An International Journal of Optimization and Control: Theories & Applications
ISSN: 2146-0957 eISSN: 2146-5703
Vol.16, No.3, pp.981-996 (2026)
https://doi.org/10.36922/IJOCTA025480215

RESEARCH ARTICLE

A two-phase branch-and-bound algorithm for solving vehicle routing problem with maximum duration constraint

Asri Bekti Pratiwi[1,2], Salmah Salmah[1*], and Irwan Endrayanto[1]

[1]Department of Mathematics, Faculty of Mathematics and Natural Sciences, Universitas Gadjah Mada, Yogyakarta, Indonesia
[2]Department of Mathematics, Faculty of Science and Technology, Universitas Airlangga, Surabaya, East Java, Indonesia
asribektipratiwi@mail.ugm.ac.id, syalmah@yahoo.com, endrayanto@ugm.ac.id, salmah@ugm.ac.id

ARTICLE INFO

Article History:
Received: November 27, 2025
Revised: February 12, 2026
Accepted: February 24, 2026
Published Online: April 30, 2026

Keywords:
Branch and bound
Dynamic programming
Heuristic algorithm
Vehicle routing problem
Maximum duration constraint

AMS Classification 2010:
90B10; 97M40; 90C27; 90C59
65K05

ABSTRACT

Branch-and-bound (B&B) methods are widely recognized for their ability to provide exact solutions to combinatorial optimization problems, including routing models. This study presents a Two-phase B&B algorithm developed to address the Vehicle routing problem (VRP) under maximum duration constraints, offering an effective framework for obtaining a cost-optimal routing solution. The first phase of the algorithm employs a heuristic to construct a tour that satisfies the classical VRP constraints. In the second phase, a recursive dynamic programming procedure is applied to partition the obtained tour into a set of feasible subtours that meet the maximum allowable duration per vehicle. This dual-phase approach addresses the computational challenges of traditional B&B methods by decoupling tour optimization from constraint satisfaction. Computational experiments were conducted on benchmark VRP datasets and compared with exact methods and heuristic algorithms. The computational results demonstrate that the proposed algorithm consistently outperforms in terms of solution quality, number of explored nodes, and computation time. These findings demonstrate the effectiveness of the algorithm in obtaining optimal solutions for VRP with maximum duration constraints.

1. Introduction

The branch and bound (B&B) algorithm is a commonly employed method for solving optimization problems to find optimal solutions. It works by iteratively partitioning the solution space into smaller regions, a procedure known as branching, and systematically eliminating areas that cannot yield an optimal solution, known as pruning or bounding. Through the application of bounds, the algorithm reduces the scope of the search, thereby expediting the discovery of optimal solutions. Essentially, bounding allows the algorithm to focus only on promising areas of the solution space while disregarding those that are unproductive or irrelevant. After exploring the entire search space, the algorithm considers the best identified solution as optimal.[1,2] The B&B algorithm structures its search process in the form of a decision tree. Due to its simplicity, the algorithm can be adapted or enhanced to improve performance and computational efficiency. A faster identification of the optimal solution typically results in a smaller search tree, which reflects the algorithm's effectiveness in navigating the solution space.

According to, Morrison *et al.*[1] the B&B algorithm consists of three fundamental strategies that can be adapted to suit the nature of the

*Corresponding Author

problem: the search strategy, the branching strategy, and the pruning strategy. In the context of routing-related research, specifically for the Vehicle routing problem (VRP), studies commonly employ two main search strategies: depth-first search and best-first search. Studies such as those by Theurich *et al.*[3], Duman *et al.*[4] and Ozbaygin *et al.*[5] have implemented depth-first search to address various VRP variants. Due to its tree-based nature, depth-first search typically requires extensive exploration time, as it progresses vertically through sub-problems without leveraging model-specific information, such as objective function values or constraint evaluations. This strategy focuses on fully exploring one branch of the search tree before backtracking to others. In contrast, Li *et al.*[6] applied the best-first search strategy in solving the split delivery VRP, emphasizing pruning techniques to enhance performance. However, as noted in, Morrison *et al.*[1] best-first search may incur delays in mid-level exploration and can sometimes overlook branches that contain the optimal solution.

Branching strategies in the B&B algorithm are generally categorized into two types: binary branching and wide branching. In binary branching, each decision node gives rise to two mutually exclusive branches for obtaining exact solutions. Several studies[4,7–9] have implemented binary branching due to its straightforward logic and compatibility with decision variables in routing problems. However, as noted by, Morrison *et al.*[1] binary branching can become computationally demanding, as the number of branches increases significantly and leads to more complex sub-problems. On the other hand, wide branching[3,10] employs multiple branch directions within a single step. In this approach, two branching mechanisms were combined: one extends a route by adding a task at the end, while the other inserts a task into an existing route. Although this approach may enrich the search space, it can also lead to redundant exploration of similar sub-problems, potentially increasing the computational burden through repeated or overlapping searches.

Pruning rules are one of the components of the B&B algorithm that enhance its performance. Several studies have applied additional methods this this pruning strategy. For example, research by Theurich *et al.*[3], Laporte *et al.*[7] and Fischetti *et al.*[11] used a lower bound as a pruning technique. This technique is simple because it does not require high complexity, but it can be time-consuming in terms of computation. The branching strategy is the main focus of research by, Theurich *et al.*[3] while research by Duman *et al.*[4] and Ozbaygin *et al.*[5] more favored its pruning strategy in completing the routing problem. The adaptability of the B&B framework makes it suitable for solving complex combinatorial problems such as the VRP.

Vehicle routing problem was introduced by Dantzig and Ramser in 1959, discussing the optimal route for gasoline delivery vehicles in serving a large number of service stations provided by a depot.[12] VRP extends the classical Traveling salesman problem (TSP) by involving multiple vehicles and additional constraints, such as capacity, time windows, and route duration limits.[13,14] In addition, in line with the development of the VRP model itself, most VRP models with additional constraint are expressed in the form of mixed-integer linear programming.[15–17] These added complexities make exact solution methods, such as B&B, particularly valuable. Several studies have developed exact B&B approaches to solve VRP with corresponding constraints in real-world cases, including: branch-and-cut,[8] branch-and-price,[18] and branch-price-and-cut.[19,20] However, these exact methods are sometimes highly sensitive to the model formulation, their implementation is often complex, and computation time is unpredictable.

Several researchers employ non-exact approaches such as heuristics and metaheuristics [21], as demonstrated by[22] and[23]. Several studies have used heuristic approaches to find feasible solutions for the given mixed-integer linear programming.[24–26] Meanwhile, the study conducted by[22] applied an improved tabu search (TS) to solve the VRP with specific constraints, whereas[23] applied adaptive learning neighborhood search to address a VRP with different constraints. Although heuristic-based algorithms can reduce computational effort, they may fail to guarantee the discovery of an optimal solution.

