme

No.	Soft constraints	Vilolation
	Pre-specified	
A-1	Crew pair	0
A-2	#Break	0
A-3	Break day	0
A-4	Shift type	0
A-5	Sing-in/Sign-off time	0
	Crew preference	
B-1	Priority bus	5
B-2	Priority line (4 levels)	0,5,15,30
B-3	Duty spreadover	1
	Schedule cost	
C-1	#Crew	500
C-2	Duty operating cost (DOC)/min	1.05
C-3	Overtime (OT)/min	5

ตารางที่ 3.2 เงื่อนไขรองทุกตัวในกลุ่ม Pre-specified มีค่าความขัดแย้ง (Violation) เท่ากับ 0 หรือกล่าวอีกนัยหนึ่ง เงื่อนไขในกลุ่มนี้ได้แก่ การจับคู่ในการทำงานระหว่างพนักงานขับรถและพนักงานเก็บสตางค์ (Crew pair) จำนวนวันหยุดประจำเดือน (#Break) วันหยุดประจำสัปดาห์ (Break day) ประเภทของกะการทำงาน (Shift type) และเวลาเริ่มต้นและสิ้นสุดการทำงานในแต่ละวัน (Sign-in/ Sign-off time) ได้ถูกกำหนดค่าไว้ล่วงหน้าแล้วโดยนักวางแผน เพื่อให้เกิดความยืดหยุ่นในการปฏิบัติงาน และลดพื้นที่ในการค้นหาคำตอบของอัลกอริทึม เงื่อนไขในกลุ่มที่ 2 แสดงค่าความขัดแย้งที่เกิดจากความชอบในการใช้รถประจำทางที่กำหนด (Priority bus) เส้นทางที่พนักงานมีทักษะในการขับขี่ หรือเป็นสายประจำ (Priority line) โดยที่ค่าความขัดแย้งจาก Priority line จะถูกแบ่งออกเป็น 4 ระดับ นั่นคือ รถประจำทางสายแรกนั้นจะมีค่าความขัดแย้งเป็นศูนย์ และสายต่อมามีค่าความขัดแย้งเป็น 5 และ 15 ตามลำดับ และนอกเหนือ 3 สายแรกแล้วทั้งหมดจะมีค่าความขัดแย้งเท่ากันทั้งหมดคือ 30 เงื่อนไขรองในกลุ่มที่ 2 ยังแสดงค่าความขัดแย้งที่เกิดจากความเท่าเทียมกันของจำนวนเที่ยวการทำงาน หรือจำนวนชั่วโมงการทำงานของพนักงานในแต่ละวัน ซึ่งเงื่อนไขรองแต่ละตัวในกลุ่มที่ 2 นี้อาจจะถูกกำหนดระดับความสำคัญ (ค่าความขัดแย้ง) ที่แตกต่างกันตามนโยบายของบริษัท

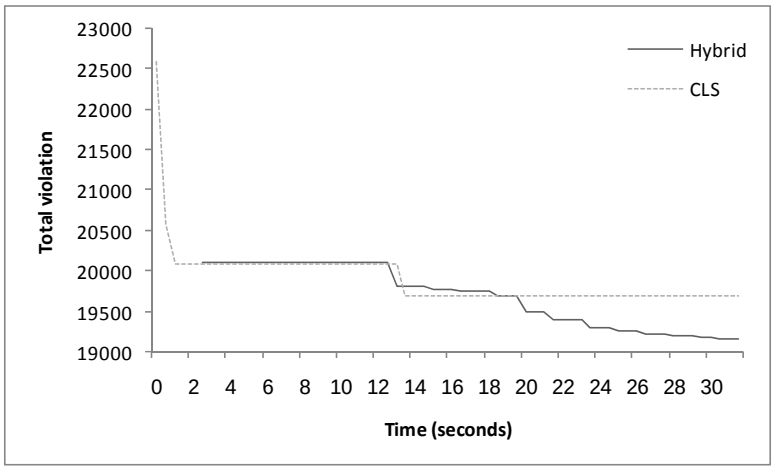
เงื่อนไขรองในกลุ่มสุดท้าย (Schedule cost) แสดงค่าความขัดแย้งที่เกิดจากค่าใช้จ่ายในการทำงานของพนักงาน และสามารถกำหนดค่าได้อย่างตรงไปตรงมา ได้แก่ ค่าใช้จ่ายที่เกิดจากการจ้าง

พนักงานต่อวัน (#Crew) ค่าใช้จ่ายที่เกิดจากการทำงานของพนักงานต่อนาที (Duty operation cost: DOC) และค่าใช้จ่ายที่เกิดจากค่าล่วงเวลาต่อนาที (Overtime)

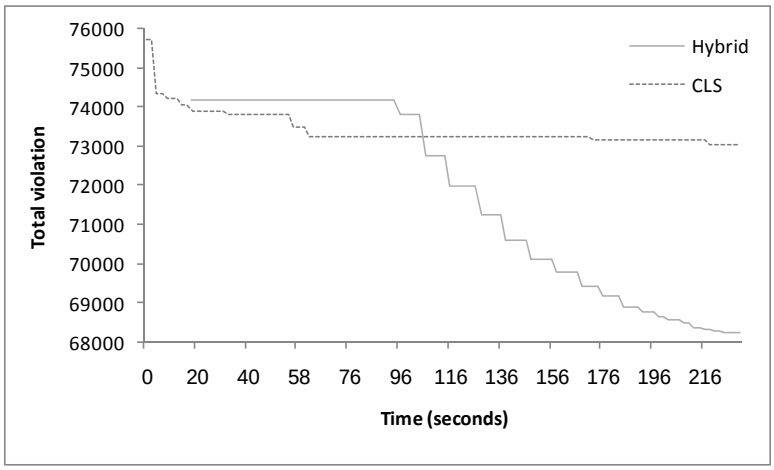
ตารางที่ 3.3 ผลการทดลองการจัดตารางทำงานของพนักงาน

Problem	Trip	CLS				Hybrid			
		#Crew	Avg.Cost	Best Cost	Time to best	#Crew	Avg.Cost	Best Cost	Time
P1	75	25.4	19,834	19,696	14	24	19,182	19,155	31
P2	75	23	16,041	15,947	11	22	15,520	15,520	29
P3	75	24	17,023	16,982	15	23	16,471	16,442	36
P4	75	26	21,005	20,998	10	24	19,383	19,322	25
P5	75	19	13,236	13,221	9	18	12,950	12,950	32
P6	72	25	19,885	19,885	10	24	19,416	19,410	26
P7	72	24	19,530	19,422	8	24	19,285	19,285	32
P8	72	25.5	21,492	20,947	18	24	19,756	19,756	41
P9	72	24	19,329	19,211	15	24	19,014	18,945	28
P10	72	26	22,032	22,004	20	25	21,498	21,329	37

ตารางที่ 3.3 แสดงผลการทดลองการจัดตารางทำงานของพนักงานของรถประจำทางหนึ่งสาย จะเห็นได้ว่าวิธีแบบพันธุ์ผสมนั้นสามารถหาคำตอบที่ใกล้เคียงคำตอบที่ดีที่สุดมากกว่าวิธี CLS โดยสามารถลดจำนวนพนักงานได้ในบางปัญหา และในปัญหาที่ใช้พนักงานจำนวนเท่ากันก็สามารถเสียค่าใช้จ่ายน้อยกว่า โดยค่าใช้จ่ายดังกล่าว คือค่าความขัดแย้งทั้งหมดนอกจากค่าใช้จ่ายจากเวลาทำงาน และค่าใช้จ่ายในการทำงานล่วงเวลาแล้วรวมถึงการความขัดแย้งจากพนักงาน ยานพาหนะ และสายของรถประจำทางด้วย



ก.



ข.

ภาพที่ 3.2 การเปรียบเทียบการพัฒนาคำตอบในแต่ละรอบการทำงาน

จากภาพที่ 3.2 ก. แสดงการพัฒนาคำตอบในแต่ละรอบการทำงานของปัญหา P1 และ ข. แสดงการพัฒนาคำตอบในแต่ละรอบการทำงานของปัญหา P12 จะเห็นว่าวิธี CLS นั้นสามารถพัฒนาคำตอบที่ดีขึ้นได้อย่างรวดเร็วกว่าวิธีพันธุ์ผสม เมื่อถึงระยะเวลาหนึ่งวิธี CLS นั้นจะไม่สามารถพัฒนาคำตอบที่ดีขึ้นได้ และวิธีพันธุ์ผสมนั้นจะเริ่มพัฒนาคำตอบที่ช้ากว่าแต่คำตอบที่ได้นั้นเป็นคำตอบที่ดีกว่าวิธี CLS

ตารางที่ 3.4 ผลการทดลองการจัดตารางทำงานพนักงานรวมหลายสาย

ตารางที่ 3.4 จะเห็นได้ว่าการจัดตารางทำงานพนักงานโดยการรวมรถประจำทางหลายสายนั้นจะสามารถลดจำนวนพนักงานในการทำงานได้ในทุกปัญหา โดยค่าความขัดแย้งของในความถนัดของสายรถประจำทางของพนักงานแต่ละรายนั้นจะมีค่าเพิ่มขึ้น แต่ค่าความขัดแย้งของค่าความแตกต่างกันระหว่างระยะเวลาทำงานของพนักงานจะลดลง และค่าใช้จ่ายของจำนวนพนักงานนั้นจะมีค่าลดลง

3.3 ปัญหาการรับ/ ส่งสินค้าสำหรับยานพาหนะ

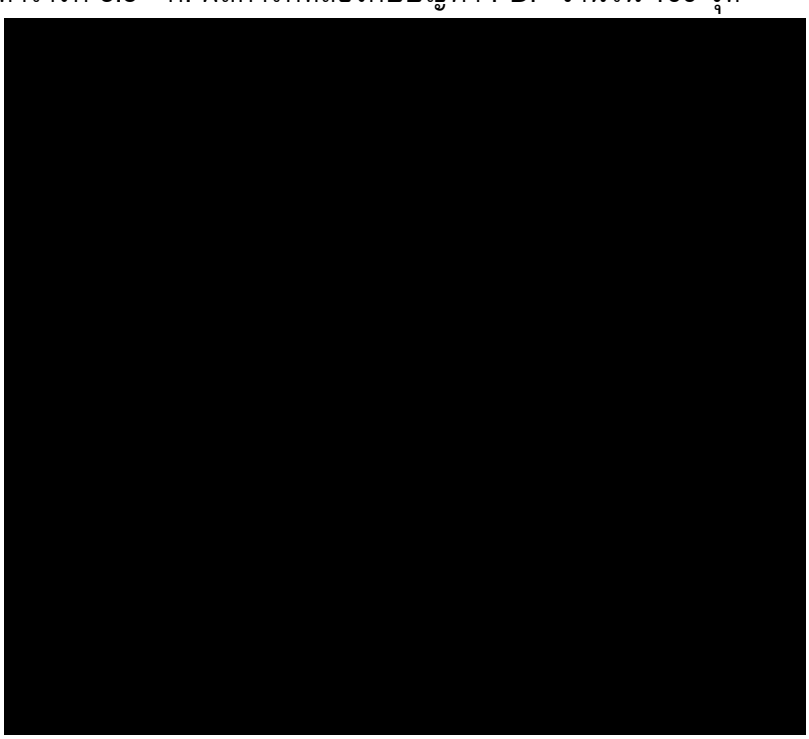
ปัญหาการจัดเส้นทางยานพาหนะโดยมีจุดรับ/ ส่งหลายจุด (Pickup and delivery problem: PDP) นั้น มีพื้นฐานของปัญหามาจากปัญหาการจัดเส้นทางของยานพาหนะ (Vehicle routing problem: VRP) ซึ่งจะมีจุดเริ่มต้นงานที่จุด ๆ หนึ่ง และออกเดินทางจากจุดเริ่มต้นนั้นไปให้บริการยังจุดต่าง ๆ โดยลักษณะของการบริการนั้นจะเป็นเพียงการรับ หรือการส่งสินค้าอย่างใดอย่างหนึ่งเท่านั้น และเมื่อสิ้นสุดงานจะกลับมายังจุดเริ่มต้นอีกครั้ง โดยปัญหาการจัดเส้นทางของยานพาหนะโดยมีจุดรับ/ ส่งหลายจุดนั้นจะมีความซับซ้อนของปัญหาและเงื่อนไขเพิ่มมากขึ้นโดยจะสามารถรับและส่งสินค้าได้ในระหว่างทาง

