A Customized Tabu Search for the Vehicle Routing Problem with Simultaneous Pick-up and Delivery

Thanchuda Phannikul and Sombat Sindhuchao
Department of Industrial Engineering
Ubonratchathani University
E-mail: thanchuda@hotmail.com

Abstract

The Vehicle Routing Problem with Simultaneous Pick-up and Delivery (VPSPD) is classified as one of the most complicated and challenging Routing Problems. Having both pick-up and delivery loads at the same customer makes total loads uncertain on the vehicles along all their service routes. This paper presents an implementation of the Savings method for finding initial solutions of 72 benchmark instances. After that, an adaptive metaheuristic, Customized Tabu Search was applied in order to improve those initial solutions. The unique aspects of this search are the acceptance of the move which is tabu with some 'allowance' value and the extension of the tabu tenure with some 'penalty' charges. The allowance and penalty values used in each problem are varied by the problem's size. The results indicate a number of new best known solutions found by this proposed method compared with previous research.

Keywords: Tabu Search; Pick-up and Delivery; Vehicle Routing Problem.

1. Introduction

Nowadays, manufacturing, trading, services, as well as tourism sectors play an important role in Thailand's economy. In a competitive market, these activities require a right quantity with a best quality at the lowest cost in a rapid time. All in all, transportation and logistics are the vital key success factors.

In this paper, the Vehicle Routing Problem with Simultaneous Pick-up and Delivery (VRPSPD) is studied. The unique characteristic of this problem is that there are together, pick-up and delivery demands at each point of customers. Pick-up demands must be collected from each customer and sent to a depot. In the meantime, delivery demands must be loaded from the depot and carried to each customer. A vehicle has a limited capacity. With the nature of the problem, a fluctuation of vehicle load will occur all the time. Fig.1 presents the example of VRPSPD.

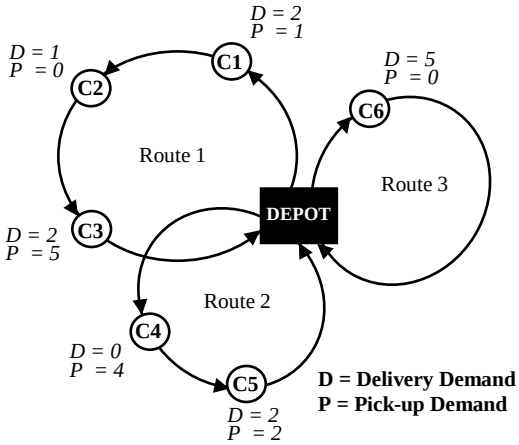


Fig. 1 Example of the Vehicle Routing Problem with Simultaneous Pick-up and Delivery (VRPSPD)

In a real situation, we can see many activities that can be applied with the VRPSPD model. The best practical sample is soft drink bottles distribution. Full bottles of soft drink must be supplied from a central depot, loaded on trucks in a sufficient quantity and delivered to grocery stores. After sending each order to each shop, empty bottles (in different quantities) must be collected and carried back to the depot.

However, every Vehicle Routing Problem including VRPSPD is categorized in NP-Hard combinatorial optimization problems. This means we cannot find the exact solution for medium and large scale instances in a reasonable time. Having only 15 customers in a problem can cause more than one thousand billion feasible solutions to evaluate. Heuristics and metaheuristics are always considered to handle these types of situations. The solution gained from heuristic methodology cannot be guaranteed for an optimal solution, but the solution's quality is good enough to use, considering the computation saving from solving for the exact solution. More importantly, in most real cases, solving for the exact solution is impossible, thus, using heuristics becomes the only feasible choice in practice.

In this article, the Savings method is a heuristic we propose to find initial solutions. After that, the modified Tabu Search called 'Customized Tabu Search' is developed to find final solutions. The rest of the paper is previous studies, mathematical modeling, an illustration of Savings and Customized Tabu Search in detail, four types of neighborhood search, comparison of our results with others and finally the summary of the research with a future study suggestions.

2. Literature Review

The first person who defined the VRPSPD problem is widely known as Min [1]. The author solved a 22-customer

problem by three-phase heuristics. Firstly, customers were clustered by the average linkage method. In the second phase, vehicles are assigned to these clusters and the Traveling Salesman Problem (TSP) algorithm was used to find routing solutions in the last phase. The initial phase of Halse [2] also began with clustering customers, but then 3-Opt and exchange moves are used to improve those routes.

Later, Dethloff [3] presented an insertion-based procedure applied from cheapest insertion but considered three metrics: travel distance, capacity and radial surcharge with four different criteria. Both real and randomly generated instances were tested on these methods.

In 1999, Salhi and Nagy [4] solved these type of problems by constructing routes from partial customers and then inserting the remaining customers into the existing routes. Then in 2005, they developed another heuristic by constructing initial solutions first and improving them with VRP Routine to become feasible solutions [5]. Solving VRPSPD with multiple depots and the maximum distance constraint has made their research very interesting. These two papers also proved that the VRPSPD can be extended for use with other types of VRP problem. Wassan and Nagy [6] studied the old sets of instances but they moved from their own heuristic to study a general metaheuristic, Reactive Tabu Search. In this article, the initial solutions were formed by the modified Sweep method, and then the searching mechanism of Tabu Search was implemented. The special characteristic of the Reactive Tabu Search is the changing of tenure (period of tabu-move) due to the repetition of answers found. The solution's quality of research was improved in many instances.