However, solving the VRP using standard B&B can be computationally intensive due to the enlarged solution space and the presence of multiple feasible routes. The B&B algorithm applied to a TSP model derived from the assignment problem can produce feasible solutions according to the mathematical formulation, but these solutions may be invalid in the context of the real problem. This occurs because the method may generate subtours, which are not valid solutions for TSP. To address this issue, subtour elimination constraints must

be added to the model, which typically introduce additional continuous decision variables.[2,27]

In contrast, the VRP allows the presence of subtours, where each subtour represents the route of a vehicle starting and ending at the depot. The maximum duration constraint may necessitate the use of more than one vehicle, as each vehicle has a limited operational duration. Moreover, the model does not include a decision variable representing the number of vehicles; therefore, it is generally assumed that the number of available vehicles is limited but sufficient. As a result, when traditional B&B is used, repeated iterations are required to obtain the optimal solution. An alternative scheme, such as introducing new decision variables into the original model, often increases model complexity. Therefore, additional modifications to the B&B algorithm are required, potentially by incorporating heuristic components to guide the solution process more efficiently.

Our research is a preliminary study on combining a heuristic approach with the B&B algorithm to solve a specific VRP model. To the best of our knowledge, no previous study has investigated solution search for a specific VRP model using a heuristic approach within a B&B framework. We begin with a VRP model that incorporates an additional constraint in the form of a maximum allowable duration for each vehicle. This constraint is determined not only by the vehicle's travel time but also by the deterministic service time required at each customer.

This study introduces a Two-phase B&B algorithm that incorporates a heuristic approach and a recursive step of dynamic programming procedure within B&B framework to obtain optimal solution for VRP with a maximum duration constraint. This work focuses on a specific variant of the VRP, namely addressing the limitation imposed by the maximum operational duration of each vehicle. Given the limitations of heuristic algorithms, which may fail to guarantee optimality, this study adopts a combinatorial optimization approach to efficiently identify optimal solutions, whereby the VRP can be formally defined using graph theory.

The proposed B&B algorithm is modified into two phases: the first phase employs a heuristic approach, and the second phase utilizes dynamic programming. As both stages operate within B&B framework, the first phase employs a best-first search strategy with complete branching. This strategy provides enhanced efficiency and is suitable for the Two-phase B&B algorithm. It begins with wide branching to all unexplored candidate solutions using a heuristic approach until a complete tour is obtained. These strategies are intended to eliminate the possibility of repeatedly exploring the same solution space. In the second phase, dynamic programming with a recursive procedure is applied to determine an optimal set of subtours that satisfies the maximum duration constraint. This two-stage process is executed at each iteration and terminates when all branches have been explored, resulting in an optimal solution.

This two-stage combination of the heuristic and the B&B algorithm change to: can produce an optimal solution, addressing the limitation of the heuristic, which may yield only a near-optimal result. Although it requires more computation time than the heuristic method, it is faster compared to the B&B algorithm. The detailed aims of this study are as follows:

i To develop a Two-phase B&B algorithm to solve a specific VRP involving a maximum duration constraint.
ii To conduct a comparative study among the proposed method, B&B methods, and heuristic algorithms on several benchmark problem instances.
iii To demonstrate the efficiency of the proposed algorithm and identify its limitations.

The rest of the paper is organized as follows: the VRP model with maximum duration is presented in section 2. The proposed Two-Phase Branch and Bound algorithm is given in section 3. Section 4 presents the simulation and discussion. The conclusion is in the last section.

2. Model formulation

This section discusses the VRP model with maximum duration constraint. The objective of this problem is to minimize the total travel expenses by considering the distance and the travel speed. In contrast to the capacitated VRP,[28] this problem aims to determine the route of each vehicle while considering the constraint that the total travel and service duration within a route must not exceed the vehicle's maximum operational time in serving the customers. This problem assumes a deterministic service duration and travel distance.[29] Furthermore, several assumptions are made, including that the vehicle's speed is constant and the cost of travel depends solely on the trip's duration, which is influenced by distance and speed (not traffic conditions), so that fuel consumption is constant.

Most integer programming formulations of the classical VRP represent vehicle flows using binary variables that specify whether a vehicle travels between a pair of customers in the resulting solution.[30] Through this formulation, the decision variables simultaneously capture assignment constraints, which define the sequence of customer visits, and commodity flow constraints, which describe the movement of goods along the constructed routes.[31]

Based on the terminology of VRP,[13,32] the relationship between customers and a depot can be described as a graph. Let G be a complete graph with $N(G)$ as the set of vertices and $A(G)$ as the set of edges. Furthermore, the set $N(G)$ is written as N, and $A(G)$ is written as A. The set of vertices in the graph G is denoted as $N = \{0\} \cup N_p$, where 0 is the depot and N_p is the set of customers. The set $A = \{(i,j) \mid i,j \in N, i \neq j\}$ is the set of edges connecting vertex i to vertex j. Each edge $(i,j) \in A$ has corresponding distance h_{ij} which represents the distance from vertex i to vertex j. In other words, h_{ij} is the distance traveled by a vehicle from the location of customer i to the location of customer j. The set of vehicles is notated as P. Each vehicle has a maximum duration limit per assignment. This maximum duration limit is denoted by $T_{\max}$, where the service duration is calculated from when the vehicle arrives at the customer's location until the vehicle leaves to travel back. The duration of this service for any customer is denoted by s. Defining variables and parameters can be seen in Table 1.

The complete model is presented as follows. The decision variable relates to the assignment of vehicles to visit multiple customer locations. These trips form a route, with a vehicle visiting a customer exactly once. The variable takes the following form.

$$x_{ijk} = \begin{cases} 1, & \text{if the travel from customer } i \text{ to } j \\ & \text{is performed by vehicle } k, \\ 0, & \text{otherwise.} \end{cases}$$

The objective function is stated as follows:

$$\min f(x_{ijk}) = c \sum_{i \in N} \sum_{j \in N} \sum_{k \in P} \tau_{ij} x_{ijk}, \qquad (1)$$

with the duration of the vehicle's travel time in visiting a customer defined as

$$\tau_{ij} = \frac{h_{ij}}{v_f}.$$

The constraints include:

$$\sum_{\substack{i \in N \\ i \neq j}} \sum_{k \in P} x_{ijk} = 1, \qquad \forall j \in N_p \qquad (2)$$

$$\sum_{\substack{j \in N_p \\ j \neq i}} \sum_{k \in P} x_{ijk} = 1, \qquad \forall i \in N \qquad (3)$$

$$\sum_{i \in N} x_{ilk} - \sum_{\substack{j \in N \\ j \neq i}} x_{ljk} = 0, \qquad \forall k \in P,\ l \in N \qquad (4)$$

$$\sum_{j \in N_p} x_{0jk} = 1, \qquad \forall k \in P \qquad (5)$$

$$\sum_{j \in N_p} x_{j0k} = 1, \qquad \forall k \in P \qquad (6)$$

$$\sum_{i,j \in S} x_{ijk} \leq |S| - 1, S \subseteq \{1,2,\ldots,n\}, \quad |S| \geq 2,\ \forall k \in P \qquad (7)$$

$$\sum_{j \in N} \sum_{\substack{j \in N \\ j \neq i}} (\tau_{ij} + s) x_{ijk} \leq T_{\max}, \qquad \forall k \in P \qquad (8)$$

Equation 1 represents the objective function of the model. The objective is to minimize the total travel costs. Equations 2 and 3 represent that each customer is visited exactly once by a vehicle. Equation 4 represents that every vehicle that visits a customer, after serving the customer, will leave the customer. Equations 5 and 6 represent that every vehicle leaves and returns to the depot. Equation 7 represents the subtour elimination constraint. Equation 8 represents that the total working time of the vehicle in serving customers does not exceed the maximum working time given by the depot.