เพื่อประเมินประสิทธิภาพของอัลกอริทึมที่พัฒนาขึ้น เราคำนวณหาคำตอบของปัญหา Benchmark (Li & Lim, 2003) ที่อยู่ในคลาส LR และ LRC ซึ่งประกอบด้วยจุดรับ/ ส่งสินค้า (PD points) จำนวน 100 จุด กระจายตัวแบบ Randomly distributed และ Clustered and randomly distributed ตามลำดับ นอกจากนี้ เราทดสอบกับปัญหา LRC ของ Ropke and Pisinger (2006) ซึ่งมีจุดรับ/ ส่งจำนวน 200 จุด กระจายตัวแบบ Clustered and randomly distributed

เรากำหนด Violation scheme สำหรับวิธี CLS คือ p_l , p_v = 10 (สมการ 5.51 และ 5.52) และ $Iter_i$ = 20,000 ตารางที่ 3.5 แสดงคำตอบที่ได้ เปรียบเทียบกับ Best known, veh คือ จำนวนยานพาหนะ, dist คือระยะทางในการขนส่ง และ time (s) คือเวลาที่ใช้ในการประมวลผลในหน่วยวินาที

ตารางที่ 3.5 จะเห็นได้ว่าอัลกอริทึมที่พัฒนาขึ้นสามารถหาคำตอบที่มีคุณภาพทัดเทียมกับ Best known ภายในเวลาที่รวดเร็ว โดยเฉพาะปัญหา PDP 100 จุด เราได้จำนวนยานพาหนะ (veh) เท่ากับ Best known ทุกปัญหา สังเกตด้วยว่าปัญหา LR109, LRC102, LRC106, และ LRC108 อัลกอริทึมที่พัฒนาขึ้นสามารถหาคำตอบได้ดีกว่าอัลกอริทึมของ Li and Lim (2003) สำหรับปัญหา PDP 200 จุด อัลกอริทึมหาคำตอบที่มีคุณภาพดีน้อยกว่า Best known เพียงเล็กน้อย อย่างไรก็ตามคำตอบที่ได้อาจจะถูกชดเชยกับความความยืดหยุ่นของอัลกอริทึมที่ใช้ และความเร็วที่ใช้ในการหาคำตอบ ในภาคผนวกแสดงตัวอย่างคำตอบที่ได้จากอัลกอริทึมในการแก้ปัญหา LR101, LRC101 และ LR1_2_1

ตารางที่ 3.5 ก. ผลการทดลองกับปัญหา PDP จำนวน 100 จุด



ตารางที่ 3.5 ข. ผลการทดลองกับปัญหา PDP จำนวน 200 จุด

BVH (Bent & Hentenryck, 2006); Li and Lim (2003); SAM (SINTEF, 2003); RP (Ropke & Pisinger, 2006); TS (TetraSoft A/S, 2003)

4. สรุปผลงานวิจัย

จากงานวิจัย เราได้พัฒนาและประยุกต์ใช้อัลกอริทึมในการแก้ปัญหาการตัดสินใจในภาคอุตสาหกรรมโลจิสติกส์ที่มีขนาดใหญ่และซับซ้อน โดยขั้นต้น เราประยุกต์ใช้วิธี CLS ในการแก้ปัญหาการจัดเวลาการทำงานของยานพาหนะและพนักงาน ผลการศึกษาแสดงให้เห็นว่าอัลกอริทึมที่พัฒนาขึ้นมีประสิทธิภาพ ซึ่งสามารถหาคำตอบที่ดีภายในเวลาอันรวดเร็ว จากนั้น เราพัฒนาวิธีพันธ์ผสมระหว่างวิธี CG และ CLS ทำให้ผลของคำตอบมีคุณภาพที่ดีกว่าเมื่อเปรียบเทียบกับการประยุกต์ใช้วิธี CLS เพียงอย่างเดียว หลังจากนั้น เราประยุกต์ใช้อัลกอริทึมที่พัฒนาขึ้นกับปัญหาการจัดเส้นทางขนส่งสินค้าแบบมีจุดรับส่งหลายจุด โดยสามารถหาคำตอบได้อย่างมีประสิทธิภาพ ทั้งนี้เราทดสอบโดยการนำปัญหาอ้างอิง (Benchmark) มาเปรียบเทียบ ผลการศึกษาชี้ให้เห็นว่า วิธีพันธ์ผสมที่พัฒนาขึ้นสามารถหาคำตอบดีเทียบเท่าคำตอบ Best known ในปัจจุบัน และดีกว่าวิธีอื่นๆ อีกหลายวิธี ทั้งนี้วิธี CLS นั้นมีโครงสร้างที่ง่ายและยืดหยุ่น โดยเราสามารถประยุกต์ใช้ในการแก้ปัญหาหลายปัญหา โดยมีขั้นตอนการทำงานคล้ายคลึงกัน แม้ว่าการประยุกต์ใช้วิธี CG กับวิธี CLS นั้นสามารถหาคำตอบที่ดีกว่าวิธี CLS อย่างเดียว อย่างไรก็ตาม อัลกอริทึมจะใช้ในการประมวลผลมากกว่า ดังนั้น เราอาจสรุปได้ว่า ปัญหาที่ต้องการคุณภาพของคำตอบที่ดีและสามารถรอเวลาในการคำนวณได้นั้น เราสามารถใช้วิธี CG ในการแก้ปัญหาได้อย่างมีประสิทธิภาพ และปัญหาที่ต้องการคำตอบที่ดีในระยะเวลาอันรวดเร็วเราสามารถใช้วิธี CLS ในการแก้ปัญหาได้

บรรณานุกรม

Beasley, J.E. & Cao, B. (1996). A tree search algorithm for the crew scheduling problem. *European Journal of Operational Research, 94*, 517 – 526.

Bent, R., & Hentenryck, P.V. (2006). A two-stage hybrid algorithm for pickup and delivery vehicle routing problem with time Windows. *Computers & Operations Research, 33*, 875 - 893.

Bertossi, A.A., Carraresi, P., & Gallo, G. (1987). On some matching problems arising in vehicle scheduling models. *Networks, 17*, 271 – 281.

Freling, R., Huisman, D., & Wagelmans, A.P.M. (2003). Models and algorithms for integrations of vehicle and crew scheduling. *Journal of Scheduling, 6*, 59 – 81.

Ftulis, S.G., Giordano, M., Pluss, J.J., &Vota, R.J. (1998). Rule-based constraints programming: application to crew assignment. *Expert systems with application, 15*, 77-85.

Hadjar, A. Marcotte, O., & Soumis, F. (2006). A branch and cut algorithm for the multiple depot vehicle scheduling problem. *Operations Research, 54*, 130 – 149.

Haghani, A., Banihashemi, M., & Chiang, M.H. (2003). A comparative analysis of bus transit vehicle scheduling models. *Transportation Research Part B, 37*, 301 – 322.

Huisman, D., Freling, R., & Wagelmans, A.M.P. (2005). Multiple-depot integrated vehicle and crew scheduling. *Transportation Science, 39*(4), 491 – 502.

Indra-Payoong, N., Kwan, R.S K., & Proll, L. (2005). Rail container service planning: a constraint-based approach. In Kendall, G., Petrovic, S., Burke, E., & Gendreau, M. (editors) *Multidisciplinary Scheduling: Theory and Applications*. (pp. 343-365). Springer-Verlag.

Kliewer, N., Mellouli, T., & Suhl, L. (2006). A time-space network based exact optimization model for multi-depot bus scheduling. *European Journal of Operational Research, 175*(3), 1616 – 1627.

Li, H., & Lim, A. (2003). A Metaheuristic for the pickup and delivery with time windows. *International Journal on Artificial Intelligent Tools, 12*(2), 173 - 186.

Lobel, A. (1997). *Optimal vehicle scheduling in public transit*. PhD thesis, Technische University, Berlin.

_____. (1998). Vehicle scheduling in public transit and Lagrangian pricing. *Management Science, 4*, 1637 – 1649.

Mesquita, M., & Paias, A. (2007). Set partitioning/covering-based approaches for the integrated vehicle and crew scheduling. *Computers & Operations Research (2007)*.

Nanry, W.P., & Barnes, J.W. (2000). Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B, 34*, 107 - 121.

Pepin, A.S., Desaulniers, G., Hertz, A., & Huisman, D. (2006). *Comparison of heuristic approaches for the multiple depot vehicle scheduling problem*. Report EI2006-34, Econometric Institute, Erasmus University Rotterdam.

Rodrigues, M.M., Souza, C.C., & Moura, A.V. (2006). Vehicle and crew scheduling for urban bus lines. *European Journal of Operational Research, 170*, 844 – 862.

Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science, 40*(4), 455 – 472.

Santos, A.G., & Mateus, G.R. (2007). Hybrid approach to solve a crew scheduling problem: an exact column generation algorithm improved by metaheuristics. In *Proceedings of 7th International Conference on Hybrid Intelligent Systems*.

Shibghatullah, A.S., Eldabi, T., & Kuljis, J. (2006). A proposed multiagent model for bus crew scheduling. In *Proceedings of the 2006 Winter Simulation Conference*.

SINTEF Applied Mathematics. (2003). Department of Optimization. Technical Report, Unpublished results.

TetraSoft A/S. (2003). MapBooking algorithm for pickup and delivery solutions with time windows and capacity restraints. Unpublished results.

Valouxis, C., & Housos, E. (2002). Combined bus and driver scheduling. *Computers & Operations Research, 29*, 243 – 259.

Vanitchakornpong, K., Indra-Payoong, N., Sumalee, A., & Raothanachonkun, P. (2008). Constrained local search method for bus fleet scheduling problem with multi-depot with line change. *Evoworkshop 2008, M. Giacobini et al. (Eds.) LNCS 4974*, 669 – 678.

Xu, H., Chen, Z.L., Rajagopal, S., & Arunapuram, S. (2003). Solving a practical pickup and delivery problem. *Transportation Science, 37*(3), 347-364.

ผลลัพธ์โครงการวิจัย (Output)

1. ผลงานตีพิมพ์ในวารสารวิชาการนานาชาติ

Indra-Payoong, N., Sumalee, A., Vanitchakornpong, K., and Sseto, W.Y. “A hybrid column generation and local search algorithm for pickup and delivery problems,” Submitted to

Transportation Research Record (SCI Journal: Impact 0.145)
Number: 09-1490
Submission Date: Aug 1 2008 1:02PM

Vanitchakornpong, K., **Indra-Payoong,** N., Sumalee, A., and Raothanachonkun, P. (2008) “Constrained local search method for bus fleet scheduling problem with multi-depot with line change,” *Lecture Notes in Computer Science (LNCS)*, 4974, pp. 669 – 678.

2. การนำผลงานวิจัยไปใช้ประโยชน์

-

3. ผลงานตีพิมพ์ในวารสารวิชาการในประเทศ การเสนอผลงานในที่ประชุมวิชาการ

เกรียงศักดิ์ วณิชชากรพงศ์ กฤษณะ ชินสาร **และณกร อินทร์พยุง** (2551) “การประยุกต์ใช้วิธีคอลัมน์เจนเนอร์เรชันสำหรับปัญหาการตารางทำงานรถประจำทาง,” การประชุมวิชาการด้านการวิจัยดำเนินงานแห่งชาติ ครั้งที่ 5, 24 – 25 กรกฎาคม, กรุงเทพฯ หน้า 55 – 62.

Vanitchakornpong, K. **Indra-Payoong, N.**, Sumalee, A., and Raothanachonkun, P. (2008) “Solving Bus Crew Scheduling Problem: A constraint-based Approach,” In: Proceedings of the 13th International Symposium on Logistics, 6-8 July, Bangkok, pp. 340 – 348.

Indra-Payoong, N., Sumalee, A., Vanitchakornpong, K., and Chinnasarn, K. "A column generation based local search for pickup and delivery problem," ECTI Transactions on Computer and Information Technology, (to appear).

Vanitchakornpong, K., **Indra-Payoong,** N., Sumalee, A., and Raothanachonkun, P. (2008) “Constrained local search method for bus fleet scheduling problem with multi-depot with line change,” Lecture Notes in Computer Science (LNCS), 4974, pp. 669 – 678.

Vanichakornpong, K., Chinnasarn, K., and **Indra-Payoong**, N. (2007) “A column generation based local search for pickup and delivery problem,” In: Proceedings o f the 11th National Computer Science and Engineering Conference, 19-21 November 2007, Bangkok, pp. 90 – 99. (Best Paper Awards).