In the same year, Montane and Galvao [7] also used a Tabu Search with four types of neighborhood search: relocation, interchange, crossover and all those three movements combined. Two strategies of intensification and diversification are implemented together with frequency penalization. Chen and Wu [8] presented a hybrid heuristic based on record-to-record travel as well as an insertion-based procedure to solve VRPSPD. Chen and Chia [9] introduced a hybrid metaheuristic between Simulated Annealing (SA) and Tabu List to solve VRPSPD. The quick solutions are obtained by a parallel-insertion procedure before their hybrid search was used.

Bianchessi and Righini [10] solved the VRPSPD by merging arc-exchange-based with node-exchange-based in Tabu List construction. They also compared the results from Constructive Heuristics and Local Search with Tabu Search.

The most recent papers (published in 2009) about VRPSPD are Zachariadis et al. [11] and Ai and Kachitvichyanukul [12]. The first paper constructed the initial solutions by Cost Savings method then the combination between Tabu Search and Guide Local Search was applied to improve the results. The latter paper used another Metaheuristic which had not been implemented for this type of problem before Particle Swarm Optimization (PSO). This method is inspired by social behavior of bird flocking or fish schooling. A population of random solutions are searched and they are approved by following the current optimum particles. However, only a few of the results from PSO in this article are better than previous studies.

3. Mathematical Model

3.1 Problem Characteristics

The assumptions of problem are set as follows:

- All pick-up and delivery demands are deterministic.
- Deliveries must be supplied from depot only.

- No interchanges of goods between customers.
- No restricted pick-up and delivery orders.
- Delivery and pick-up demands cannot be split
- Each customer is visited only once.
- Number of available vehicles is sufficient.
- Every vehicle has equally limited capacity.
- No time window requirement.
- No maximum distance constraints.

3.2 Notations

i, j index for depot and customers (depot = 0, customer = 1, 2, ..., n)

k vehicle index

C_{ij} distance (cost) between customers i and j

n total number of customers

P_j pick-up demand of customer j where j = 1, 2, ..., n

D_j delivery demand of customer j where j = 1, 2, ..., n

Q vehicle capacity

TP_{ij} total pick-up demand from beginning to customer i and traveling in arc (i,j)

LD_{ij} demand left to deliver to customers after customer i and traveling in arc (i,j)

x_{ij}^k { 1, if vehicle k travels from customer i to j; 0, otherwise

3.3 Mathematical Formulation

$$\text{Minimize} \quad \sum_{k=1}^{m}\sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij} X_{ij}^{k} \qquad (1)$$

Subject to

$$\sum_{i=1}^{n}\sum_{j=1}^{n} D_j x_{ij}^k \le Q \qquad \forall k \qquad (2)$$

$$\sum_{i=1}^{n}\sum_{k=1}^{m} x_{ij}^k = 1 \qquad \forall j > 0 \qquad (3)$$

$$\sum_{i=1}^{n} x_{ij}^k - \sum_{i=1}^{n} x_{ji}^k = 0 \qquad \forall j,k \qquad (4)$$

$$\sum_{i=1}^{n} TP_{ji} - \sum_{i=0}^{n} TP_{ij} = P_j \qquad \forall j > 0 \qquad (5)$$

$$\sum_{i=1}^{n} LD_{ji} - \sum_{i=0}^{n} LD_{ij} = D_j \qquad \forall j > 0 \qquad (6)$$

$$TP_{ij} + LD_{ij} \le Q\sum_{k=1}^{m} x_{ij}^k \qquad \forall i, j \qquad (7)$$

$$x_{ij}^k \in \{0,1\} \qquad (8)$$

$$n, C_{ij}, P_j, D_j, Q, TP_{ij}, LD_{ij} \ge 0 \qquad (9)$$

The objective function (1) aims to minimize the total cost or total distances that occur in all routes. Constraints (2) ensure that all delivery demands in each route are not greater than vehicle capacity. Constraints (3) count the number of vehicles that serve each customer and guarantee that each customer is visited by only one vehicle (except depot). Constraints (4) ensure that the same vehicle arrives and departs from each customer while constraints (5) and (6) dictate the pick-up and delivery demand constraints, respectively. Constraints (7) verify that at each customer node, both types of demands must not be greater than the vehicle capacity. Constraints (8) define the possible members in the set of decision variables that is only 1 or 0 and the last constraints are the non-negativity constraints.

4. Initial Solution

In general, solving procedures of the Vehicle Routing Problem and other combinatorial problems are frequently divided into two phases: an initial phase and an improvement phase. The initial phase begins with finding a basic feasible solution from any heuristic method. Then, some iterated Local Search methods will be employed in order to find a better solution from the initial one in the improving step. A

proper starting solution certainly leads to a better final result. According to our study, we found that the Savings method outperforms other heuristics and gave the most appropriate initial solution. Thus, in this research, the Savings method is selected to be an initial solution constructive algorithm.