3. Proposed Method

Determining the shortest route is one of the problems in combinatorial optimization. Graphs are a basic combinatorial structure where many combinatorial optimization problems are most naturally expressed using concepts from graph theory.[33] As is known, the VRP with the maximum duration as explained in

Table 1. Description of notation

Notation	Description
N	Set of a depot and the customers
N_p	Set of customers
P	Set of vehicles
c	Fixed cost of vehicle travel per unit time
v_f	Vehicle speed
τ_{ij}	Duration of the vehicle's travel time from the location of customer i to customer j
h_{ij}	Distance traveled by a vehicle from the location of customer i to customer j
s	Service duration by vehicle in serving customer
$T_{\max}$	Maximum duration limit

the previous section can be represented as a problem in graph theory. The proposed algorithm is designed specifically to solve a particular type of problem, namely the VRP, where the distance or travel time constraints must not exceed the predetermined maximum limits. This proposed method modifies the B&B framework by dividing the algorithm into two phases, which is referred to as the Two-phase B&B algorithm. The first phase of the proposed method relies on a heuristic procedure, while the second phase applies dynamic programming.

In the first phase, a heuristic is used to obtain a tour. Since both phases are embedded within a B&B framework, the first phase follows best-first search strategy with complete branching. It expands all unexplored candidate solutions through wide branching and a heuristic mechanism until a full tour is obtained.

The best-first search strategy with complete branching is an enhancement of the best-first search strategy,Morrison *et al.*[1] in which branching occurs based on the minimal cost at each level until a complete tour is achieved. This strategy is more efficient and particularly well suited to the Two-phase B&B algorithm. Branching starts from node 0, which represents the depot location $l_{\pi(1)}$. A complete branching is then performed over all customers, resulting in paths from $l_{\pi(1)}$ to each subsequent node. The cost is evaluated to determine the path with the minimum cost from $l_{\pi(1)}$ to the next node. The node with the minimum cost or total traveled distance is then selected, and a complete branching is repeated toward the remaining nodes. This process continues iteratively until a complete tour is formed. The overall procedure is illustrated in Figure 1.

In the second phase, because any sub-partitions of a minimal tour remain minimal, a recursive step of the dynamic programming procedure is employed to identify an optimal configuration of subtours that satisfies the maximum duration constraint. The cost associated with this configuration serves as a temporary upper bound. To further narrow the search space, a pruning mechanism updates the upper bound whenever an improved tour configuration is found, and any branch whose cost exceeds this bound is subsequently discarded. In essence, the objective of the second phase is to identify an optimal partition of the tour that satisfies the constraint in Equation 8. The first and second phases are executed as a single iterative process. This process continues iteratively and terminates when no more branches can be explored, and the optimal solution is obtained.

To facilitate the explanation of the proposed method, the VRP illustrated in the previous section can be clarified into the following graph. Given a complete graph $G = (N, A)$, where N is the set of vertices representing the locations of customers and a depot, and A is the set of edges connecting each pair of vertices in N. This problem involves a set of locations $N = \{l_1, l_2, \ldots, l_n\}$ that includes the locations of all customers and the depot, and each pair of locations $(l_i, l_j) \in N \times N$ has a distance $h(l_i, l_j)$, which simplifies to h_{ij}. The solution to this problem is a tour commonly referred to as a Hamilton circuit, expressed

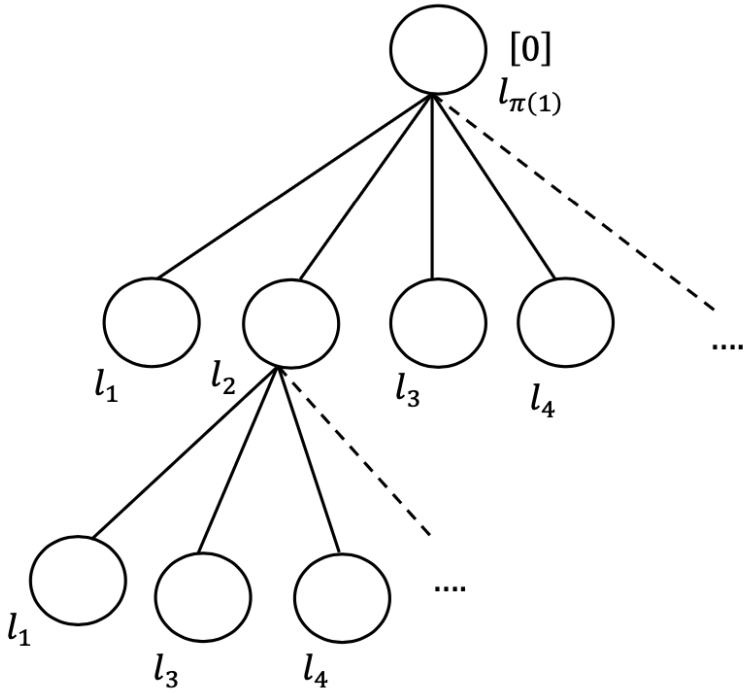

Figure 1. Best-first search strategy with complete branching.

as $(l_{\pi(1)}, l_{\pi(2)}, \ldots, l_{\pi(n)})$, which minimizes the following costs.

$$f = \frac{c}{v_f}\left(\sum_{i=1}^{n-1} h(l_{\pi(i)},\, l_{\pi(i+1)}) + h(l_{\pi(n)},\, l_{\pi(1)})\right), \tag{9}$$

where $\pi(n)$ represents the n-th sequence and $l_{\pi(1)}$ denotes the location of the depot.

An optimal tour indicates that there is no better sequence of vertices or customers than the one given by this optimal tour. This forms the basis on which the heuristic algorithm in the first phase is built.

In the first phase of the Two-phase B&B algorithm, a tour is required, which is the solution of the constraints in Equations 2–7 for $P = 1$. The resulting optimal tour has been defined by Lin[34] as follows.

Definition 1. [34] *A tour $(l_{\pi(1)}, l_{\pi(2)}, \ldots, l_{\pi(n)})$ is called n-optimal (or simply called n-opt) if there is no other tour with a smaller cost that replaces any n number of edges, with the cardinality of the edge set being n.*

Based on Definition 1, $n - opt$ tour can be obtained if there is a complete tour that begins at the depot, covers all customers, and ends at the depot. To obtain such a solution, a wide branching is done by adding one vertex or a customer to each existing sub-problem. Once a vertex has been added, the corresponding vertex in the solution is removed. Finding the cost can be obtained by choosing the minimum distance that can be formed for each initial vertex. The first branching process (route) starts from vertex $l_{\pi(1)}$, and calculates the cost denoted as $\hat{z}_{l_{\pi(1)}}$, obtained from

$$\hat{z}_{l_{\pi(1)}} = \sum_{i \in N} \min\{h_{il_1}, h_{il_2}, \ldots, h_{il_n}\}, \tag{10}$$

where h_{ij} is the elements of an adjacency matrix A_{ij} with $i, j \in N$. The value of $\hat{z}_{l_{\pi(1)}}$ is called as lower bound (LB). This leads to the definition of the LB, which is used as a convergence criterion.