ณกร อินทร์พยุง, เกรียงศักดิ์ วณิชชากรพงศ์, ขวัญชัย ศรีงิ้วราย (2550) “ระบบการจัดตารางเวลาและเส้นทางการขนส่งสินค้าของรถบรรทุกโดยใช้ข้อมูลการเดินทางแบบเรียลไทม์” เอกสารการประชุมวิชาการ การขนส่งแห่งชาติครั้งที่ 4, 23 พฤศจิกายน 2550 เชียงใหม่ (NTC4-64)

ภาคผนวก

บทความสำหรับการเผยแพร่

Indra-Payoong, N., Sumalee, A., Vanitchakornpong, P., Sseto, W.Y. “A hybrid column generation and local search algorithm for pickup and delivery problems,” Submitted to

Transportation Research Record (SCI Journal: Impact 0.145)
Number: 09-1490
Submission Date: Aug 1 2008 1:02PM

Vanitchakornpong, K., **Indra-Payoong,** N., Sumalee, A., and Raothanachonkun, P. (2008) “Constrained local search method for bus fleet scheduling problem with multi-depot with line change,” *Lecture Notes in Computer Science (LNCS)*, 4974, pp. 669 – 678.

A HYBRID COLUMN GENERATION AND LOCAL SEARCH ALGORITHM FOR PICKUP AND DELIVERY PROBLEMS

Dr. N. Indra-Payoong
Assistant Professor
Faculty of Logistics
Burapha University
Chonburi, Thailand
Email: nakorn.ii@gmail.com

Dr. A. Sumalee (corresponding author)
Assistant Professor
Department of Civil and Structural Engineering
The Hong Kong Polytechnic University
Hong Kong SAR, China
Email: cesumal@polyu.edu.hk

Mr. K. Vanitchakornpong
Research Associate
Faculty of Logistics
Burapha University
Chonburi, Thailand
Email: kriangsv@buu.ac.th

Dr. W.Y. Sseto
Assistant Professor
Department of Civil Engineering
National University of Singapore
1 Engineering Drive 2, E1A 07-03
Singapore 117576
E-mail: cveswy@nus.edu.sg

Word Count

No. of words:		5,449
No. of tables:	250x3 =	750
No. of figures:	250x3 =	750
Total:		6,949

Submission Date: August 1, 2008

Paper Submitted for Presentation and Publication at the
Transportation Research Board's 88th Annual Meeting, Washington, D.C., 2009

Abstract: The paper proposes a heuristic algorithm for solving pickup and delivery problems (PDP). The PDP involves an assignment of pickup and delivery jobs to different fleets in optimal sequences to minimize the total logistic costs whilst satisfying several practical constraints (e.g., time-window constraint). An efficient pickup and delivery plan can reduce empty hauls and transportation cost of logistic operators significantly. The PDP in the real world is a large NP-hard problem. The paper handles this by combining column generation (CG) technique with constrained local search method (CLS). The paper applies the proposed algorithm to several well-known benchmark tests. The test results show satisfactory performance of the proposed algorithm in all cases compared to the benchmarks. The paper also applies the proposed algorithm to an actual PDP of a vehicle-carrier company in Thailand. The test result shows potential significant saving of the logistic cost through reductions of empty hauls.

Keywords: Pickup and delivery problem, column generation method, constrained local search, logistics efficiency, freight scheduling and planning

INTRODUCTION

A pickup and delivery problem (PDP) underpins the efficiency and cost of logistics operations. The PDP is an instance of the vehicle routing problem but involves more complex structures and constraints. In the PDP, the customers put requests to logistics operators to pickup and delivery commodities from and to certain locations potentially within certain time windows. There may be several requests involving multiple pickup and delivery points in the network. The main aim of the PDP is to assign these jobs to different vehicles in optimal sequences so as to minimize the total transportation costs whilst satisfying several constraints (e.g., time-window constraint or job sequencing). The problem has drawn a significant attention from researchers in the past few decades (*1*).

The PDP is an NP-Hard problem with complex constraints and large problem size. Often an optimal solution may not be found within a limited time. The PDP solution algorithms proposed thus far are based on either exact method, heuristics method, or meta-heuristics method (*2, 3*). Recently, due to the complexity of the practical constraints of the PDP and the need to find a good solution within a limited time different heuristics have been employed either as the main search algorithm or as a part of the hybrid approach (e.g., combined with an exact method or other heuristics) (*2*).

Nanry and Barnes (*4*) proposed the Relative Tabu Search (TS) for solving the PDP. The algorithm was tested with the modified version of the Solomon's benchmark problem. Li and Lim (*5*) proposed an approach which combines the TS and Simulated Annealing (SA) methods. Their approach uses the SA to restart the search in a new feasible partition after the TS cannot improve the solution anymore. This helps the algorithm to escape from local optima. Bent and Hentenryck (*1*) proposed the hybrid algorithm with two stages in which the SA is used to reduce the number of feasible routes or vehicles in the first stage. Then, in the second stage the large neighbourhood search (LNS) method is applied to minimize the total travelled distances of vehicles. The algorithm was reported to perform well with the test problem with around 600 pickup and delivery points. Ropke and Pisinger (*6*) also proposed the adaptive LNS approach which is similar to the method proposed in (*1*).

Xu et al. (*7*) tackled the problem size issue of the PDP by using column generation (CG) technique to set up a relaxed master problem (linear program) which then iteratively generates a number of sub-problems. Each sub-problem (also an NP-hard problem) corresponds to each vehicle and includes all complex PDP constraints (e.g., time window constraint). Thus, they proposed two fast heuristics, called MERGE and TWO-PHASE to solve the sub-problems and generate a new column for the relaxed master problem. The algorithm developed in this paper follows the framework of Xu et al. (*7*) which integrates the strengths of the CG and heuristics. However, our algorithm utilizes constrained local search (CLS) approach (*8*) to solve the sub-problem. The main impetus for using the CLS is to increase flexibility of the algorithm in handling complex constraints of the PDP. The framework of the CLS can be easily adjusted to handle new types of such constraints.

The rest of the paper is structured into four sections. The next section formulates the PDP. Then, the third section explains the proposed hybrid solution algorithm which combines the CG and CLS. The proposed algorithm is then tested with different well-known PDP benchmark cases in the fourth section. The algorithm is also applied to a real-world case of a car-carrier company in Thailand. The final section concludes the paper and discusses future research issues.

PROBLEM FORMULATION

Consider a directed graph G = (A, N) where A and N are sets of links and nodes respectively. Each link is defined by its start and end nodes, i.e., (i, j) where $i, j \in N$. Each node can be a pickup point, a delivery point, or a vehicle depot. Let $L \subseteq N$, $U \subseteq N$, and $D \subseteq N$ denote the sets of pickup points, delivery points, and vehicle depots respectively. Each link, $(i, j) \in A$, is associated with c_{ijk} and t_{ijk} which are the transportation cost and time for vehicle $k \in V$ (V is the set of vehicles) to traverse from nodes i to j respectively. Each vehicle $k \in V$ has the capacity of Q_k. q_s denotes the amount of commodity with order $s = 1,\ldots,S$ which will be picked up from node $o(s) \in N$ and delivered to node $d(s) \in N$. For each node i, let y_{ik}, a_{ik}, and s_i denote the amount of commodity carried by vehicle k to node i, arrival time of vehicle k to node i, and service time required at node i respectively. The decision variable for the PDP includes the binary variables $x_{ijk} \in \{0,1\}$ in which $x_{ijk} = 1$ if vehicle k traverses through link (i, j), and $x_{ijk} = 0$ otherwise. Also, the decision variable $e_{sk} \in \{0,1\}$ represents the dummy variable in which $e_{sk} = 1$, if the order s is assigned to vehicle k, and $e_{sk} = 0$ otherwise. The PDP can thus be defined mathematically as follows:

$$\min_{x_{ijk}, e_{sk}} \sum_{k \in V} \sum_{i \in N} \sum_{j \in N} c_{ijk} x_{ijk}$$

$s.t.$

$$\begin{aligned}
&\sum_{j \in N} x_{jik} - \sum_{\tilde{j} \in N} x_{\tilde{i}jk} = 0 && \forall i \in L \cup U; \forall k \in V \\
&\sum_{k \in V} \sum_{j \in N} x_{ijk} = 1 && \forall i \in L \\
&\sum_{j \in N} x_{o(s),j,k} - \sum_{j \in N} x_{j,d(s),k} = 0 && \forall s \in S; \forall k \in V \\
&\left(y_{jk} - y_{ik} - \sum_{\forall s | o(s) = i} e_{sk} q_s + \sum_{\forall s' | d(s') = i} e_{sk} q_{s'} \right) x_{ijk} = 0 && \forall i, j \in N; \forall k \in V \\
&Q_k - y_{ik} \geq 0 && \forall i \in N; \forall k \in V \\
&\left(a_{jk} - a_{ik} + s_i + t_{ij} \right) x_{ijk} = 0 && \forall i, j \in N; \forall k \in V \\
&l_{o(s)} - a_{o(s)} \geq 0; \tilde{l}_{d(s)} - a_{d(s)} \geq 0 && \forall s \in S \\
&x_{ijk}, e_{sk} \in \{0,1\} && \forall i, j \in N; \forall k \in V; \forall s \in S.
\end{aligned} \tag{1}$$

The objective function of the PDP is the total transportation cost. The first and fourth constraints are related to the flow conservation of the vehicles and commodities at all nodes. The second constraint ensures the solution to assign each order to only one vehicle. Each vehicle, after pick up the order, is required to deliver the order on the same itinerary which is represented by the third constraint. The fifth constraint imposes the capacity constraint on each vehicle. The sixth constraint defines the relationship between the arrival times of each vehicle to a sequence of nodes on a vehicle's route. The seventh constraint represents the time-window constraints for the arrival times of vehicles to different nodes in which $l_{o(s)}$ and $\tilde{l}_{d(s)}$ denotes the latest times to pick

up and deliver job s at nodes $o(s)$ and $d(s)$ respectively. The last constraint defines the decision variables to be binary variables. (1) represents the standard PDP formulation. The structure of the PDP as formulated in (1) involves a large number of decision variables and constraints. To remedy this, the method of column generation (CG) (*3*) will be adopted. To apply the CG, the PDP is reformulated on the basis of vehicle routes.

Let $p \in P_k$ be a feasible service route of vehicle k in which P_k is a set of feasible service routes of vehicle k. Any feasible service route included in P_k should satisfy all practical constraints as defined in (1). Each service route is already associated with a number of job, i.e., the value of e_{spk} is already defined in which if order s is assigned to route p of vehicle k, then $e_{spk} = 1$, and $e_{spk} = 0$ otherwise. The sequence of these jobs which will be picked up and delivered by vehicle k is also already determined by the sub-problem which will be described later. From the sequence of the assigned jobs, the arrival time of vehicle k to node i on route p (a_{ipk}), can be calculated. Let $\lambda_{pk} \in \{0,1\}$ denotes a binary decision variable in which $\lambda_{pk} = 1$ if the route p of vehicle k is selected, and $\lambda_{pk} = 0$ otherwise.

The PDP now involves the selection of the best route from the set of P_k for each vehicle such that all jobs are assigned to one and only one vehicle (route), i.e., $\sum_{k \in K} \sum_{p \in P_k} e_{spk} \lambda_{pk} = 1$. Note that under this formulation all other constraints are already satisfied since P_k contains only feasible service routes. Note that at most one route can be selected for each vehicle, i.e., $\sum_{p \in P_k} \lambda_{pk} \le 1$, in which vehicle k may not be used at all at the optimal solution of the PDP. The time-window constraint is also included as $\sum_{k \in K} \sum_{p \in P_k} a_{ipk} \lambda_{pk} \le l_i$ where l_i is the required arrival time at node i for pickup or delivery tasks. Thus, the PDP can be reformulated as:

$$
\begin{aligned}
&\min_{\lambda_{pk}} \sum_{k \in K} \sum_{p \in P_k} \overline{c}_{pk} \lambda_{pk} \\
&s.t. \\
&\quad \sum_{p \in P_k} \lambda_{pk} \le 1 \qquad \forall k \in K \\
&\quad \sum_{k \in K} \sum_{p \in P_k} e_{spk} \lambda_{pk} = 1 \qquad \forall s \in S \\
&\quad \sum_{k \in K} \sum_{p \in P_k} a_{ipk} \lambda_{pk} \le l_i \qquad \forall i \in N \\
&\quad \lambda_{pk} \in \{0,1\} \qquad \forall k \in K; \forall p \in P_k .
\end{aligned}
\tag{2}
$$

Note that $\overline{c}_{pk}$ denotes the transportation cost for the service route p of vehicle k in which $\overline{c}_{pk} = \sum_{i \in N} \sum_{j \in N} c_{ij} \delta_{ij,kp}$ where $\delta_{ij,kp} = 1$ if service route p of vehicle k uses link (i, j), and $\delta_{ij,kp} = 0$ otherwise. In (2), each column of the decision variable vector corresponds to a feasible service route of each vehicle. The set P_k can be extremely large and difficult to enumerate at once. Nevertheless, the structure of (2) allows us to iteratively generate a new feasible route (and hence

a new column/decision variable) as needed. The next section will present the application of the CG and CLS to solve the PDP as defined in (2).