The Savings method was first introduced by Clarke and Wright in 1964. We apply this heuristic to our VRPSPD problem with the following steps.

Step 1 For each possible pair of customers, calculate the savings value between customers *i* and *j* (S_{ij}) from the distance saved if they are combined in the same route instead of visited by a vehicle separately. The savings can be computed using Equation (10); where C_{io} represents distance from customer *i* to the depot, C_{oj} represents distance from the depot to customer *j* and C_{ij} is the distance from customer *i* to customer *j*.

$$S_{ij} = C_{io} + C_{oj} - C_{ij} \qquad (10)$$

Step 2 Arrange customers in a decreasing order of savings.

Step 3 Choose the highest savings and then consider if both customers can be possibly linked together. If so, join them in the same route. If not, discard them and select the next savings.

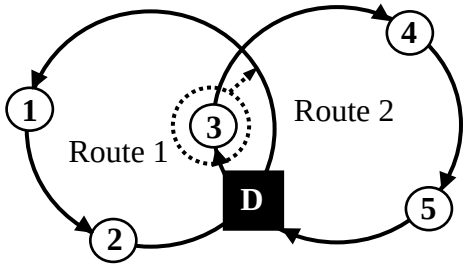
Step 4 Continue with step 3 until no more pairs of customers can be combined with positive savings.

5. Neighborhood Search

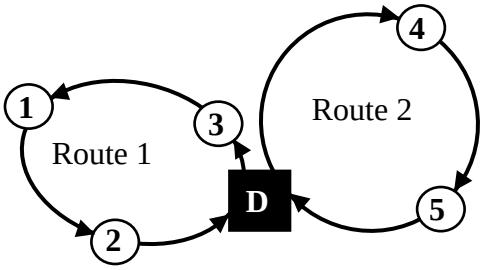
After an initial solution is obtained from the Savings method, the improvement phase will be implemented. The concept of route improvement comes from customer exchange, both intra-route and inter-route. Moving a customer from one position to another creates many possible alternative routes. The number and the order of neighborhood structures have direct effect on search space. Too few structures would result in less exploration, whereas too many structures applied can cause too large search areas, and result in uncontrollable solutions. Our Customized Tabu Search applies four types of neighborhood search (given below).

5.1 Insertion move

An insertion move begins by selecting a customer, and then moving it to a new position. This move considers all possible positions, within current and new routes, that each customer can be relocated to. Please see Fig. 2.



Insert C3 into route 1 before C1



After Move

Fig. 2 Example of Insertion move

5.2 Two-opt move

This type of move considers only intra-route moving. In this improvement method, two non-adjacent arcs are selected and then swapped, if this exchange results

in a better solution. All exchanges of two edges are tested until no better solution can be obtained. Please note that, a Two-opt move can be applied only with a route that has more than two customers. Fig. 3 presents the example of this move.

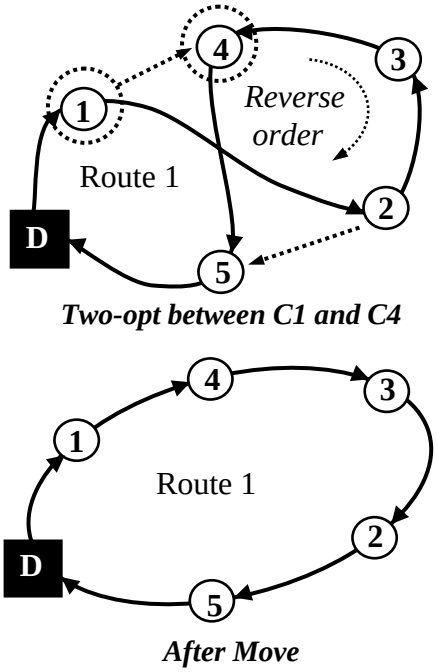


Fig. 3 Example of Two-opt move

5.3 Crossover move

An example of a Crossover move shown in Fig.4. All customers after the selected customers will be swapped. In other words, two arcs from two routes are removed and two new ones are connected. This allows the initial section of the first route to connect to the final section of the second route and vice versa.

5.4 Reverse Move

A reverse move is similar to a crossover move in that two arcs from two routes are removed and two new ones are connected, but in a different way. This type of move will connect both initial sections of the first and the second routes together. Consequently, the final parts of those two routes will be joined. The reversal of the customer order occurs within the second route. Please see Fig. 5 for more understanding.