Definition 2. *An LB value is a total cost obtained from the minimum distance formed by the initial vertex.*

After obtaining the LB, the next step is to begin branching by adding one vertex or customer after the depot, or node 0. This step is carried out by introducing a pivot. The intersection of the two vertices on A is called a pivot, so the next branching is to assign the value ∞ to the p-th row and q-th column of the A matrix. This shows that the route order pair (p, q) has been chosen so that travel is impossible from vertex p to a vertex other than q, and from a vertex other

than vertex p to a vertex q. Consequently, the cost on branch q for the next vertex from vertex p can be obtained from

$$\hat{z}_{(p,q)} = \sum_{i \in N} z_i, \tag{11}$$

$$z_i = \begin{cases} h_{pq}, & \text{if } p \text{ is a pivot row or } h_{ij} = \infty, \forall j \in N, \\ \min\{h_{il_1}, h_{il_2}, \ldots, h_{il_n}\}, & \text{otherwise.} \end{cases}$$

The goal of the first phase is to find a tour effectively by employing a best-first search strategy with complete branching. Following this procedure, a sub-problem is selected and explored based on the lowest cost, and then the process continues to branching using Equation 11 until a tour is obtained. The procedure is described in detail in the following pseudocode.

Algorithm 1: A tour construction

Input: Set of nodes N, and $A = (a_{ij})$
Output: Updated UB

Select $l_{\pi(1)}$ as the starting node;
Compute $\hat{z}_{l_{\pi(1)}}$;
Set $p \leftarrow l_{\pi(1)}$ and $S_t \leftarrow \{p\}$;
$t \leftarrow 1$;
for $i \leftarrow 2$ **to** $|N| - t$ **do**
    Append a node q to p with $q \in N \setminus S_t$, forming the path (p, q);
    Assign ∞ to the pivot row and column of matrix A and set $a_{qp} \leftarrow \infty$;
    Compute the cost $\hat{z}_{(p,q)}$;
    Update $S_{t+1} \leftarrow S_t \cup \{q\}$;
Select the node with the minimum value of $\hat{z}_{(p,q)}$ and set it as p;
Update $t \leftarrow t + 1$;
if $t < |N|$ **then**
    Go back to Step 5;
else
    Update UB using the minimum value of $\hat{z}_{(p,q)}$ over the complete tour if it improves the previous UB;
    End;

The algorithm then proceeds to the second phase, where the tour obtained from the first phase is partitioned into subtours that satisfy he constraint in Equation 8 using dynamic programming. A theorem is developed as the basis for the construction of the shortest path search algorithm derived from the optimality principle by Cormen *et al.*[35]

Theorem 1. [35] *For a graph whose edge weights are non-negative, any sub-path of a minimal path is itself a minimal path.*

This property follows from the optimal substructure of shortest paths, where any sub-path of a shortest path is itself a shortest path. Next, we will give the triangle inequality theorem used in determining the shortest path.

Theorem 2. [35] *For all* $l_i, l_j, l_n \in N$, *it holds* $h^*(l_{\pi(1)}, l_{\pi(n)}) \leq h^*(l_{\pi(1)}, l_{\pi(m)}) + h^*(l_{\pi(m+1)}, l_{\pi(n)})$, *where* $h^*(l_i, l_j)$ *is the minimum cost of the path from* l_i *to* l_j.

Based on both theorems, two or more paths that are each minimal will produce a minimal overall path, and vice versa; if there is a minimal path, then the cut of the path is also minimal. Therefore, the partitioned paths can be found using a dynamic programming algorithm. Dynamic programming algorithm is developed to handle multi-stage processes that exhibit specific unchanging characteristics. It offers a structured method to identify the optimal set of decisions.[36–38]

Adopting the use of dynamic programming algorithm with recursion step, the second phase of the Two-phase B&B algorithm proposes a dynamic programming approach to find optimal subtours based on the constraint in Equation 8. Suppose $f(\Phi)$ is the minimum cost found in the first phase, then there is a possibility that the obtained tour $(l_{\pi(1)}, l_{\pi(2)}, \ldots, l_{\pi(n)})$ does not meet the constraint in Equation 8. Therefore, the tour will be checked for all possibilities that can meet the constraint by making it as subtours. Each subtour shows a trip made by another vehicle.

Suppose $f_i(l_i)$ is the shortest distance to vertex l_i at stage i, and define $h(l_{i-1}, l_i)$ as the distance from vertex l_{i-1} to vertex l_i, then f_i is calculated based on f_{i-1} using the following equation.

$$f_i(l_i) = \min_{\substack{\text{all feasible} \\ (l_{i-1}, l_i)\text{route}}} [h(l_{i-1}, l_i) + f_{i-1}(l_{i-1})] \quad i = 1, 2, \ldots, n. \tag{12}$$

The maximum duration constraint for each vehicle is explicitly taken into account at the second stage, which uses a dynamic programming algorithm. There is a path that permits a vehicle to leave and/or return to the depot at every stage, according to the dynamic programming

value function. Therefore, Equation 12 can be stated as

$$f_i(l_i) = \min_{\substack{\text{all feasible}\\(l_{i-1},l_i)\\\text{partitions}}} \begin{cases} h(l_{i-1},l_i) + f_{i-1}(l_{i-1}), \\ h(l_{i-1},l_i) + h(l_i,l_0) + f_{i-1}(l_{i-1}), \\ h(l_0,l_i) + f_{i-1}(l_0), \\ h(l_0,l_i) + h(l_i,l_0) + f_{i-1}(l_0) \end{cases} \quad i = 1,2,\ldots,n. \tag{13}$$

where l_0 is a starting vertex or a depot location. The following theorem establishes that an optimal tour yields an optimal sequence of tour partitions under this constraint:

Theorem 3. *Let $h^*(l_{\pi(1)}, l_{\pi(n)})$ denote the optimal distance of a tour. For the particular problem considered in this study, any partition of this tour constitutes an optimal route for each vehicle if it satisfies $\frac{1}{v_f}\left(\sum_{i=1}^{n_t-1} h(l_{\pi(i)}, l_{\pi(i+1)}) + h(l_{\pi(n_t)}, l_{\pi(1)})\right) \leq T_{\max} - n_t s$, where n_t denotes the number of vertices in partition t.*