HYBRID SOLUTION ALGORITHM

General Framework and Column Generation Method

The general framework of the proposed hybrid algorithm is similar to that proposed in Xu et al. (*7*) in which the CG is used to solve the relaxed problem (RP) of (2). However, in our proposed algorithm the time-window constraint remains in the RP and the relaxation is only made against the integer requirement of the decision variables in (2). Each feasible service route with an assigned sequence of the jobs is considered as a column in (2) in which a new decision variable associated with this route, λ_{pk}, is created. The CLS is used to solve a sub-problem which is to find a new route that minimizes the reduced cost (defined by the dual variables of the RP). The CLS also takes into account all practical constraints of the PDP. The new route found by the CLS, if its reduced cost is negative, is then included as a new column in the RP. The algorithm then solves the RP again as a linear program (e.g., using CPLEX) to find a new solution and to form a new sub-problem. The proposed hybrid algorithm iterates between the RP and sub-problem until a set of new routes with negative reduced costs can not be found from the sub-problem. The outputs from the RP of (2) may not be integer solutions. Thus, the Branch and Bound (B&B) method will be applied to (2) but formulated with only those columns generated by the CG.

Let RP be the relaxed problem of (2) by excluding the integral requirement:

$$
\begin{aligned}
(\mathrm{RP}) \qquad & \min_{\lambda_{pk}} \sum_{k\in K}\sum_{p\in P_k} \overline{c}_{pk}\lambda_{pk} \\
& s.t. \\
& \sum_{p\in P_k} \lambda_{pk} \le 1 && \forall k \in K \\
& \sum_{k\in K}\sum_{p\in P_k} e_{spk}\lambda_{pk} = 1 && \forall s \in S \\
& \sum_{k\in K}\sum_{p\in P_k} a_{ipk}\lambda_{pk} \le l_i && \forall i \in N \\
& 0 \le \lambda_{pk} \le 1 && \forall p \in P_k; \forall k \in K.
\end{aligned} \tag{3}
$$

Let RP' be the restricted version of the RP problem with a subset $P' \equiv \bigcup_{k\in K} P'_k \subseteq \bigcup_{k\in K} P_k \equiv P$ of all possible service paths of all vehicles and P' be the set of the service routes with assigned jobs as generated by the CLS. The RP' can then be formulated as:

$$
\begin{aligned}
(\text{RP}') \qquad & \min_{\lambda_{pk}} \sum_{k\in K}\sum_{p\in P'_k} \overline{c}_{pk}\lambda_{pk} \\
& s.t. \\
& \sum_{p\in P'_k} \lambda_{pk} \le 1 && \forall k \in K \\
& \sum_{k\in K}\sum_{p\in P'_k} e_{spk}\lambda_{pk} = 1 && \forall s \in S \\
& \sum_{k\in K}\sum_{p\in P'_k} a_{ipk}\lambda_{pk} \le l_i && \forall i \in N \\
& 0 \le \lambda_{pk} \le 1 && \forall p \in P'_k; \forall k \in K.
\end{aligned} \tag{4}
$$

The general framework of the algorithm can be summarized as follows:

Step 1 Generate a set of service routes (columns) P' which satisfies all constraints in (2) by using CLS, i.e., find the set of services that gives an initial feasible solution to the PDP. Note that in this step the CLS is only used to find a feasible solution with the minimum number of vehicles.

Step 2 With P', formulate RP', as defined in (4), and then apply the Simplex method to solve RP' to obtain λ_{pk} and dual variables.

Step 3 Define the reduced cost for each route based on λ_{pk} and dual variables obtained from Step 2, and then use the CLS to find a set of new service routes (one for each vehicle) to minimize the total reduced costs. Let $\tilde{P}$ be the set of these new service routes found by the CLS.

Step 4 If the reduced costs from $\tilde{P}$ is negative, then let $P' = P' \cup \tilde{P}$, and return to Step 2; otherwise proceed to Step 5.

Step 5 If all λ_{pk} obtained from the Simplex method in Step 2 are binary variables, then terminate the algorithm in which λ_{pk} is the solution of the PDP. Otherwise, apply the B&B approach to the RP' with the integral constraint to get an integer solution of λ_{pk}.

As reported in Xu et al. (*7*), λ_{pk} found by the Simplex method are often already binaries. Thus, the B&B approach should not take too much time in solving the problem in Step 5.

The optimal solution from RP' will be the solution of the original RP, if a new set of service routes (columns) with a negative reduced cost can be found. The reduced cost is defined from the optimal dual variables associated with the solution to the RP'. From (4), the reduced cost can be defined as:

$$\hat{c}_{pk} = \overline{c}_{pk} - \sum_{\forall i\in N}\sum_{\forall j\in N} \left(c_{ij} - \pi_i\right) x_{ijk} - \pi_o , \tag{5}$$

where π_i is the optimal dual variable associated with the third constraint in (4) for node *i*; π_o is the dual variables associated with the first constraint in (4) for route *p* of vehicle *k*. To find new columns, one needs to solve (5) for all vehicles simultaneously. This sub-problem is a set-partitioning problem. The CLS will be used to solve this sub-problem which is still a NP-hard problem. In fact, it is also possible to solve the sub-problem separately for each vehicle if (4) only involves constraints defined separately for each vehicle. This may limit the flexibility of the

PDP to include more complex constraints involving the whole fleets. In addition, the complexity for the CLS in solving the sub-problem with all vehicles simultaneously is not significantly higher than solving the sub-problem for each vehicle separately. Note that using the CLS alone can also find a feasible solution for PDP, but in the expense of the solution quality. Therefore, we only apply the CLS to construct an initial set of feasible columns and to find the solution of the sub-problem (SP).

Constrained Local Search Method for Generating New Service Routes

The CLS is a type of meta-heuristic optimization method. The algorithm will start with an initial solution and then iteratively moves to neighbour feasible solutions. Thus, the definition of the neighbourhood solutions must be defined. The CLS will decide on the move to a neighbour solution probabilistically.

The sub-problem of RP' can be considered as a set partitioning problem (SP). The SP involves assigning each job, $s \in S$, to different vehicles, $k \in K$. Fig. 1 illustrates the problem in the structure of a set-partitioning problem. The upper column represents vehicle $k \in K$ (e.g., K1 or K2). Under each upper column, the sub-columns (e.g., columns 1-4 under K1) are associated with the job sequence for that vehicle. The number of sub-columns for each vehicle is defined by the maximum number of jobs that vehicle is allowed to take. Each job, $s \in S$, is associated with two rows (pickup and delivery tasks). For instance, rows 2+ and 2- represent the pickup and delivery tasks for job 2 in this example.

If the job s is assigned to vehicle k in which the pickup task is defined as the w_k-th job of vehicle k, then the value in the sub-column w_k of vehicle k and in the row s+ will be set to 1, i.e., $\theta_{s+}^{k,w_k} = 1$. Consequently, the delivery task of job s must also be allocated to vehicle k, say defined as task w_k' ($w_k < w_k'$), in which $\theta_{s-}^{k,w_k'} = 1$. Otherwise, $\theta_{s+}^{k,w_k} = \theta_{s-}^{k,w_k'} = 0$. Denote o_{s+}^k and o_{s-}^k as the job numbers of the pickup and delivery of job s by vehicle k respectively. Denote $\hat{l}_t = l_i - \pi_t l_i$ as the upper bound of the pickup and delivery late times for each customer.

From Fig. 1, there are two jobs with four tasks (pickup and delivery). There are two vehicles, K1 and K2, and each vehicle can accept only four tasks. The cell values in the current example indicate that the job 2 is assigned to vehicle K1 in which the pickup and delivery tasks are its first and third tasks respectively. Similarly, job 1 is assigned to vehicle K2. This matrix representation will be adopted in the CLS to solve the SP.

	K1				K2			
	1	2	3	4	1	2	3	4
1+	0	0	0	0	0	1	0	0
2+	1	0	0	0	0	0	0	0
1-	0	0	0	0	0	0	1	0
2-	0	0	1	0	0	0	0	0

FIGURE 1 Example of representation of PDP as a set-partition problem.

Two types of constraints are defined for the SP following the CLS setting *(3)*: soft and hard constraints. The hard constraints involve all practical constraints of the PDP which are (6)-(9) below where W denotes the maximum number of tasks allowed for each vehicle. Note that

any other constraints can also be introduced without changing the search operators of the CLS. This allows for a more flexible formulation of the PDP.

$$h_o = \max\left(0, \sum_{s\in S}\left\lceil\left|\sum_{k\in V}\sum_{w_k=1}^{W}\left(\theta_{s+}^{k,w_k}-1\right)\right|\right\rceil\right), \tag{6}$$

$$h_{c1} = \max\left(0, \sum_{s\in S}\sum_{k\in K}\left\lceil\left|\sum_{w_k=1}^{W}\left(\theta_{s+}^{k,w_k}-\theta_{s-}^{k,w_k}\right)\right|\right\rceil\right), \tag{7}$$

$$h_{c2} = \sum_{s\in S}\sum_{k\in V}\left(\max\left(0, o_{s+}^{k}-o_{s-}^{k}\right)\right), \tag{8}$$

$$h_l = \sum_{k\in K}\sum_{i\in N}\left(\max\left(0, y_{ik}-Q_k\right)\right)\varepsilon_l, \tag{9}$$

$$h_t = \sum_{k\in K}\sum_{i\in N}\left(\max\left(0, a_{ik}-\hat{l}_i\right)\right). \tag{10}$$

(6) requires each order to be assigned to only one vehicle. (7) requires the order to be picked up and delivered during the same trip. (8) and (9) ensure that the orders of pickup and delivery tasks are in the correct sequences, and the vehicle load is less than vehicle capacity. Note that ε_l is an additional weight given to the h_l constraint. (10) imposes the constraint on the pickup and delivery times. Let $H = h_o + h_{c1} + h_{c2} + h_l + h_t$ to be the total violation function of the hard constraints. H is required to be 0 due to the definition of the hard constraint. The soft constraint is associated with the objective function of the sub-problem, i.e.,

$$\gamma_v = \max\left(0, \sum_{k\in V} v_k\right), \tag{11}$$

$$\gamma_d = \max\left(0, \sum_{k\in V}\sum_{i\in N}\sum_{j\in N}\hat{c}_{ij}x_{ijk}\right), \tag{12}$$

where γ_v is the total number of vehicles used (v_k = 1 if vehicle k is used and v_k = 0 otherwise). γ_d is the total reduced costs in which $\hat{c}_{ij} = c_{ij} - \pi_i$. Let $\Omega = \gamma_v + \gamma_d$ which represents the total soft constraint level.

The CLS will move from a current solution to its neighbour solutions with the aim to reduce $\Theta = \Omega + H$. The neighbourhood search operators adopted in the CLS are (i) intra-vehicle trial flip and (ii) inter-vehicle trial flip. The CLS will ensure the satisfaction of the hard constraints as defined in (6)-(8) by the specifications of the search operators. The constraint (9) and other additional constraints will be considered by the neighbourhood movement decision. The overall steps of the CLS can be defined as follows:

Overall CLS algorithm

Step 1 Initialize by forming the SP matrix (denoted by A) (see Figure. 1) and then for each job $s \in S$ randomly select a vehicle for the job. Then, randomly select a task sequence, $1 \le w_k < W$, for the pickup task, and then set the row $s+$ and column w_k to be 1.

Similarly, randomly select w'_k, $W \geq w'_k > w_k$, for the delivery task and set the row *s*- and column w'_k to be 1. After assigning all jobs, evaluate $\Theta = \Omega + H$ for the solution A.

Step 2 Set $u = 1$.

Step 3 Perform the intra-vehicle trial flip to define a set of neighbourhood solutions.

Step 4 If all neighbourhood solutions of A as found by the intra-vehicle trial flip cannot improve $\Theta = \Omega + H$, then perform the inter-vehicle trial flip.

Step 5 Set $u = u + 1$; if $u >$ *maxIter* then terminate; otherwise return to Step 3.