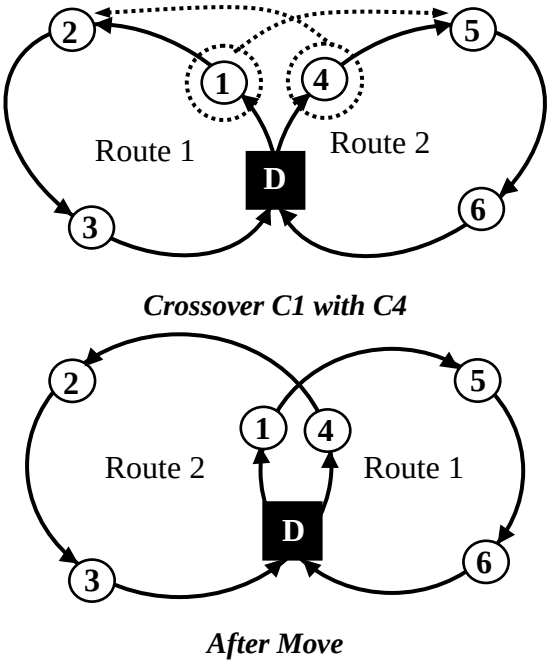


Fig. 4 Example of Crossover move

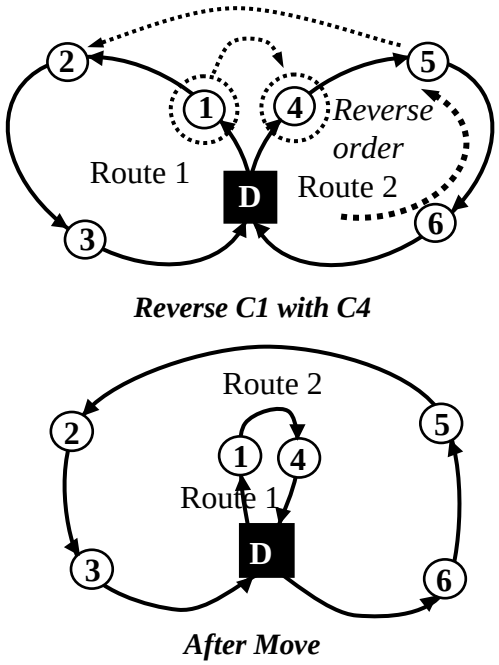


Fig. 5 Example of Reverse Move

The four types of moves stated above are done with the Best Improvement principal. During these moves, if any move results in an infeasible solution (excess vehicle capacity), that solution is neglected and the next move is considered instead. Only feasible solutions are recorded. All possible movements are explored before the best feasible one that gives the lowest cost is selected.

6. Customized Tabu Search

According to the article of Gendreau [13], Tabu Search is attributed to Fred Glover (1986), while some researchers assumed that this methodology is from the publication named 'Tabu Search' by Glover and Laguna in 1997. However, the main rule of Tabu Search is the memory structures used to record determined solutions, that will be marked in the 'Tabu' or 'Taboo' list, to prevent cycling phenomena.

By using the same concept as the standard Tabu Search, we modify our own technique called 'Customized Tabu Search' to solve VRPSPD. Its name is taken from its characteristic in controlling tenure with Penalty and Allowance values, which are set by the number of customers in each problem. Consequently, the algorithm can fit problems of any size.

After conducting some tests, we prefer to set all tabu tenure to begin at 50. If the selected move in any iteration is repeatedly visited (had been recorded in tabu list), the tenure number will be multiplied by a Penalty value. After trials, we discovered that the ratio of 1/25 or four percent of the number of customers gave preferable results.

Allowance value is a special feature of our Customized Tabu Search. In our approach, any move that has a tenure period less than an allowable bound (set as 1/5 or 20 percent of the number of customers) can be selected if it is the best answer found in that iteration. This Allowance value gives an opportunity for a good answer to be selected, even it is still in the tabu period. However, this permission differs from reduction of the tenure with allowance value due to the penalty process. For example, setting Tenure at 50, Penalty at 2, and Allowance at 10, if the move re-occurs, the tenure after penalty will be 100, but if we decrease tenure with allowance (tenure = 50-10 = 40), the tenure after penalty will be 80. This causes different final solutions.

In order to compare the results, we decided to use the same stopping criteria from the latest best known paper (Zachariadis et al. [11]). The searching process will be terminated when
6,000 iterations are reached without any improvement in the objective function.