Proof. Suppose there exists an optimal tour with n customers, which is decomposed into n stages. Let $\phi_i(l_{\pi(i)})$ denote the minimum travel distance to customer i at stage i. As in Equation 13, the value function is defined as $\phi_i(l_i) = \begin{cases} h(l_{\pi(i-1)}, l_{\pi(i)}) + \phi_{i-1}(l_{\pi(i-1)}), \\ h(l_{\pi(i-1)}, l_{\pi(i)}) + h(l_{\pi(i)}, l_{\pi(1)}) + \phi_{i-1}(l_{\pi(i-1)}), \\ h(l_{\pi(1)}, l_{\pi(i)}) + \phi_{i-1}(l_{\pi(1)}), \\ h(l_{\pi(1)}, l_{\pi(i)}) + h(l_{\pi(i)}, l_{\pi(1)}) + \phi_{i-1}(l_{\pi(1)}) \end{cases}$ for $i = 1,2,\ldots,n$, where $\pi(1)$ denotes the depot. Furthermore, suppose that each stage has a value function ϕ_i satisfying $\frac{1}{v_f}\left(\sum_{i=1}^{n_t-1} h(l_{\pi(i)}, l_{\pi(i+1)}) + h(l_{\pi(n_t)}, l_{\pi(1)})\right) \leq T_{\max} - n_t s$. Consequently, this condition implies that the constraint in Equation 8 is satisfied, and therefore the corresponding tour partition is feasible. Let $(l_{\pi(1)}, l_{\pi(n)})$ be a globally feasible and optimal solution. Assume that there exists a partition of $(l_{\pi(1)}, l_{\pi(n)})$ that is feasible but not optimal, corresponding to the decisions from stage i to stage n. Then there exists another partition of the same tour with a strictly smaller total cost from stage i to stage n, assuming constant cost parameters and vehicle speed v_f. Replacing this partition yields a global solution with a smaller total cost across all vehicles, contradicting the optimality of the original solution. Hence, the assumption that a partition from stage k is not optimal is false. Therefore, every feasible partition of a globally optimal tour constructed via a recursive step must itself be optimal. □

The search process for the optimal solution is conducted by calculating a backward procedure starting from the n-th stage to the first stage. The objective function reached from the optimal solution in the second phase is called as upper bound (UB). The next definition is related to the pruning strategy using UB. In the pruning strategy, the UB is introduced, which is the minimum cost that has been found for the tour that has been explored at that time.

Definition 3. *Suppose there is a tour solution of the branch, namely Φ_i, or commonly called a sub-problem, with $i \in \eta_t$ representing the index of the sub-problem branch that appears at the t iteration. A sub-problem with tour solution Φ_i can be called bounded if it satisfies $f(\Phi_i) \leq f(\Phi_{UB})$, where $f(\Phi_{UB})$ is the UB.*

The condition for a search space to be bounded is when a tour solution is found in the sub-problem that is smaller than the UB. As a result, the UB value can be updated with $f(\Phi_{UB}) = f(\Phi_i)$. The algorithm proposed also ensures that each iteration has a feasible solution, namely a tour that can be updated based on Definition 3. Therefore, the algorithm presents it in the following Lemma:

Lemma 1. *Each t-th iteration contains a solution of the Φ_i tour, with $i \in \eta_t$ representing the sub-problem branch index.*

Proof. Let Φ_{i_j} be a solution to the sub-problem with $j = l_{\pi(1)}, l_{\pi(2)}, \ldots, l_{\pi(n)}$. Each branch of the sub-problem is done by adding one decision variable with the value of $x_{pq} = 1$ with $p, q \in N, p \neq q$ until $q = n$. Consequently, there is a complete tour solution to Φ_{i_j} starting from $j = l_{\pi(1)}$ to $j = l_{\pi(n)}$. □

Another sub-problem is explored continuously until a complete tour is formed, and the UB of the current iteration is updated if the tour cost is better than the UB from the previous iteration. The next convergence theorem guarantees that the algorithm will not terminate before the stopping criterion is satisfied.

Theorem 4. *Suppose the minimum or $n-opt$ tour solution is $\Phi_{\min} = \arg\min f(\Phi_i)$, where $i \in \eta_t$ represents the sub-problem branch index that occurs at the t-th iteration. Furthermore, for each $\varepsilon > 0$, there are a number of iterations $t \geq 1$ such that $f(\Phi_{\min}) - f(\tilde{\Phi}_{LB}) < \varepsilon$.*

Proof. For each t iteration, the property $f(\tilde{\Phi}_{LB}) \leq f(\Phi_{\min}) \leq f(\Phi_{UB})$ and the increasing number of sub-problems explored results in the search space size getting smaller, so that

$f(\Phi_{UB}) - f(\tilde{\Phi}_{LB}) < \varepsilon$ applies. After finite iterations, based on Lemma 1, the $n-opt$ tour solution is obtained with $f(\Phi_{UB}) = f(\Phi_{\min})$. As a result, $f(\Phi_{\min}) - f(\tilde{\Phi}_{LB}) < \varepsilon$. □

Each iteration of the Two-phase B&B process comprises a first phase, which is then followed by a second phase. This process is performed iteratively by updating the value of UB until an optimal solution is obtained, that is, when no remaining solution space can yield a value better than the current UB. Next, a theorem is given that guarantees that the first phase and second phase can solve the given problem.

Theorem 5. *The first phase and the second phase of the Two-phase B&B algorithm solve the VRP problem with maximum duration constraint.*

Proof. In the Two-phase B&B framework, the first phase obtains a minimal tour, which is then passed to the second phase to determine the optimal routes for each vehicle corresponding to that tour. This procedure is carried out iteratively until the conditions of Theorem 4 are satisfied, in other words, the optimal solution is obtained. □

Theorem 5 shows that the proposed B&B algorithm can find the optimal solution of the problem. A numerical example is provided to illustrate the solution of the problem, which involves partitioning a tour to obtain optimal routes for each vehicle associated with the tour, as a validation of this procedure.

Example 1. *Consider a set of nodes* $N = \{0, 1, 2, 3, 4, 5\}$*, where node* 0 *denotes the depot. Suppose the first phase of the Two-phase B&B algorithm in iteration i generates the tour 0-5-3-4-2-1-0. In the second phase, this tour is partitioned into feasible vehicle routes subject to the maximum duration constraint* $T_{\max} = 400$*. Given that* $v_f = 0.5$*,* $s = 90$*, and* $c = 2$*,*

$$A = \begin{bmatrix} \infty & 18.6 & 20.6 & 16.1 & 18.1 & 15.1 \\ 18.7 & \infty & 2.0 & 3.6 & 3.0 & 4.2 \\ 20.6 & 2.0 & \infty & 5.0 & 3.6 & 5.8 \\ 16.1 & 3.6 & 5.0 & \infty & 2.0 & 1.0 \\ 18.1 & 3.0 & 3.6 & 2.0 & \infty & 3.0 \\ 15.1 & 4.2 & 5.8 & 1.0 & 3.0 & \infty \end{bmatrix}.$$

The iterative procedure in the second stage of the dynamic programming framework is illustrated in Figure 2. At the end of stage 5 in dynamic programming, an optimal cost of $f = 298$ *is obtained, corresponding to a travel duration of* 149*. By performing backward tracing for each stage, the resulting partition is 0-5-3-0 and 0-4-2-1-0. Looking at another partition, such as 0-5-3-4-0 and 0-2-1-0, yields a travel duration of* 154.8*, corresponding to* $f = 309.6$*. The recursive step selects this solution over other possible partitions, such as 0-5-3-4-0 and 0-2-1-0, although all of them satisfy* $T_{\max}$*, because the optimal partition yields a smaller objective function value than these alternatives.*

Algorithm 2: Two-phase branch and bound algorithm

Input: A set of depot and customer locations, c, v_f, s, and $T_{\max}$
Output: Optimal routing solution

```
Initialize NodeList ← ∅;
Set ẑ_{l_{π(1)}} as LB, UB ← ∞;
while true do
    /* Phase 1: First-level */
    node ← Append a new vertex to the path ; // Perform for each branching
    Determine cost;
    Select node with the lowest cost and add other node to NodeList;
    if node is a tour then
        /* Phase 2: Second-level */
        Evaluate tour using recursive procedure ; // Perform dynamic programming
        if cost ≤ UB then
            UB ← cost;
            Update solution;
    foreach node in NodeList do
        if cost > UB then
            Remove node from NodeList ; // Perform pruning
    if NodeList = ∅ then
        break ; // Stopping condition
return solution
```

The algorithm is developed based on the theoretical foundation presented earlier, where the iterative process involving the two phases terminates when all branches of the sub-problems have been fully explored. The steps to solve this problem using the Two-phase B&B algorithm are explained in pseudocode.