Note that the *maxIter* parameter is predefined by the user. The details of the intra-vehicle trial flip and inter-vehicle trial flip are as follows:

Intra-vehicle Trial Flip

There are two stages in the intra-vehicle trial flip operation: the exchange of the job sequences and the insertion of the jobs into empty job sequences. The pseudo code for the exchange phase of the intra-vehicle trial flip is as follows:

Intra-vehicle trial flip: exchange phase

Step 1 Randomly select a vehicle $k \in K$ and a column $1 \leq w_k \leq W$ with an assigned task (either pickup or delivery). Let r_1 be the row with value of 1 of the randomly selected column. Set $e = 0$.

Step 2 Set $e = e + 1$. Randomly select another column, w'_k, with an assigned task (either pickup or delivery). Let r_2 be the row of column w'_k with the value of 1.

Step 3 Exchange the values in the rows r_1 of columns w_k and w'_k. Similarly exchange the values in the rows r_2 of columns w_k and w'_k. Define the new solution as $A'(e)$. Evaluate $\Theta'(e)$ and h_{c2}. If $h_{c2} \neq 0$, return to Step 3. Otherwise, go to Step 4.

Step 4 If $e >$ *maxSampling*, go to Step 5. Otherwise return to Step 2.

Step 5 Let Θ' be $\min(\Theta'(e) : e = 1, \ldots, \text{maxSampling})$. If $\Theta < \Theta'$, then set $A = A'(e)$ and call the insertion phase. Otherwise, call inter-vehicle trial flip.

Note that the parameter *maxSampling* is predefined by the users and represents the number of neighbourhoods explored by each step of the CLS.

The exchange phase involves a random selection of a vehicle (normally those with constraint violation) and a column of that vehicle with an assigned task. This is illustrated in Figure 2a in which column 3 of vehicle K1 is randomly selected. Then, row 2+ is identified as the row with 1. In Step 2, the operation then randomly selects the second column which is assumed to be column 1 in this example in which row 3+ is identified. The values in these two rows of columns 3 and 1 are then exchanged as shown in Fig. 2b. For this example the new solution (in Fig. 2b) violates the constraint on the sequence of the tasks, i.e., the pickup task of job 3 is after the delivery task of the same job. Thus, this neighbour solution will not be considered. A new column will then be randomly selected instead (assume to be column 2 in this example). The exchange of the values can then be carried out as shown in Fig. 2c. For this

solution, the sequence of the tasks is feasible. Thus, the algorithm will evaluate the $\Theta'(e)$ of this solution.

The algorithm will carry out the same operation for a number of times as defined by the users (i.e., *maxSampling* parameter). Then, the solution with the lowest $\Theta'(e)$ among all feasible neighbourhood solutions will be selected (as define in Step 5). If $\Theta'(e)$ is less than the value of the original solution, then the search will move to this new solution and perform the insertion phase. Otherwise, the search will remain at the original solution and perform the inter-vehicle flip instead.

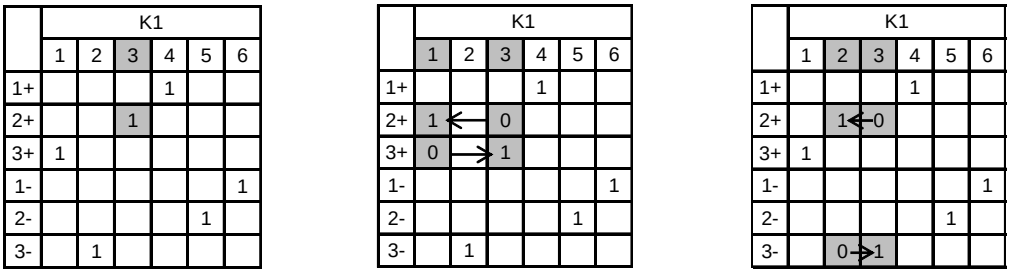


FIGURE 2 Intra-vehicle trial flip operation: exchange phase.

The insertion phase is described as follows:

Intra-vehicle trial flip: insertion phase

Step 1 Randomly select a vehicle $k \in K$ and a column $1 \le w_k \le W$ of that vehicle with a pickup task. Let r_1 be the row with the cell value of 1 under the randomly selected column. Let *s* be the pickup task associated with column w_k. Set r_2 to be *s*- and w_k' to be the column of the cell in row r_2 with the cell value of 1. Set $e = 0$.

Step 2 Identify all columns in which all cells under those columns have value of 0 (i.e., no task assigned to these vehicle sequences). Let Δ be the set of these column numbers in the ascending order in which $\Delta(b)$ refers to the column number in the order *b* in the set Δ.

Step 3 Change the value of column w_k and row r_1 to 0 and the value of column $\Delta(1)$ and row r_1 to 1. Then, exclude $\Delta(1)$ from Δ and include column w_k to Δ. Let $|\Delta|$ denote the size of this set.

Step 4 Set $e = e + 1$.

Step 5 Change the value of column $\Delta(e)$ and row r_2 to 1 and the value in the cell under column w_k' and on row r_2 to 0. Define this solution as $A'(e)$ and evaluate $\Theta'(e)$.

Step 6 If $e = |\Delta|$ then go to Step 7. Otherwise, return to Step 4.

Step 7 Compare the values of $\Theta'(e)$ for all $e = 1,\ldots, |\Delta|$ and define *e'* as the solution with the lowest $\Theta'(e)$. Set A = $A'(e')$.

Figure 3 illustrates the insertion phase of the intra-vehicle flip operation. From this example, let suppose that column 4 of vehicle K1 is randomly selected (see Fig. 3a) in Step 1. Then, as defined in Step 1, row 2+ can be identified, and also column 6 and row 2- can be identified (as the corresponding delivery task of job 2). Then, the list of columns without any tasked assigned to can be identified as {1, 2, 5, 8} in Step 2. The values in columns 4 and 1 on row 2+ are then exchanged as shown in Fig. 3b. Then, $\Delta = \{2,4,5,8\}$. With this set, there are only four possible moves. Fig. 3b, 3c, and 3d show three examples of the exchanges (out of four possible moves) of the delivery task of job *w* from under its original column 6 to columns 2, 4, and 8 respectively.

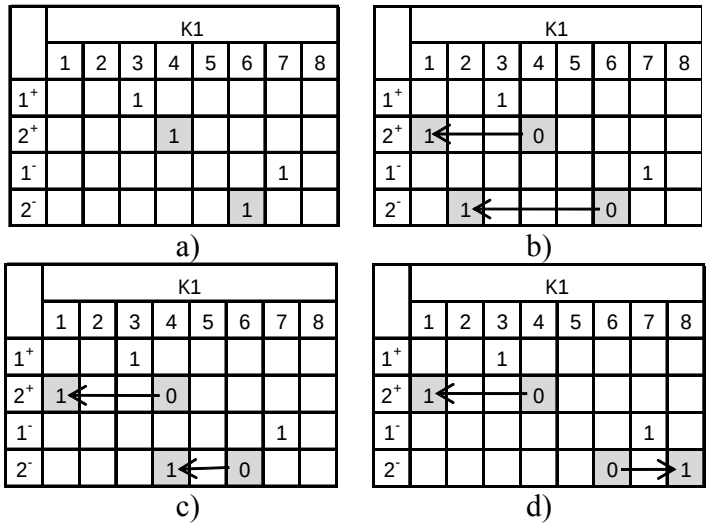


FIGURE 3 Examples of in-vehicle trial flip operation: insertion phase.

Inter-vehicle Trial Flip

The inter-vehicle trial flip will be called if the exchange phase of the intra-vehicle trial flip cannot find a solution with lower Θ'. With a selected vehicle and column from the intra-vehicle trial flip (from the exchange phase), the inter-vehicle flip will randomly select other vehicles. Then, the insertion phase will be carried out but with the columns of other vehicles. With a randomly selected vehicle, the set Δ will be created and then Steps 3-7 of the insertion phase will be carried out. The CLS will move to a new solution whether $\Theta' < \Theta$ or not. This will help the CLS to avoid local optima.

EXPERIMENT WITH BENCHMARK PROBLEMS AND REAL-WORLD CASE

Benchmark Tests Results

The proposed hybrid algorithm is applied to the benchmark tests as defined in (*5*). The benchmarks are categorized into two types, LR (100 randomly distributed pickup/delivery points) and LRC (100 randomly combined cluster pickup/delivery points), shown in Table 1a. See the detail definitions of these two problem classes in (*5*). In addition, the LRC typed-

problem with 200 pickup/delivery points as tested in (*6*) are also considered (see Table 1b). The algorithm is implemented in JAVA language. The tests are carried out with Pentium IV (2.8 GHz) computer under Windows OS. ε_1 in (9) and *maxIter* are set to be 10 and 20,000 respectively.

TABLE 1a) Test Results against 100 Points PDP Benchmarks

Problem	Best known			Full				
	veh	dist	ref.	avg. dist	avg. veh	best dist	best veh	avg. time (s)
LR101	19	1650.8	Li & Lim	1650.8	19	1650.8	19	127
LR102	17	1487.57	Li & Lim	1529.11	17	1515.88	17	103
LR103	13	1292.68	Li & Lim	1349.39	13	1317.45	13	114
LR104	9	1013.39	Li & Lim	1061.54	9	1016.93	9	132
LR105	14	1377.11	Li & Lim	1406.32	14	1386.27	14	98
LR106	12	1252.62	Li & Lim	1257.82	12	1256.66	12	146
LR107	10	1111.31	Li & Lim	1111.31	10	1111.31	10	145
LR108	9	968.97	Li & Lim	968.97	9	968.97	9	123
LR109	11	1208.96	SAM	1228.58	11	1208.96	11	134
LR110	10	1159.35	Li & Lim	1191.02	10	1159.35	10	166
LR111	10	1108.9	Li & Lim	1108.9	10	1108.9	10	183
LR112	9	1003.77	Li & Lim	1005.43	9	1003.77	9	177
LRC101	14	1708.8	Li & Lim	1717.61	14	1708.8	14	131
LRC102	12	1558.07	SAM	1570.12	12	1558.07	12	146
LRC103	11	1258.74	Li & Lim	1272.33	11	1258.74	11	130
LRC104	10	1128.4	Li & Lim	1138.96	10	1128.4	10	125
LRC105	13	1637.62	Li & Lim	1644.05	13	1637.62	13	147
LRC106	11	1424.73	SAM	1424.73	11	1424.73	11	134
LRC107	11	1230.15	Li & Lim	1231.18	11	1230.15	11	124
LRC108	10	1147.43	SAM	1155.58	10	1147.43	10	112

TABLE 1b) Test Results against 200 Points PDP Benchmarks

Problem	Best known			Full				
	veh	dist	ref.	avg. dist	avg. veh	best dist	best veh	avg. time (s)
LR1_2_1	20	4819.12	Li & Lim	5324.58	20	4948.79	20	1397
LR1_2_2	17	4621.21	RP	4749.88	18	4614.82	18	594
LR1_2_3	15	3612.64	TS	3912.3	15	3870.36	15	1111
LR1_2_4	10	3037.38	RP	3400.55	11	3111.16	11	852
LR1_2_5	16	4760.18	BVH	4721.72	17	4392.35	17	626
LR1_2_6	14	4175.16	BVH	4329.93	15	4218.42	15	699
LR1_2_7	12	3550.61	RP	3715.76	13	3468.55	13	1039
LR1_2_8	9	2784.53	RP	3005.28	9.63	2915.95	9	2235
LR1_2_9	14	4354.66	RP	4650.48	14.57	4404.79	14	1540
LR1_2_10	11	3714.16	RP	3792.87	12	3602.41	12	1324
LRC1_2_1	19	3606.06	SAM	3639.96	19	3625.46	19	1265
LRC1_2_2	15	3673.19	BVH	3958.43	16	3719.88	16	1407
LRC1_2_3	13	3161.75	BVH	3291.08	14	3371.52	14	1584
LRC1_2_4	10	2631.82	RP	2913.63	11	2713.94	11	2323
LRC1_2_5	16	3715.81	BVH	4043.92	17	4015.44	17	1425
LRC1_2_6	17	3368.66	SAM	3463.06	17	3441.72	17	1338
LRC1_2_7	14	3668.39	RP	3677.115	17	3580.77	15	1659
LRC1_2_8	13	3174.55	RP	3316.6	14	3228.94	14	1506
LRC1_2_9	13	3226.72	RP	3433.04	14	3307.79	14	1131
LRC1_2_10	12	2951.29	RP	3258.5	12.83	3409.05	12	744

BVH (4); Li & Lim (5); SAM (7); RP(6); TS (8)

Table 1 presents the best known results from the literature in terms of the number of optimal vehicles used (column 2) and total transportation distant (column 3) for all benchmark tests. Due to the heuristic nature of the proposed algorithm, the algorithm is applied to each benchmark case 20 times separately to collect the statistics of the results, e.g., best found and average. Table 1 reports the results found by the proposed hybrid algorithm (average values over 20 runs and the best found) for all benchmark tests; in which the results in columns 5-9 show the average travel distance, average number of vehicles used, best found total distant, best found number of vehicles used, and average processing time used in seconds in that order.