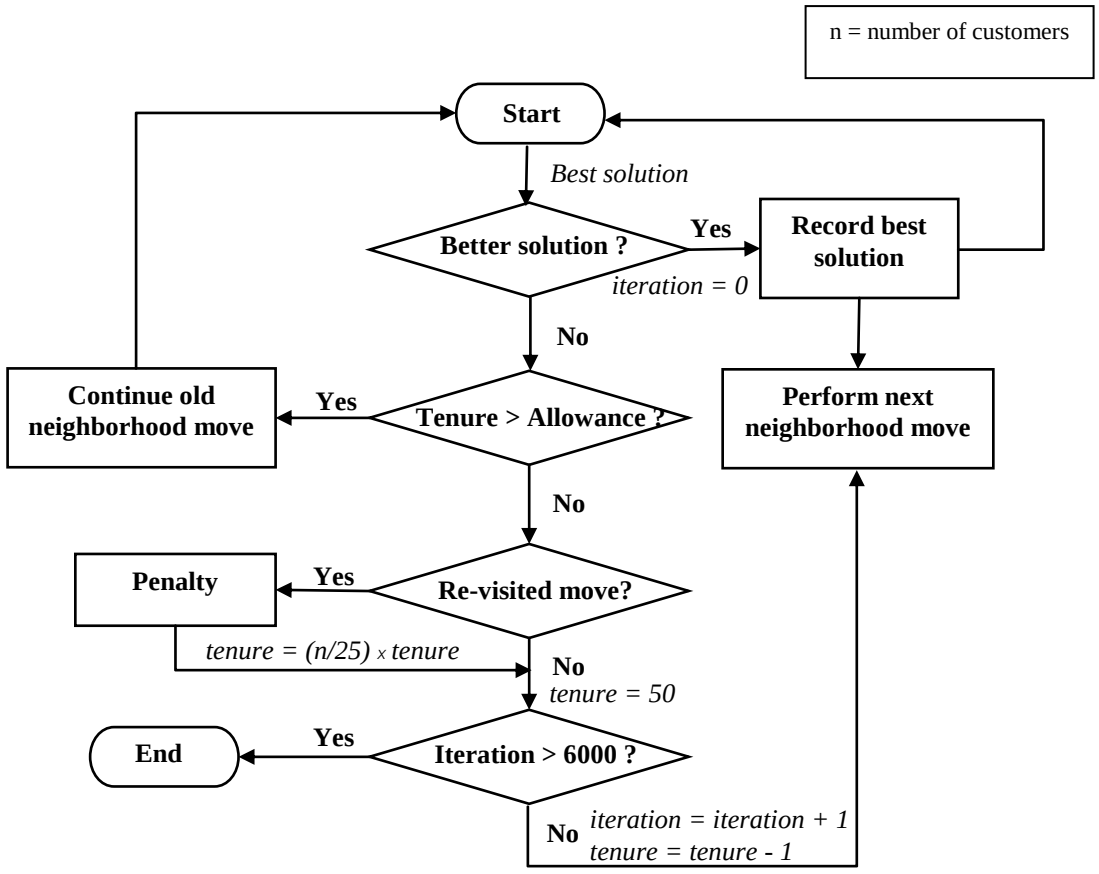


Fig. 6 The working procedure of the Customized Tabu Search used after each neighborhood.

Fig. 6 explains the working procedure of the Customized Tabu Search. An initial solution obtained from the Savings method is searched with the first neighborhood move (e.g. Two-opt move). If the best answer found from this move is better than other best solutions, it will be recorded. This answer is then used to be an initial solution to perform the next move (e.g. Crossover Move). But if the best answer found is not better and its Tenure is greater than the Allowance value (being in tabu period), the next best answer from the first neighborhood move is searched after that. In case the Tenure is less than Allowance, this solution is accepted, Tenure is given and recorded in Tabu list. If the answer is selected before, the tenure is extended longer by penalty process. The iteration and tenure period are counted at this point. Because the tenure is counted after each neighborhood move, totalling four types of move, tenure is set at a higher value when compared with usual Tabu Search.

This Customized Tabu Search approach is applied after each type of neighborhood move until 6,000 iterations are reached without any improvement.

7. Computational Results

In order to test the performance of our proposed algorithm, we use VRPSPD instances from three papers. There are 14 problems from Salhi and Nagy [4], 18

problems from Montane and Galvao [7] and 40 problems from Dethloff [3].

A comparison is made between our solutions and those from previous studies. The best known solutions in each dataset are marked with bold characters.

The implementation of the Savings method and the Customized Tabu Search are done in Visual C++ (version 6.0). All 72 results are reported in the following tables.

Table 1 presents the computational results of Salhi & Nagy's problems. This dataset is one of the most popular benchmarks among VRPSPD instances. Although there are no new best known solutions found by our method, we can observe that Customized Tabu Search is competitive with Zachariadis[11] and better than both Montane & Galvao[7] and Ai & Kachitvichyanukul [12] on average.

Table 1 Computational Results of Salhi & Nagy's Problem

Problem	No.of Cust.	Montane & Galvao (2006)			Zachariadis et al. (2009)			Ai & Kachitvichyanukul (2009)			Customized Tabu Search		
		No. of Vhcls	Total Dist.	CPU Time[a]	No. of Vhcls	Total Dist.	CPU time[b]	No. of Vhcls	Total Dist.	CPU time[c]	No. of Vhcls	Total Dist.	CPU Time[d]
CMT1X	50	3	472	3.73	3	470.48	4.06	3	**467**	-	3	470.48	2.08
CMT1Y	50	3	470	4.37	3	470.48	3.21	3	**467**	-	3	470.48	86.13
CMT2X	75	7	695	6.91	6	**682.39**	6.53	6	710	-	6	690.31	144.52
CMT2Y	75	7	700	7.61	6	**682.39**	7.94	6	710	-	6	686.86	42.63
CMT3X	100	5	721	11.04	5	**719.06**	10.52	5	738	-	5	719.90	1229.31
CMT3Y	100	5	**719**	12.01	5	719.06	13.25	5	740	-	5	723.67	771.66
CMT12X	100	6	675	12.23	5	**658.83**	12.04	5	691	-	5	659.82	1423.34
CMT12Y	100	6	689	12.80	5	**660.47**	10.43	5	697	-	5	667.12	192.47
CMT11X	120	4	900	18.17	4	**831.09**	16.82	4	895	-	4	846.94	1.67
CMT11Y	120	5	910	18.04	4	**829.85**	15.26	4	900	-	4	849.89	563.75
CMT4X	150	7	880	24.60	7	**854.21**	22.98	7	912	-	7	859.78	1755.11
CMT4Y	150	7	878	29.07	7	**852.46**	28.65	7	913	-	7	860.68	981.05
CMT5X	199	11	1098	51.50	10	**1030.56**	57.62	10	1167	-	10	1039.77	5277.00
CMT5Y	199	10	1083	56.21	10	**1031.69**	53.80	10	1142	-	10	1038.11	2353.97
Average			777.86			749.50			796.36			755.99	