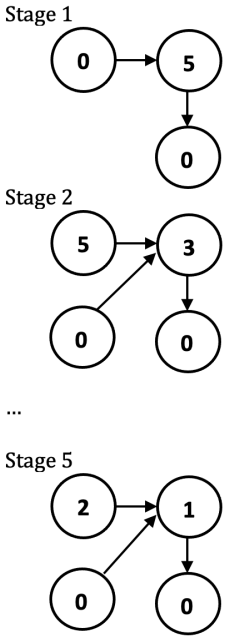

Figure 2. Dynamic programming stages.

4. Computational results and discussion

In this section, the computational results of using the Two-phase B&B algorithm to solve the VRP with maximum duration are discussed. The data used was obtained from the Solomon's benchmark instance VRP with Time Windows[39] in the form of the location coordinates of each customer. The data were taken for 20 customers in C101, R101, and RC101. The computational results of the Two-phase B&B algorithm were compared with two alternative B&B algorithms employing wide and binary branching techniques.[1,9,40]

The wide and binary branching described here utilize an integer linear programming approach to find the optimal solution. This branching technique is commonly used to solve integer optimization problems. The wide branching strategy, using the integer linear programming approach,[40] referred to in this article as B&B1, performs branching at the initial level for each decision variable. In contrast, binary branching divides the branching process into two mutually exclusive branches, which continue to expand based on the number of decision variables.[9] The B&B algorithm with this binary branching strategy is hereafter referred to as B&B2. For both alternative algorithms used, Gurobi Optimization, LLC: USA, was employed to find the optimal solution for each branch by relaxing binary variables into continuous ones.

Since the Two-phase B&B algorithm incorporates heuristic elements in its modification, numerical experiments were also conducted on two well-known heuristic algorithms, namely the improved TS[22] and the adaptive large neighborhood search (ALNS),[23] to enable a more comprehensive analysis of the proposed algorithm's efficiency. The TS algorithm is widely recognized for its ability to store previously explored solutions, thereby preventing redundant searches. The ALNS operates by gradually improving an initial solution through a series of removal and insertion operators applied in a systematic manner.

The performance of the proposed algorithm was evaluated based on several criteria, including the obtained UB, the number of explored nodes or branches, and the computation time. The solution algorithms were developed using Python Software Foundation: USA, and tested through computational experiments conducted on a computing cluster equipped with an Intel(R) Core(TM) i5-8257U CPU running at 1.40 GHz. This study used the Euclidean distance to obtain the distance between locations so that the matrix obtained was a symmetric matrix. The experiments were carried out with varying data sizes or numbers of customers : 5, 8, 10, 15, 20, 30, and 50 using $c = 2$, $v_f = 0.5$, and varying values of $T_{\max}$.

The datasets were categorized into three types: small instances with 5 and 8 customers, medium instances with 10 and 15 customers, and large instances with 20, 30, and 50 customers. In addition to comparisons with B&B1, B&B2,

improved TS, and ALNS, supplementary experiments were conducted utilizing Gurobi, employing a branch-and-cut algorithm for small and medium cases to verify solution optimality and efficacy. For datasets with sizes 5, 8, 10, 15, and 20, the Two-phase B&B, B&B1, B&B2, and Gurobi were terminated after a maximum running time of 180. Whereas for datasets with sizes 30 and 50 the maximum computation time was set to 5×10^3. The comparative heuristic algorithms, namely improved TS and ALNS, were tested for each parameter combination with 10 independent runs, with the maximum number of iterations set to 100. Consequently, the solutions documented for the heuristic approaches signify the best outcomes achieved upon the conclusion of the iterative procedure.

Table 2 displays the computational results for small and medium datasets, which are based on the UB obtained after completing the optimal solution search process. For heuristic algorithms, including improved TS and ALNS, the UB corresponds to the best objective value achieved throughout the total computational duration. Among all types of datasets with varying numbers of customers, the Two-phase B&B algorithm consistently achieves better UB than the other five algorithm. Although the five comparison algorithms can obtain the same best solutions as the proposed algorithm, the consistency of the proposed algorithm across different numbers of customers is superior.

Furthermore, the UB obtained by the Two-phase B&B algorithm are identical to those produced by the Gurobi solver, indicating that the corresponding solutions are optimal. This algorithm successfully obtains both best and valid solutions for the given dataset sizes. The heuristic procedure in the first phase effectively finds possible solutions, which may not be unique, with respect to the constraints in Equations 2–7, while the dynamic programming algorithm in the second phase identifies the optimal solution according to the constraint in Equation 8 among all those obtained in the first phase. In the first phase of the Two-phase B&B algorithm, it is crucial to store multiple non-unique solutions for the obtained UB, as not every solution from the first phase will become the optimal solution to the VRP under the given constraints. Dynamic programming is responsible for finding a series of optimal subtours that satisfy the maximum duration constraint of the overall solution obtained in the first phase.

Considering the two comparison algorithms, B&B1 and B&B2, and focusing on the number of explored nodes as presented in Table 2, it is evident that the Two-phase B&B algorithm requires fewer node explorations than the other two algorithms. This difference arises because the B&B1 algorithm performs only limited exploration of sub-problems when the optimal solution consists of a single tour or a single vehicle. The algorithm does not further explore cases involving more than one vehicle, i.e., for $k > 1$. Although B&B1 explores fewer nodes in several cases, it was unable to consistently produce optimal and valid solutions for larger datasets. Given its poor performance on larger instances, it can be concluded that the Two-phase B&B algorithm generally requires fewer branch explorations than the other two algorithms for any value of k.

Similarly, the Two-phase B&B algorithm demonstrates faster computation times compared to the B&B1 and B&B2 algorithms. Its execution time is significantly faster, particularly when compared to B&B2. While B&B1 may show faster computation in some instances, such as the trial with 20 customers, this is primarily due to the algorithm terminating early as it fails to find an optimal and valid solution. This result indicates that the Two-phase B&B algorithm is more efficient than the other two B&B algorithms in finding the optimal solution. The B&B algorithm successfully reduced computation time by dividing the process into two phases. Although both phases require a non-trivial amount of computation time, the first phase effectively shortens the overall time by identifying the optimal solution for a single tour. As a result, the second phase becomes faster since it only explores the already optimal tour solution to find possible subtours that satisfy the given constraints, referred to in this context as valid solutions.

In comparison with the other heuristic algorithms, namely the improved TS and ALNS, the Two-phase B&B algorithm provides a fairly competitive computation time for small-sized datasets. For larger datasets, the time required to obtain the best solution is longer than that of the two heuristic algorithms. However, this is proportional to the quality of the results, as the Two-phase B&B algorithm is able to obtain better solutions. In contrast, the two heuristic algorithms tend to become trapped in local optima, causing the search process to stop early or to produce identical solutions before reaching the maximum number of iterations.