From Table 1, the algorithm proposed performs satisfactorily against all benchmark tests as compared to the best found solutions reported in the literature. In particular for the tests with 100 jobs (Table 1a), the numbers of vehicles used found by our proposed algorithm are the same as the best results reported in the literature for all benchmark tests. In some cases (i.e., LR109, LRC102, LRC106, and LRC108), the algorithm can even find better solutions compared to those reported in (*5*). However, for the tests with 200 orders (Table 1b), the solutions found by the proposed algorithm are slightly worse than the best results reported in the literature. This illustrates the trade-off between the performance of the algorithm and its flexibility. The proposed algorithm is very flexible for including additional constraints. On the other hand, most algorithms with the best known results are problem specific to some extent in which some specific heuristics are designed and used in the algorithms. Thus, with these specific heuristics other PDP constraints cannot be introduced without modifying the algorithms. This issue will be illustrated in the next section when a constraint on the arrival times of vehicles back to depots is introduced.

A Real World PDP Case Study of a Vehicle-Carrier Company

The real world case study considered in this paper is related to a vehicle-carrier company in Thailand. The company offers services to transport cars in different areas in Thailand. The main origins of its customers are car factories and second-hand car depots. Similarly the main destinations of the jobs are official car dealers and second-hand car dealers around Thailand. The car PDP problem in this case involves an additional constraint on the arrival times of the trailers back to the company depot (due to security and safety reasons). This constraint can be defined as:

$$h_t = \max\left(0, \sum_{k \in V} \sum_{i \in D} \left(a_{ik} - l_i\right)\right) \cdot \alpha \tag{13}$$

where D defines the set of company depots and α is the multiplier similar to ε. This is similar to the time-window constraint for the jobs or working-hour constraint for drivers. The proposed algorithm, in particular the CLS, can solve the PDP with this new constraint straightaway without any modification to its search operators or solution algorithm. In the test, α is set to be 10 and h_t is included in the hard constraint violation function of the CLS. Table 2 shows five datasets for the tests with different numbers of depots, types and numbers (#) of trailers, and numbers of jobs. The datasets 1-3 are typical problems encountered in the day to day by operation of the company. The datasets 4 and 5 are occasional cases with relatively large numbers of jobs. These occasional cases normally occur when several car manufacturers release new models during the same period.

TABLE 2 Datasets for the Real-World Problem

Data Set	Depot	Trailers		Orders	PD Points
		#	Types		
1	1	13	2	100	42
2	2	15	2	150	52
3	2	17	3	200	70
4	3	18	3	250	86
5	3	22	3	300	110

TABLE 3 Test Results with the Real-World Problem

Data Set	Laden KM			Empty KM			CPU (s)
	Manual	DSS	Diff (%)	Manual	DSS	Diff (%)	
1	7,232	6,445	10.87	7,010	6,108	12.86	18
2	12,756	11,576	9.25	13,516	12,100	10.47	23
3	29,975	26,548	11.43	20,350	17,553	13.74	46
4	37,550	32,453	13.57	34,811	30,090	13.56	63
5	39,953	34,259	14.25	38,055	32,182	15.43	112

In the Table 3, columns DSS and Manual show the results obtained by the proposed algorithm and current plans of the company (generated manually) respectively. The columns Laden KM and Empty KM show the total vehicle-kilometres covered by the loaded and empty vehicles respectively. From Table 3, the algorithm can potentially help reducing the vehicle-kilometres covered by both the loaded and empty vehicles at least 9.25% and 10.47% respectively for all test cases. This can potentially result in a substantial saving in terms of the company logistics costs. The computational times in all cases are also acceptable in which the runtime for the largest problem (Data Set 5) is only around 2 minutes. This illustrates the practicality of the proposed algorithm in handling a large scale realistic PDP.

CONCLUSION AND DISCUSSION

The paper proposed a hybrid method combining the column generation technique (CG) and constrained local search (CLS) algorithm for tackling the Pickup and Delivery problem (PDP). The algorithm developed is non-domain specific which allows any additional practical constraints to be flexibly considered in the PDP. The hybrid CG and CLS also enhances the stability and robustness of the algorithm in which the CG helps reducing the problem size (which is typically very large for the realistic PDP) and the CLS provides the robustness in handling complex constraints of the PDP. The algorithm proposed was tested against the well-known benchmark cases. The proposed algorithm performed satisfactorily in all tests compared to the best-known results. The algorithm was also applied to the real-world PDP based on the problem of a car-carrier company in Thailand. Five different scenarios of job requests were chosen for the tests. In all scenarios, the plans found by the proposed algorithm can significantly decrease the total distances of the loaded and empty vehicles operated by the company compared to the current manual plans (at least around 9%-14% and 10%-15% respectively). Future research will

extend the algorithm to take into account uncertainties of travel times and dwell times of vehicles in the PDP.

ACKNOWLEDGEMENT

This research has been supported partially by a grant from the Thai Research Fund (TRF), the National Electronics and Computer Technology Center (NECTEC), the University Research Grant of the Hong Kong Polytechnic University (Project no: A-PH65), and the start-up grant R-264-000-229-112 from the National University of Singapore.

REFERENCES

1. Bent, R., and P.V. Hentenryck. A Two-stage Hybrid Algorithm for Pickup and Delivery Vehicle Routing Problems with Time Windows. Computers & Operations Research. Vol. 33, No.4, 2006, pp. 875-893.
2. Desaulniers, G., J. Desrosiers, and M.M. Solomon. *Column generation*. Springer, New York, N.Y., 2005.
3. Indra-Payoong, N., R. Kwan, and L. Proll. Rail Container Service Planning: A Constraint-Based Approach. In *Multidisciplinary Scheduling: Theory and Applications* (G. Kendall, S. Petrovic, E. Burke, and M. Gendreau), Springer-Verlag, 2005, pp. 343-368.
4. Nanry, W.P., and J. Wesley Barnes. Solving the Pickup and Delivery Problem with Time Windows using Reactive Tabu Search. *Transportation Research Part B*, Vol. 34, No. 2, 2000, pp. 107-121.
5. Li, H., and A. Lim. A Metaheuristic for the Pickup and Delivery Problem with Time Windows. *International Journal on Artificial Intelligent Tools*, Vol. 12, No. 2, 2003, pp. 173-186.
6. Ropke, S., and D. Pisinger. An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation Science*, Vol. 40, No. 4, 2006, pp. 455-472.
7. Xu, H., Z-L. Chen, S. Rajagopal, and S. Arunapuram. Solving a Practical Pickup and Delivery Problem. *Transportation Science*, Vol. 37, No. 3, 2003, pp. 347-364.
8. TetraSoft A/S. *MapBooking Algoritm for Pickup and Delivery Solutions with Time Windows and Capacity restraints*. 2003.

List of Tables

List of Figures

TABLE 1a) Test Results against 100 Points PDP Benchmarks

Problem	Best known			Full				
	veh	dist	ref.	avg. dist	avg. veh	best dist	best veh	avg. time (s)
LR101	19	1650.8	Li & Lim	1650.8	19	1650.8	19	127
LR102	17	1487.57	Li & Lim	1529.11	17	1515.88	17	103
LR103	13	1292.68	Li & Lim	1349.39	13	1317.45	13	114
LR104	9	1013.39	Li & Lim	1061.54	9	1016.93	9	132
LR105	14	1377.11	Li & Lim	1406.32	14	1386.27	14	98
LR106	12	1252.62	Li & Lim	1257.82	12	1256.66	12	146
LR107	10	1111.31	Li & Lim	1111.31	10	1111.31	10	145
LR108	9	968.97	Li & Lim	968.97	9	968.97	9	123
LR109	11	1208.96	SAM	1228.58	11	1208.96	11	134
LR110	10	1159.35	Li & Lim	1191.02	10	1159.35	10	166
LR111	10	1108.9	Li & Lim	1108.9	10	1108.9	10	183
LR112	9	1003.77	Li & Lim	1005.43	9	1003.77	9	177
LRC101	14	1708.8	Li & Lim	1717.61	14	1708.8	14	131
LRC102	12	1558.07	SAM	1570.12	12	1558.07	12	146
LRC103	11	1258.74	Li & Lim	1272.33	11	1258.74	11	130
LRC104	10	1128.4	Li & Lim	1138.96	10	1128.4	10	125
LRC105	13	1637.62	Li & Lim	1644.05	13	1637.62	13	147
LRC106	11	1424.73	SAM	1424.73	11	1424.73	11	134
LRC107	11	1230.15	Li & Lim	1231.18	11	1230.15	11	124
LRC108	10	1147.43	SAM	1155.58	10	1147.43	10	112

TABLE 1b) Test Results against 200 Points PDP Benchmarks

Problem	Best known			Full				
	veh	dist	ref.	avg. dist	avg. veh	best dist	best veh	avg. time (s)
LR1_2_1	20	4819.12	Li & Lim	5324.58	20	4948.79	20	1397
LR1_2_2	17	4621.21	RP	4749.88	18	4614.82	18	594
LR1_2_3	15	3612.64	TS	3912.3	15	3870.36	15	1111
LR1_2_4	10	3037.38	RP	3400.55	11	3111.16	11	852
LR1_2_5	16	4760.18	BVH	4721.72	17	4392.35	17	626
LR1_2_6	14	4175.16	BVH	4329.93	15	4218.42	15	699
LR1_2_7	12	3550.61	RP	3715.76	13	3468.55	13	1039
LR1_2_8	9	2784.53	RP	3005.28	9.63	2915.95	9	2235
LR1_2_9	14	4354.66	RP	4650.48	14.57	4404.79	14	1540
LR1_2_10	11	3714.16	RP	3792.87	12	3602.41	12	1324
LRC1_2_1	19	3606.06	SAM	3639.96	19	3625.46	19	1265
LRC1_2_2	15	3673.19	BVH	3958.43	16	3719.88	16	1407
LRC1_2_3	13	3161.75	BVH	3291.08	14	3371.52	14	1584
LRC1_2_4	10	2631.82	RP	2913.63	11	2713.94	11	2323
LRC1_2_5	16	3715.81	BVH	4043.92	17	4015.44	17	1425
LRC1_2_6	17	3368.66	SAM	3463.06	17	3441.72	17	1338
LRC1_2_7	14	3668.39	RP	3677.115	17	3580.77	15	1659
LRC1_2_8	13	3174.55	RP	3316.6	14	3228.94	14	1506
LRC1_2_9	13	3226.72	RP	3433.04	14	3307.79	14	1131
LRC1_2_10	12	2951.29	RP	3258.5	12.83	3409.05	12	744

TABLE 2 Datasets for the Real-World Problem

Data Set	Depot	Trailers		Orders	PD Points
		#	Types		
1	1	13	2	100	42
2	2	15	2	150	52
3	2	17	3	200	70
4	3	18	3	250	86
5	3	22	3	300	110

TABLE 3 Test Results with the Real-World Problem

Data Set	Laden KM			Empty KM			CPU (s)
	Manual	DSS	Diff (%)	Manual	DSS	Diff (%)	
1	7,232	6,445	10.87	7,010	6,108	12.86	18
2	12,756	11,576	9.25	13,516	12,100	10.47	23
3	29,975	26,548	11.43	20,350	17,553	13.74	46
4	37,550	32,453	13.57	34,811	30,090	13.56	63
5	39,953	34,259	14.25	38,055	32,182	15.43	112

	K1				K2			
	1	2	3	4	1	2	3	4
1+	0	0	0	0	0	1	0	0
2+	1	0	0	0	0	0	0	0
1-	0	0	0	0	0	0	1	0
2-	0	0	1	0	0	0	0	0

FIGURE 1 Example of representation of PDP as a set-partition problem.