[a] CPU Seconds in Athlon 2.0 GHz and 256 MB of RAM.
[b] CPU Seconds in Pentium IV 2.4 GHz and 1 GB of RAM.
[c] No available data
[d] CPU Seconds in Intel Centrino Duo 1.66 GHz (Laptop) and 1 GB of RAM.

Results shown in Table 2 indicate that Customized Tabu Search can find the best known solution of as many as 32 problems from 40 of Dethloff's problems. Among these, there are five answers that are the new best solutions; CON3-4, CON3-6, CON3-9, CON8-2 and CON8-5. The CPU time is difficult to compare because of the difference between computer specifications and stopping criteria. However, our proposed algorithm can find better or equivalent results with less time used in many problems.

Table 2 Computational Results of Dethloff's Problem

Problem	No.of Cust.	Dethloff (2001)			Montane & Galvao (2006)			Zachariadis et al. (2009)			Customized Tabu Search		
		No. of Vhcls	Total Dist.	CPU Time[a]	No. of Vhcls	Total Dist.	CPU Time[b]	No. of Vhcls	Total Dist.	CPU Time[c]	No. of Vhcls	Total Dist.	CPU Time[d]
SCA3-0	50	-	689.00	-	4	640.55	3.37	4	**636.06**	2.83	4	**636.06**	7.27
SCA3-1	50	-	765.60	-	4	**697.84**	3.25	4	**697.84**	2.12	4	**697.84**	0.48
SCA3-2	50	-	742.80	-	4	**659.34**	3.52	4	**659.34**	2.58	4	**659.34**	1.81
SCA3-3	50	-	737.20	-	4	**680.04**	3.31	4	**680.04**	3.13	4	**680.04**	18.09
SCA3-4	50	-	747.10	-	4	**690.50**	3.43	4	**690.50**	2.68	4	**690.50**	0.86
SCA3-5	50	-	784.40	-	4	**659.90**	3.67	4	**659.90**	2.56	4	**659.90**	0.55
SCA3-6	50	-	720.40	-	4	653.81	3.35	4	**651.09**	4.40	4	**651.09**	4.45
SCA3-7	50	-	707.90	-	4	**659.17**	3.33	4	**659.17**	2.98	4	666.60	34.09
SCA3-8	50	-	807.20	-	4	**719.47**	3.40	4	**719.47**	3.38	4	**719.47**	0.11
SCA3-9	50	-	764.10	-	4	**681.00**	3.41	4	**681.00**	3.86	4	**681.00**	4.50
SCA8-0	50	-	1132.90	-	9	981.47	4.14	9	**961.50**	3.21	9	**961.50**	13.55
SCA8-1	50	-	1150.90	-	9	1077.44	4.27	9	**1050.20**	3.55	9	1051.56	36.56
SCA8-2	50	-	1100.80	-	10	1050.98	4.20	9	**1039.64**	4.67	9	**1039.64**	82.95
SCA8-3	50	-	1115.60	-	9	**983.34**	4.17	9	**983.34**	3.29	9	**983.34**	0.14
SCA8-4	50	-	1235.40	-	9	1073.46	4.13	9	**1065.49**	2.68	9	**1065.49**	4.27
SCA8-5	50	-	1231.60	-	9	1047.24	4.02	9	**1027.08**	4.50	9	1027.12	36.86
SCA8-6	50	-	1062.50	-	9	995.59	3.85	9	**971.82**	2.67	9	**971.82**	22.84
SCA8-7	50	-	1217.40	-	10	1068.56	4.22	9	**1052.17**	4.32	9	1062.30	37.47
SCA8-8	50	-	1231.60	-	9	1080.58	3.85	9	**1071.18**	3.43	9	1082.12	4.25
SCA8-9	50	-	1185.60	-	9	1084.80	4.20	9	**1060.50**	4.12	9	**1060.50**	117.61
CON3-0	50	-	672.40	-	4	631.39	3.64	4	**616.52**	3.89	4	**616.52**	5.95
CON3-1	50	-	570.60	-	4	**554.47**	3.31	4	**554.47**	2.97	4	**554.47**	12.88
CON3-2	50	-	534.80	-	4	522.86	3.45	4	**519.26**	3.32	4	523.16	141.34
CON3-3	50	-	656.90	-	4	**591.19**	3.28	4	**591.19**	2.78	4	**591.19**	6.89
CON3-4	50	-	640.20	-	4	591.12	3.47	4	589.32	3.12	4	**588.79**	59.92
CON3-5	50	-	604.70	-	4	**563.70**	3.38	4	**563.70**	3.45	4	**563.70**	1.97
CON3-6	50	-	521.30	-	4	506.19	3.32	4	500.80	2.98	4	**499.05**	19.33
CON3-7	50	-	602.80	-	4	577.68	3.51	4	**576.48**	2.40	4	**576.48**	45.83
CON3-8	50	-	556.20	-	4	**523.05**	3.66	4	**523.05**	5.02	4	**523.05**	3.80
CON3-9	50	-	612.80	-	4	580.05	3.36	4	580.05	3.14	4	**578.25**	33.08
CON8-0	50	-	967.30	-	9	860.48	4.19	9	**857.17**	3.40	9	857.40	13.39
CON8-1	50	-	828.70	-	9	**740.85**	3.89	9	**740.85**	3.73	9	**740.85**	34.45
CON8-2	50	-	770.20	-	9	723.32	3.76	9	713.44	2.87	9	**712.89**	1.09
CON8-3	50	-	906.70	-	10	811.23	4.12	10	**811.07**	3.82	10	**811.07**	2.20
CON8-4	50	-	876.80	-	9	**772.25**	3.75	9	**772.25**	2.98	9	**772.25**	7.28
CON8-5	50	-	866.90	-	9	756.91	3.99	9	756.91	5.76	9	**754.88**	1.77
CON8-6	50	-	749.10	-	9	**678.92**	4.04	9	**678.92**	4.00	9	**678.92**	9.34
CON8-7	50	-	929.80	-	9	814.50	4.00	9	**811.96**	2.46	9	812.70	9.83
CON8-8	50	-	833.10	-	9	775.59	3.74	9	**767.53**	4.21	9	**767.53**	3.34
CON8-9	50	-	877.30	-	9	809.00	4.13	9	**809.00**	3.87	9	**809.00**	0.53
Average			842.72			764.25			758.78			759.48	