Additional computations were conducted on large-scale datasets to further evaluate the

Table 2. Comparison of the upper bound, computation time, and number of nodes for different algorithms under small- and medium-sized datasets

Number of Cust	Dataset	Algorithms	UB		Computation Time (s)		Number of Nodes	
			$T_{max} = (150; 400)$					
5	C101	Two-Phase B&B	**709.32**	**298.62**	0.010	0.010	**51**	**51**
		B&B1	**709.32**	**298.62**	0.140	75.10	152	130
		B&B2	**709.32**	**298.62**	7.060	1.060	1,802	368
		Gurobi	**709.32**	**298.62**	0.130	0.060		
		Improved TS	**709.32**	**298.62**	0.014	0.030		
		ALNS	**709.32**	**298.62**	**0.004**	**0.004**		
	R101	Two-Phase B&B	**569.04**	**477.93**	0.010	0.010	**65**	55
		B&B1	**569.04**	**477.93**	123.10	0.020	112	**32**
		B&B2	**569.04**	**477.93**	88.09	0.250	1,129	217
		Gurobi	**569.04**	**477.93**	0.050	0.010		
		Improved TS	626.42	**477.93**	0.030	0.020		
		ALNS	**569.04**	**477.93**	**0.003**	**0.006**		
	RC101	Two-Phase B&B	**337.07**	**337.07**	**0.000**	**0.010**	**50**	48
		B&B1	**337.07**	**337.07**	103.00	0.040	165	**32**
		B&B2	**337.07**	**337.07**	90.30	0.280	1,203	217
		Gurobi	**337.07**	**337.07**	0.020	0.030		
		Improved TS	582.94	**337.07**	0.030	0.020		
		ALNS	**337.07**	**337.07**	**0.000**	**0.010**		
			$T_{max} = (200; 600)$					
8	C101	Two-Phase B&B	**1,134.2**	**319.38**	0.020	0.020	**170**	**170**
		B&B1	**1,134.2**	**319.38**	1.170	88.50	578	690
		B&B2	**1,134.2**	**319.38**	140.08	128.90	10,370	9,690
		Gurobi	**1,134.2**	**319.38**	4.760	0.110		
		Improved TS	**1,134.2**	330.92	0.160	0.089		
		ALNS	**1,134.2**	330.92	**0.006**	**0.014**		
	R101	Two-Phase B&B	**639.78**	**584.49**	0.020	0.020	**247**	**232**
		B&B1	**639.78**	**584.49**	126.80	110.30	698	590
		B&B2	**639.78**	**584.49**	178.90	1.600	1,890	785
		Gurobi	**639.78**	**584.49**	0.060	0.020		
		Improved TS	701.47	**584.49**	0.130	0.080		
		ALNS	**639.78**	**584.49**	**0.006**	**0.010**		
	RC101	Two-Phase B&B	**383.53**	**383.53**	0.020	0.070	**943**	**64**
		B&B1	462.87	**383.53**	180.00	1.740	2,870	74
		B&B2	462.87	**383.53**	180.00	0.110	3,201	784
		Gurobi	**383.53**	**383.53**	0.020	0.060		
		Improved TS	651.99	**383.53**	0.110	0.080		
		ALNS	**384.53**	**383.53**	**0.010**	**0.010**		
			$T_{max} = (300; 700)$					
10	C101	Two-Phase B&B	**761.23**	**341.09**	0.030	0.140	**389**	8,400
		B&B1	1,132.46	**341.09**	180.00	178.00	2,902	**1,608**
		B&B2	1,132.46	**341.09**	180.00	17.730	6,880	1,903
		Gurobi	**761.23**	**341.09**	0.280	0.140		
		Improved TS	**761.23**	344.09	0.220	0.240		
		ALNS	**761.23**	344.09	**0.007**	**0.016**		
	R101	Two-Phase B&B	**781.61**	**692.16**	0.140	0.140	**7,301**	**7,301**
		B&B1	861.46	**692.16**	180.00	0.120	9,280	9,112
		B&B2	861.46	**692.16**	180.00	120.30	10,983	12,209
		Gurobi	**781.61**	**692.16**	0.170	0.210		
		Improved TS	**781.61**	**692.16**	0.130	0.160		
		ALNS	792.85	**692.16**	**0.013**	**0.027**		
	RC101	Two-Phase B&B	**664.19**	**551.12**	0.020	0.150	**100**	7,561
		B&B1	**664.19**	**551.12**	78.50	0.120	250	**4,509**
		B&B2	**664.19**	**551.12**	18.17	92.87	3,076	7,688
		Gurobi	**664.19**	**551.12**	0.120	0.600		
		Improved TS	**664.19**	562.98	0.190	0.150		
		ALNS	**664.19**	**551.12**	**0.016**	**0.030**		
			$T_{max} = (500; 800)$					
15	C101	Two-Phase B&B	**821.39**	**543.98**	0.280	17.070	**1,447**	37,062
		B&B1	2,162.81	1,485.48	180.00	180.00	15,799	**15,097**
		B&B2	1,076.22	1,612.28	180.00	180.00	17,390	16,699
		Gurobi	**821.39**	**543.98**	6.270	13.290		
		Improved TS	**830.28**	671.75	0.500	0.800		
		ALNS	943.01	561.86	**0.010**	**0.030**		
	R101	Two-Phase B&B	**873.01**	**811.84**	17.54	17.54	**36,952**	**36,901**
		B&B1	1,017.34	916.66	180.00	180.00	42,677	41,890
		B&B2	1,053.58	916.66	180.00	180.00	44,789	47,678
		Gurobi	**873.01**	**811.84**	18.49	18.03		
		Improved TS	912.53	**811.84**	0.640	0.510		
		ALNS	929.62	**811.84**	**0.030**	**0.070**		
	RC101	Two-Phase B&B	**614.5**	**614.5**	17.61	18.02	**35,790**	**36,949**
		B&B1	1,126.91	1,126.91	180.00	180.00	36,892	37,000
		B&B2	747.77	747.77	180.00	180.00	45,899	42,789
		Gurobi	**614.5**	**614.5**	78.05	178.00		
		Improved TS	646.37	646.37	0.510	0.510		
		ALNS	**614.5**	**614.5**	**0.078**	**0.077**		

Note: Bold values indicate the best performance.