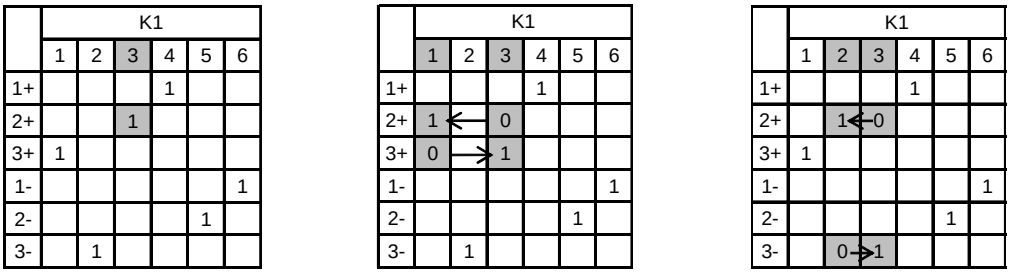


FIGURE 2 In-vehicle trial flip operation: exchange phase.

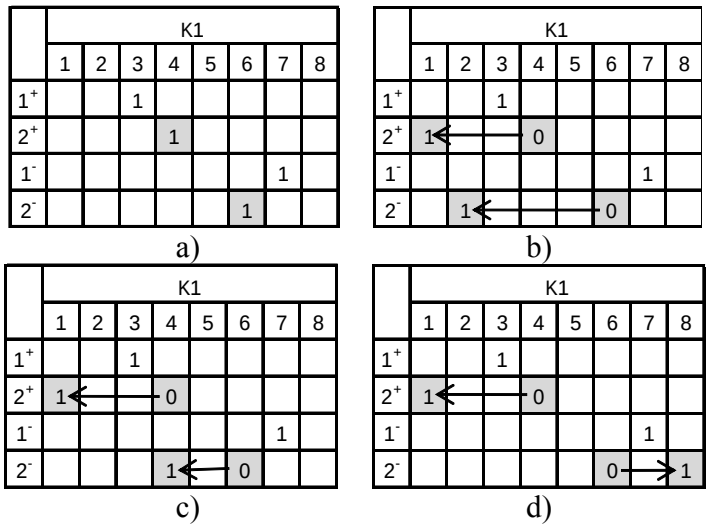


FIGURE 3 Examples of in-vehicle trial flip operation: insertion phase.

Constrained Local Search Method for Bus Fleet Scheduling Problem with Multi-depot with Line Change

Kriangsak Vanitchakornpong[1], Nakorn Indra-Payoong[1], Agachai Sumalee[2], and Pairoj Raothanachonkun[1]

[1] College of Transport and Logistics, Burapha University, Chonburi, Thailand
[2] Department of Civil and Structural Engineering, The Hong Kong Polytechnic University, Hong Kong
kriangsv@buu.ac.th, nakorn.ii@gmail.com, ceasumal@polyu.edu.hk, pairoj.iang@gmail.com

Abstract. This paper proposes a bus fleet scheduling model with multi-depot and line change operations with the aim to reduce the operating costs. The problem is constrained by various practical operational constraints, e.g. headway, travel time, and route time restrictions. A constrained local search method is developed to find better bus schedules. The method is tested with the case study of the Bangkok bus system with nine bus service lines covering around 688 trips per day. The test result shows that around 10% of the total operating costs could be saved by the optimized schedule.

Keywords: Bus scheduling, Multi-depot and line change, Constrained local search.

1 Introduction

Bus scheduling is a complex decision problem for transit operators. Generally, there are four planning steps involved: i) bus routes and headway determination, ii) bus timetabling, iii) vehicle scheduling, and iv) crew scheduling. These steps are highly interdependent. (i) and (ii) are rather long-term plans whereas the other two are short-term decisions. In this paper, we focus on the vehicle scheduling problem of bus services with multi-depot and line change operations. We also focus on developing a flexible algorithm that can accommodate additional side constraints commonly found in real cases. Since the multi-depot vehicle scheduling problem is NP-hard, we propose a local search method that uses a simple local move to obtain a good quality solution within a viable time. The proposed model and algorithm are evaluated with the real data from the bus system in Bangkok.

The paper is organized as follows: the next section reviews the literature on bus scheduling; the following two sections describe the model formulation and the test results from the Bangkok case study respectively. The conclusions and future research are then discussed in the last section.

2 Literature Review

Relevant literature on the bus fleet scheduling problem can be categorized as: i) problem formulation, ii) solution algorithm development; and iii) applications to real

M. Giacobini et al. (Eds.): EvoWorkshops 2008, LNCS 4974, pp. 669–678, 2008.

life problems. Since the multi depot vehicle scheduling problem (MDVS) is NP-hard, most of the algorithms proposed thus far in the literature are heuristic. Bertossi *et al.* [1] proposed a multi-commodity matching model for MDVS and developed two heuristics to solve the problem. Ribeiro and Soumis [2] formulated MDVS as a set partitioning (SP) problem. They also proposed a method based on the column generation (CG) and branch and bound methods. Lobel [3, 4] tackled MDVS with the problem size of around 7,000 trips. The problem is formulated as a multi-commodity flow problem, and the Lagrangian pricing method was adopted. The proposed algorithm has been employed by three bus companies in Berlin and Hamburg cities. Hadjar *et al.* [5] proposed a branch and cut algorithm for MDVS. They successfully solved the problem with around 800 trips. Pepin *et al.* [6] and Kliewer *et al.* [7] formulated the MDVS as a time-space model to reduce the number of decision variables. With the proposed model, any off-shelf optimization software can be adopted to solve the problem. The proposed method is tested with the bus data from the Munich city.

In this paper, we propose a different algorithm for MDVS. The method is based on a special type of local search algorithm which operates simple local moves to improve the solution. The algorithm proposed is arguably more flexible, and can easily accommodate additional side constraints. The users can easily introduce and/or modify the constraints without any changes to the algorithm structure. For instance, a strict time-window constraint can be easily converted to a time-window constraint with some tolerances.

3 Problem Formulation

The vehicle scheduling process will be analyzed after the bus timetable has been determined. The vehicle scheduling problem for bus systems involves assigning different buses to different scheduled trips so as to minimize the total operating costs whilst satisfying several practical constraints including (i) only one bus is assigned to a scheduled trip, and (ii) other constraints (e.g. time-window or route restriction).

In this paper, the MDVS is formulated as a set partitioning problem as shown in Fig. 1. Each row represents a scheduled trip, and each column represents a vehicle.

In Fig. 1, there are five scheduled trips with two different service lines: L1 and L2. Vehicle i1 and i2 belong to L1, and i3 and i4 belongs to L2. A feasible solution is shown in Fig. 1 in which for L1 bus i2 departs at 8:00 and 9:00, and i1 departs at 8:30; for L2, bus i3 and i4 depart at 8:15 and 8:35 respectively.

Timetabled trips		L1		L2	
		i1	i2	i3	i4
L1	8:00	0	1	0	0
	8:30	1	0	0	0
	9:00	0	1	0	0
L2	8:15	0	0	1	0
	8:35	0	0	0	1

Fig. 1. Set partitioning formulation for vehicle scheduling problem

To facilitate the discussion, the following notations are used: $Z = \{1,...,z\}$ denotes a set of depots, $L = \{1,...,l\}$ is the set of bus lines and $L_z' = \{1,...,l_z\}$ is the set of bus lines for depot z, $W = \{1,...,w\}$ is the set of scheduled trips, $W_l' = \{1,...,w_l\}$ is the set of trips of line l, $V = \{1,...,v\}$ is the set of buses, $V_l' = \{1,...,k_l\}$ is the set of buses of line l, $D_i = \{1,...,d\}$ is the ordered set of the trips served by bus i, Q_i is the allowable number of trips for bus i, c_{ij} is the fixed cost of bus i serving trip j, $x_{ij} = 1$ if bus i serving trip j; $x_{ij} = 0$ otherwise, e_{ij} is the starting time of trip j for bus i, t_{ij} is the trip time for bus i servicing trip j, s_l is the number of buses serving line l, w_l is the total trips of line l, k_l is the number of dedicated buses of line l, $f_i = 1$ if bus i services at least 1 trip; $f_i = 0$ otherwise, $a_{ij} = 1$ if bus i is used for the other lines when serving trip j; $a_{ij} = 0$ otherwise, $b_{ij} = 1$ if bus i is shared between depots in serving trip j; $b_{ij} = 0$ otherwise.

The bus vehicle scheduling problem can be considered as a constraint satisfaction problem in which the violation of hard constraints (operational constraints) is prohibited, and the objective function is converted into a soft constraint. In general, the constraint violation (v) can be written as:

$$Ax \le b \Rightarrow v = \max(0, Ax - b) \quad . \tag{1}$$

where A is coefficient value, x is decision variable, and b is a bound. A solution for the problem is achieved when i) the hard violation (H) equals to zero, and ii) the soft violation (S) is minimized.

3.1 Hard Constraints

In this paper, the hard constraints are further categorized into the basic and side constraints. The basic constraints are mainly concerned with the consistency of the solution, e.g. one trip should be assigned to one vehicle. On the other hand, the side constraints are related to additional operational constraints considered, e.g. time-window constraint on arrival time of the vehicle back to the depot.

Basic constraints: Constraint (2) below states that each scheduled trip j must be assigned to only one bus. h_p is the constraint violation level for this constraint.

$$h_p = \max\left(0, \sum_{j \in W} \left| \sum_{i \in V} x_{ij} - 1 \right| \right) \quad . \tag{2}$$

Constraint (3) ensures that the bus can only start the trip after finishing the previous assigned trip.

$$h_e = \max\left(0, \sum_{i \in V} \sum_{j \in D_i} \left[\left(e_{i,j-1} + t_{i,j-1} \right) - e_{ij} \right] \right) \quad . \tag{3}$$

Side constraints: These constraints are mainly for route-time constraint, line change and vehicle transfer operations, and relaxed hard time windows.

In general, the route-time constraint may be associated with vehicle range (due to fuel limit), maintenance period, and maximum driver's working hours which can be written as:

$$h_r = \max\left(0, \sum_{i \in V} \left(e_{i,m^*+1} - \left(e_{im^*} + t_{im^*} + bk\right)\right)\right) \quad . \tag{4}$$

Where $m^* = j \bmod m \quad \forall j \in D_i$, m is the longest continuous trip served by vehicle i, bk is the break time (unit: min), and h_r is the violation for the constraints.

Constraint (5) ensures that the bus cannot run longer than the daily allowable maximum working hours (its violation level is h_c):

$$h_c = \max\left(0, \sum_{i \in V} \left| \sum_{j \in W} x_{ij} - Q_i \right|\right) \quad . \tag{5}$$

Line change and vehicle transfer operations are proposed to reduce the operating costs by efficiently utilizing the current resources. Typically, the vehicles are assigned to specific lines (routes) and depots. With this fixed plan, some vehicles may be underutilized and some lines may be lack of vehicles at some periods due to the fluctuation of travel times and demands during a day. The line change operation is the assignment of a vehicle belonging to one line to serve a trip of another line. The multi-depot operation (vehicle transfer) allows the vehicles to arrive or depart from different depots from their original depots. Thus, the vehicle scheduling with multi-depot and line change operations can potentially increase the efficiency of the bus utilization. Nevertheless, some buses may not be available for this operation due to some contractual/practical issues (e.g. advertisement media contract on a particular line). The violation for the constraints can be formulated as:

$$h_m = \max\left(0, \sum_{l \in L} \left[\sum_{i \in V} \left(\sum_{j \in W_l'} x_{ij} \right) m_{il} \right]\right) \quad . \tag{6}$$

$$h_t = \max\left(0, \sum_{z \in Z} \sum_{i \in V} \left[\sum_{l \in L_z'} \left(\sum_{j \in W_l'} x_{ij} \right) n_{iz} \right]\right) \tag{7}$$

where m_{il}=1 if bus i on line l is prohibited to serve other lines, n_{iz}=1 if bus i on depot z cannot be transferred between depots. h_m and h_t are the constraint violation levels for the constraints on line change and vehicle transfer respectively.

An additional relaxed time-window constraint is introduced to allow for journey time variability. The late time-windows of ω in (3) are relaxed; where ω is the set of scheduled trips. The time-window constraint is converted to a soft constraint as:

$$h_{e^*} = \max\left(0, \sum_{i \in V} \sum_{j \in D_i} \left[\left(e_{i,j-1} + t_{i,j-1}\right) - e_{ij}\right]\left(1 - y_j\right)\right) \quad . \tag{8}$$

$$s_e = \max\left(0, \sum_{i \in V} \sum_{j \in D_i} \left[\left(e_{i,j-1} + t_{i,j-1}\right) - e_{ij}\right] y_j\right) \tag{9}$$

Note that (3) is substituted by (8). (9) is a soft constraint y_j=1 if $j \in \omega$; otherwise y_j=0.