[a] No available data
[b] CPU Seconds in Athlon 2.0 GHz and 256 MB of RAM.
[c] CPU Seconds in Pentium IV 2.4 GHz and 1 GB of RAM.
[d] CPU Seconds in Intel Centrino Duo 1.66 GHz (Laptop) and 1 GB of RAM.

Table 3 Computational Results of Montane & Galvao's Problem

Problem	No.of Cust.	Montane and Galvao (2006)			Zachariadis et al. (2009)			Customized Tabu Search		
		No. of Vhcls	Total Dist.	CPU Time[a]	No. of Vhcls	Total Dist.	CPU Time[b]	No. of Vhcls	Total Dist.	CPU Time[c]
r101	100	12	1042.62	13.20	12	1019.48	10.50	12	**1018.17**	81.78
r201	100	3	671.03	12.02	3	**666.2**	8.70	3	666.67	1059.47
c101	100	17	1259.79	12.07	16	**1220.99**	10.20	16	1235.26	22.63
c201	100	5	666.01	12.40	5	**662.07**	5.70	5	**662.07**	1.39
rc101	100	11	1094.15	12.30	10	**1059.32**	12.90	10	1074.50	126.80
rc201	100	3	674.46	12.07	3	**672.92**	10.50	3	**672.92**	669.72
r1_2_1	200	23	3447.2	55.56	23	**3393.31**	61.80	23	3408.82	2049.30
r2_2_1	200	5	1690.67	50.95	5	**1673.65**	47.40	5	1685.40	6337.34
c1_2_1	200	29	3792.62	52.21	28	**3652.76**	66.30	28	3663.41	1542.97
c2_2_1	200	9	1767.58	65.79	9	**1735.68**	60.90	9	1762.87	2442.53
rc1_2_1	200	24	3427.19	58.39	23	**3341.25**	45.30	23	3385.39	676.19
rc2_2_1	200	5	1645.94	52.93	5	**1562.34**	62.40	5	1576.05	5307.84
r1_4_1	400	54	10027.81	330.42	54	**9758.77**	315.30	54	9793.60	15979.30
r2_4_1	400	10	3695.26	324.44	10	**3606.72**	273.60	10	3635.98	3749.77
c1_4_1	400	65	11676.27	287.12	63	11207.37	283.50	63	**11194.30**	32551.10
c2_4_1	400	15	3732	330.20	15	3630.72	336.00	15	**3626.57**	22508.00
rc1_4_1	400	52	9883.31	286.66	52	9697.65	145.80	52	**9682.67**	18220.30
rc2_4_1	400	11	3603.53	328.16	11	**3498.3**	345.00	11	3578.05	21752.80
Average			3544.30			3447.75			3462.37	

[a] CPU Seconds in Athlon 2.0 GHz and 256 MB of RAM.
[b] CPU Seconds in Pentium IV 2.4 GHz and 1 GB of RAM.
[c] CPU Seconds in Intel Centrino Duo 1.66 GHz (Laptop) and 1 GB of RAM.

Table 3 shows the results of Montane & Galvao's problems. Six of best known solutions are found by using of our Customized Tabu Search and four of them are the new best results; r101, c1_4_1, c2_4_1 and rc1_4_1. The higher computational time may come from the stopping conditions, the compiler program, and our unoptimized programming.

8. Conclusion

The Vehicle Routing Problem with Simultaneous Pick-up and Delivery or VRPSPD can be implemented to solve many situations in real life. The special aspect of VRPSPD is that there are both pick-up and delivery demands at each customer. It is nearly impossible to find the optimal solution for the medium and large-sized problems within a reasonable time. Heuristics and metaheuristics are developed to handle this type of problem instead of an exact method.

In this article, we proposed our modified metaheuristic, namely Customized Tabu Search to solve VRPSPD. Firstly, the initial solutions are constructed by the Savings method. Then four types of movement; Two-opt, Crossover, Insertion and Reverse moves are used to search neighborhood space of the beginning solution. The Customized Tabu Search differs from the standard Tabu Search in the sense of Tenure, Penalty, and Allowance values, that will be set to fit the problem size. The solutions are changed from iteration to iteration but the intensification and diversification are controlled by proper parameters during the search.

After computational results of testing 72 problem instances it can be concluded that the proposed algorithm is

competitive with existing benchmarks. Many of the new best known solutions are found by using this method. Moreover, some of them are obtained with less CPU time used.

Further study should be extended to consider more constraints on maximum distance or time window requirements. Other heuristics or metaheuristics can also be investigated with VRPSPD.

9. Acknowledgments

This research is supported by the Commission on Higher Education, Ministry of Education, Thailand. In addition, the authors wish to say thanks to Professor Fermin Alfredo Tang Montané for his inspiring paper and his kindness in providing us with datasets used in this paper.

10. References

[1] Min, H., The Multiple Vehicle Routing Problem with Simultaneous Delivery and Pickup Points, Transportation Research, Vol. 23A, pp. 377- 386, 1989.

[2] Halse, K., Modeling and Solving Complex Vehicle Routing Problems, Ph.D. Thesis. Denmark: Technical University of Denmark, 1992.

[3] Dethloff, J.,Vehicle Routing and Reverse Logistics: The Vehicle Routing Problem with Simultaneous Delivery and Pick-up, OR Spektrum, Vol. 23, pp.79-96, 2001.

[4] Salhi, S. and Nagy, G., A Cluster Insertion Heuristic for Single and Multiple Depot Vehicle Routing Problems with Backhauling, Journal of the Operational Research Society, Vol. 50, pp. 1034-1042, 1999.

[5] Salhi, S and Nagy, G., Heuristic Algorithms for Single and Multiple Depot Vehicle Routing Problems with Pickups and Deliveries, European Journal of Operational Research, Vol. 162, pp. 126-141, 2005.

[6] Wassan, N. and Nagy, G.,A Tabu Search Algorithm for the Vehicle Routing Problem with Deliveries and Pickups, in Proceedings of the International Conference on Software Knowledge Information Management and Application. Chiangmai: SKI MA, 2006.

[7] Montane, F.A. and Galvao, R. D., A Tabu Search Algorithm for the Vehicle Routing Problem with Simultaneous Pick-up and Delivery Service, Computer & Operations Research, Vol. 33, pp. 595-619, 2006.

[8] Chen, J. F. and Wu, T. H., Vehicle Routing Problem with Simultaneous Deliveries and Pickups, Journal of the Operational Research Society, Vol. 57, pp. 579-587, 2006.

[9] Chen, J. F. and Chia, F., Approaches for the Vehicle Routing Problem with Simultaneous Deliveries and Pickups, Journal of the Chinese Institute of Industrial Engineers, Vol. 23, pp. 141-150, 2006.

[10] Bianchessi, N. and Righini, G., Heuristic Algorithms for the Vehicle Routing Problem with Simultaneous Pick-up and Delivery, Computer & Operations Research, Vol. 34, pp. 578-594, 2007.

[11] Zachariadis, E. E. and et al., A Hybrid Metaheuristic Algorithm for the Vehicle Routing Problem with Simultaneous Delivery and Pick-up Service, Expert Systems with Applications, Vol. 36, pp. 1070-1081, 2009.

[12] Ai, T. J. and Kachitvichyanukul, V., A Particle Swarm Optimization for the Vehicle Routing Problem with Simultaneous Pickup and Delivery, Computer & Operations Research, Vol. 36, pp. 1693-1702, 2009.

[13] Gendreau, M., An Introduction to Tabu Search, Universitetet I Oslo. Available on. http://www.ifi.uio.no/infheur/Bakgrunn/Intro_ to_TS_Gendreau.htm., October 17, 2008.