3.2 Soft Constraint

The objective function of the problem is to minimize the total operating cost which can be formulated as a soft violation, s , as:

$$S = \max\left(0, \sum_{i \in N} \left[\sum_{j \in N} \left(c_{ij} x_{ij} + LC_i a_{ij} + VT_j b_{ij}\right) + FV_i f_i\right] + s_e\right) \tag{10}$$

FV_i is the fixed cost of running bus i. LC_j is the fixed cost of line change operation, and VT_j is the fixed transfer cost of the vehicles between depots. From (2) – (8), the hard violation can be defined as $H = h_p + h_r + h_c + h_m + h_t + h_{e^*}$, and the total constraint violation is $V = H + S$. Note that for the Bangkok bus system, the depot is used for the first and end bus stops; thus the deadhead trip is not considered.

4 Solution Methods

Typically, the bus travel time is defined as a fixed parameter. However, in a very congested traffic network (like Bangkok) there exists a large variation of bus travel time. Fig. 2 illustrates the travel time in a typical day for a bus service in Bangkok. The travel time variability can seriously affect the vehicle schedule. In this study, the travel time forecast model is based on the historical travel time in hourly timeslot of the day and the day of the week with some expert rules. The detail of this model will not be discussed here to the limited space.

The pre-processing step must be carried out by sequencing scheduled trips in an ascending order, i.e. $e'_{j-1} < e'_j \quad \forall j \in W$; where e'_j is the departure time for scheduled trip j . This process helps decreasing the complexity and size of the problem. Then, the constrained local search (CLS) [8] is used to solve the vehicle scheduling problem. The hard and soft constraints are represented in a constraint matrix. CLS employs the random variable selection strategy and simple variable flip as a local move. The move quality is assessed by its total constraint violation, V . The violation scheme and redundant constraints are also used to guide the search into more promising regions of the search space. In addition, the constraint propagation is

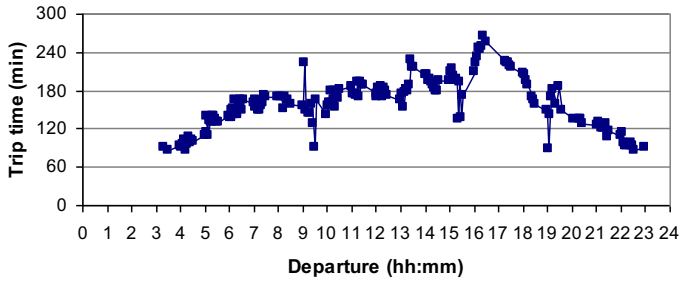


Fig. 2. Travel time variation

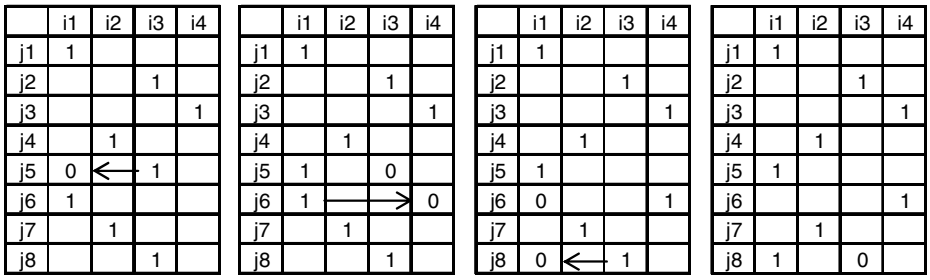


Fig. 3. Variable flip operations

maintained, i.e. once a bus is assigned to a particular scheduled trip the flip operations associated with infeasible trips (e.g. those overlapping with the current assigned trip) will be banned. This reduces the size of the search space and speed up the computation time. Fig. 3 illustrates the variable flip operations.

Rows in the matrix represent the scheduled trips and columns represent the vehicles. The flip operation is designed to ensure that a new solution will satisfy the hard constraints (e.g. one trip is assigned to one vehicle). Thus, the consistency of these constraints is always maintained. The overall pseudo-code of the CLS is depicted in Fig. 4.

Referred to Fig 4, the procedure of CLS can be summarized into three steps:

- **Step 1:** Constraint selection. After the initial assignment, A , the number of columns (nc in Line 5) are randomly selected in accordance with the violated constraints (columns) are given the priority. (Line 1-5)
- **Step 2:** Variable selection. CLS selects the row (variable assigned to 1) to perform a trial flip with other variables in the same row. (Line 8-12)
- **Step 3:** Move acceptance. CLS chooses the best V' amongst the flipped variables in every single iteration and assigns the best V' to the current solution, $A_l \leftarrow A'$. (Line 15-20)

Adaptive upper bound. Since the set partitioning formulation has a large number of columns and rows in the constraint matrix, the algorithm may spend too much computational time to obtain a feasible solution. It is possible to improve the CLS by introducing memory-based search, guided information obtained from linear

```
// Algorithm: CLS
// INPUT: soft and hard constraints
// OUTPUT: a best feasible solution found
1    A := initial solution assignment
2    A_b := best solution
3    V_b := the best violation
4    WHILE (iteration < stopping criterion) DO
5       nc := Random(numcol)
6       V_g := MAX_INTEGER
7       FOR i ← 1 TO nc
8          C_fst := select-columns (A)
9          P := select-variables(C_fst)
10         V_l := MAX_INTEGER
11         FOR j ← 1 TO Random(numcol)
12            A' := flip(C_fst, C_j, P)
13            H' := total hard violation(A')
14            V' := total violation(A')
15            IF H' = 0 AND V' < V_b
16               A_b ← A'
17               V_b ← V'
18            END IF
19            IF V' < V_l THEN
20               A_l ← A'
21            END IF
22         END FOR
23         IF V_l < V_g THEN
24            A_g ← A_l
25            V_g ← V_l
26         END IF
27      END FOR
28      A ← A_g
29   END WHILE
```

Fig. 4. The procedure of CLS

programming relaxation, or other search intensification techniques. In this study, the number of busses (columns) is gradually decreased and fixed iteratively when CLS find a better feasible solution; thus intensifying the search in more promising regions.

5 Computational Experiments

5.1 The BMTA Case Study

We test the proposed model and algorithm with the data from the Bangkok mass transit authority (BMTA), Thailand. There are currently 3,535 buses operating 108 routes (lines) clustered into 8 zones in the Bangkok city. In general, there are 3 - 4

depots for each zone, and each depot operates 5-10 bus lines. The bus schedules are currently determined by the BMTA planner manually once every few months without any computer-based tool. The schedule plan is rather ad hoc and often completely different from the actual services. For example, most of the buses are stuck in the traffic during rush hours creating the bus bunching problem.

5.2 Potential Cost Savings with Multi-depot and Line Changes

The proposed solution applies to the BMTA data with nine bus lines and 125 buses located in three depots. The model assumes that most buses are independently operated, i.e. can be utilized for the other lines or depots with the penalty cost (soft violation). The parameters in the model are set as follows: the fixed cost of each bus is 5,000 Baht/vehicle/day (1 Euro = 46 Baht). The line change cost is 100 Baht. The vehicle transfer cost is 500 Baht, which is excluded from running cost. The vehicle operating cost per Km is 30 Baht. The computational results are shown in Table 1.

The first column shows the tested lines and their depot and number of vehicles. The second column shows the number of scheduled trips for each line. The column Pure-line presents the data regarding the current operation including the number of buses required and the operating cost (in Baht). The column Line change (LC) reports the results based on the optimized vehicle schedule which allows for the line change operation. The next column then reports additional results when both line change and multi depot operations are considered. The last column shows the maximum reduction in terms of the required fleet size and total operating cost for each line. Time (sec) represents the computational time for particular test cases.

From Table 1, the schedules with line change can decrease the vehicle size from 85 (in pure line case) to 27 vehicles in total. In addition, when allowing the multi-depot operations the fleet size can be further reduced to 41 vehicles (almost 60% reduction). In terms of the operating costs, the multi-depot with line change operations can save around 91,000 Baht/day or 2.73 million Baht/month. From the computational point of

Table 1. Multiple-depot with line change operations

Line/	Trips	Pure-line			Line changes (LC)		Multi-depot with LC		Savings %	
#Bus		#Bus	Cost	Time(sec)	#Bus	Cost	#Bus	Cost	#Bus	Cost
Depot 1										
4 [18]	75	8	80687.5	2.8	7	82287.5	4	75687.5	50.00	6.20
72 [5]	78	8	80950	2.6	7	83450	7	86550	12.50	-6.92
205 [12]	76	11	94900	2.5	4	64700	4	75600	63.64	20.34
552 [15]	62	10	134630	2.5	5	115230	3	119630	70.00	11.14
					Time(sec) : 23.4					
Depot 2										
62 [14]	79	9	94770	2.8	10	104370	5	92870	44.44	2.00
77 [18]	101	12	155445	3.3	8	140445	4	131145	66.67	15.63
					Time(sec) : 10.8					
Depot 3										
12 [10]	66	7	69650	2.2	3	52450	3	64850	57.14	6.89
137 [18]	98	12	114880	3.1	8	101880	7	95780	41.67	16.63
551 [15]	53	8	91940	2.1	6	87240	4	84740	50.00	7.83
					Time(sec) : 16.3		Time(sec) : 128.4			
Total [125]	688	85	917852.5		58	832052.5	41	826852.5	51.76	9.91

Table 2. The schedule details

Scheduling Plan	KPI
Buses	41
Lines	9
Trips	688
Total Distance (Km)	15421.5
- Transfer KM	1340
- Trip KM	14081.5
Total Cost (Baht)	826852.5
- Vehicle	205000
- Line Change	43600
- Vehicle Transfer	38500
- Service KM	539752.5

view, the multi-depot with line change considering 688 trips in three depots; CLS provided a good quality solution in 128.4 seconds, which is practically acceptable.

Note that this is only the result with nine service lines. If a similar rate of improvement is applied to all 108 lines, the total cost reduction can be around 33 million Baht/month. The scheduling plan can also be assessed by the KPI (key performance indices) as shown in Table 2. The tests show that the proposed solution can potentially improve the bus fleet scheduling operation in this particular case.

6 Conclusions

The paper considered the bus fleet scheduling problem with multi-depot and line change operations, and proposed the constrained local search algorithm to solve the problem. The computational experiments were performed to evaluate the performance of the proposed method. The data sets from the Bangkok mass transit authority were used for the case study. By scheduling nine bus lines with 688 scheduled trips, the results indicated the potential operating cost savings by around 9.91 percent compared to the current schedule (manually prepare). The future research will attempt to apply the real-time information obtained from the automatic vehicle location system to generate more reliable and robust feet schedules. In addition, the integrated bus and crew schedule with improved algorithm will also be considered.

Acknowledgments. This research has been supported in part by a grant from the National Electronics and Computer Technology Center (NECTEC), Grant No. 09/2550 NT-B-22-IT-26-50-09). The authors are grateful to the BMTA for help and support.

References

1. Bertossi, A.A., Carraresi, P., Gallo, G.: On some matching problems arising in vehicle scheduling models. Networks 17, 271–281 (1987)
2. Ribeiro, C.C., Soumis, F.: A column generation approach to the multiple-depot vehicle scheduling problem. Operations Research 42, 41–52 (1994)

3. Lobel, A.: Optimal vehicle scheduling in public transit. PhD thesis, Technische University, Berlin (1997)
4. Lobel, A.: Vehicle scheduling in public transit and Lagrangian pricing. Management Science 4, 1637–1649 (1998)
5. Hadjar, A., Marcotte, O., Soumis, F.: A branch and cut algorithm for the multiple depot vehicle scheduling problem. Operations Research 54, 130–149 (2006)
6. Pepin, A.S., Desaulniers, G., Hertz, A., Huisman, D.: Comparison of heuristic approaches for the multiple depot vehicle scheduling problem. Report EI2006-34, Econometric Institute, Erasmus University Rotterdam (2006)
7. Kliewer, N., Mellouli, T., Suhl, L.: A time-space network based exact optimization model for multi-depot bus scheduling. European Journal of Operational Research 175(3), 1616–1627 (2006)
8. Indra-Payoong, N., Kwan, R.S.K., Proll, L.: Rail Container Service Planning: A Constraint-based Approach. In: Kendall, G., Petrovic, S., Burke, E., Gendreau, M. (eds.) Multidisciplinary Scheduling: Theory and Applications, pp. 343–365. Springer, Heidelberg (2005)