Table 3. Comparison of the upper bound, computation time, and number of nodes for different algorithms under a large-sized dataset

Number of Cust	Dataset	Algorithms	UB		Computation Time (s)		Number of Nodes	
			$T_{max} = (500; 800)$					
20	C101	**Two-Phase B&B**	**1,211.73**	**895.93**	101.73	107.88	**7,316**	**7,394**
		B&B1	1,529.77	1,224.72	180.00	180.00	14,753	16,538
		B&B2	1,330.27	1,055.05	180.00	180.00	28,653	26,890
		Improved TS	**1,211.73**	902.97	1.220	1.320		
		ALNS	1,292.95	904.85	**0.019**	**0.030**		
	R101	**Two-Phase B&B**	**1,024.23**	**923.54**	83.57	88.44	**6,330**	**6,432**
		B&B1	1,193.94	1,191.19	180.00	180.00	16,783	23,847
		B&B2	1,193.94	1,133.56	180.00	180.00	20,937	23,048
		Improved TS	**1,024.23**	990.31	1.000	1.040		
		ALNS	1,044.99	**923.54**	**0.050**	**0.150**		
	RC101	**Two-Phase B&B**	**1,020.24**	**909.96**	71.60	72.78	**11,612**	**11,346**
		B&B1	1,175.21	922.62	180.00	180.00	18,924	22,352
		B&B2	1,153.62	992.62	180.00	180.00	32,894	42,562
		Improved TS	**1,020.24**	913.34	0.990	1.13		
		ALNS	1,034.29	**909.96**	**0.180**	**0.150**		
			$T_{max} = (1000; 1500)$					
30	C101	**Two-Phase B&B**	**881.66**	**789.56**	4,935.0	3,158.6	**2,654,342**	**2,586,703**
		B&B1	1,092.60	953.78	5,000.0	5,000.0	3,192,070	2,590,475
		B&B2	1,084.26	953.19	5,000.0	5,000.0	2,863,731	2,688,912
		Improved TS	1,062.88	945.31	4.260	5.800		
		ALNS	982.98	973.38	**0.068**	**0.095**		
	R101	**Two-Phase B&B**	**1,258.40**	**1,258.40**	4,627.0	3,084.0	**2,097,442**	**2,099,345**
		B&B1	1,551.39	1,308.67	5,000.0	5,000.0	2,103,069	2,205,712
		B&B2	1,338.57	1,287.61	5,000.0	5,000.0	2,170,264	2,221,440
		Improved TS	1,382.26	1,382.26	1.850	1.840		
		ALNS	1,287.61	1,287.61	**0.230**	**0.340**		
	RC101	**Two-Phase B&B**	**1,231.28**	**1,231.28**	4,673.0	4,673.0	**182,935**	**187,354**
		B&B1	1,250.18	1,269.29	5,000.0	5,000.0	208,737	217,882
		B&B2	1,234.94	1,250.18	5,000.0	5,000.0	210,904	193,889
		Improved TS	1,242.09	1,242.08	2.470	3.170		
		ALNS	**1,231.28**	**1,231.28**	**0.330**	**0.260**		
			$T_{max} = (1200; 2000)$					
50	C101	**Two-Phase B&B**	**984.82**	**960.10**	4,589.0	4,398.0	**583,947**	**290,651**
		B&B1	1,405.43	1,256.56	5,000.0	5,000.0	596,272	302,782
		B&B2	1,398.85	1,256.56	5,000.0	5,000.0	580,282	318,862
		Improved TS	1,048.83	1,048.43	18.210	18.260		
		ALNS	1,571.46	1,323.41	**0.180**	**1.080**		
	R101	**Two-Phase B&B**	**1,869.64**	**1,786.99**	3,678.0	2,785.0	**44,383**	**1,230**
		B&B1	1,992.28	1,861.26	5,000.0	5,000.0	57,892	3,889
		B&B2	1,971.65	1,847.33	5,000.0	5,000.0	62,688	5,317
		Improved TS	2,058	1,934.08	24.34	18.24		
		ALNS	2,173.55	1,861.26	**0.870**	**2.150**		
	RC101	**Two-Phase B&B**	**1,601.48**	**1,487.38**	4,087.0	4,976.0	**432,175**	**396,355**
		B&B1	1,650.25	1,516.93	5,000.0	5,000.0	456,822	417,822
		B&B2	1,638.22	1,507.72	5,000.0	5,000.0	466,782	428,862
		Improved TS	1,655.82	1,545.50	17.47	18.65		
		ALNS	1,852.97	1,507.72	**5.150**	**2.000**		

Note: Bold values indicate the best performance.
Abbreviations: ALNS: Adaptive large neighborhood search; B&B: Branch and bound; TS: Tabu search; UB: Upper bound.

performance of the Two-phase B&B algorithm, as reported in Table 3. The table indicates that the Two-phase B&B algorithm achieves the optimal solution, but with greater computational time than the improved TS and ALNS algorithms. Although the proposed algorithm outperforms the traditional B&B approach and guarantees that an optimal solution to the problem can be found, the computational effort increases exponentially as the number of customers grows.

Despite its ability to consistently obtain optimal solutions, the proposed Two-phase B&B algorithm has several limitations. The algorithm is designed for VRP instances characterized by a distinct constraint structure, emphasizing a maximum route duration and homogeneous service times. Under this structure, the method can be directly applied to problems with similar characteristics, such as homogeneous customer demands with vehicle capacity constraints. However, in real-world applications, both service durations and customer demands often vary across customers. Further modifications to the dynamic programming structure would be necessary to accommodate heterogeneous service schedules or demands. Since this study represents an initial contribution, extending the Two-phase B&B algorithm to more complex problem settings is left for future research.

Overall, the computational results show that the Two-phase B&B algorithm outperforms the two traditional B&B algorithms and the two heuristic algorithms in terms of performance. For large-scale datasets, this algorithm still requires considerable exploration time to obtain the optimal solution. As the dataset size increases, the number of explored nodes and the computation time increase accordingly.

5. Conclusion

This study investigates the VRP with a maximum duration constraint and proposes a Two-phase B&B algorithm designed to guarantee optimal solutions efficiently. The algorithm consists of two stages: the first phase identifies a tour that satisfies the classical VRP constraints, while the second phase applies dynamic programming to find feasible combinations of optimal subtours that meet the maximum duration constraint. The computational results demonstrate that the Two-phase B&B algorithm performs efficiently and consistently in obtaining an optimal solution. Although the computation time and the number of explored nodes increase as the dataset size grows, the algorithm remains robust and continues to reach optimal solutions across all tested instances. Overall, these findings confirm that the proposed method is both reliable and effective for solving the VRP with maximum duration constraints.

Our research represents an initial investigation of the integration between heuristic and B&B approaches, which is restricted to a special-case formulation of the VRP. There are several directions and suggestions for future research. First, still related to duration-based constraints, the VRP model can be extended to incorporate customer time windows with heterogeneous service durations. Furthermore, since pruning strategies play a significant role in reducing the search space, further modifications may be applied to the search strategy by adopting more efficient branching techniques that can predict promising nodes.

Acknowledgments

The authors are truly thankful for the financial assistance received from the Indonesian Education Scholarship (BPI), the Center for Higher Education Funding and Assessment (PPAPT), and the Indonesian Endowment Fund for Education (LPDP).

Funding

None.

Conflict of interest

The authors report no conflicts of interest. The authors alone are responsible for the content and writing of this article.

Author contributions

Conceptualization: All authors
Formal analysis: Asri Bekti Pratiwi
Methodology: Asri Bekti Pratiwi, Salmah
Writing–original draft: Asri Bekti Pratiwi
Writing–review & editing: Salmah Salmah

Availability of data

The data that support the findings of this study are openly available in CVRPLIB at http://vrp.galgos.inf.puc-rio.br/index.php/en/, reference number [39].

AI tools statement

The authors confirm that no AI tools were used in the preparation of this manuscript.

References

1. Morrison DR, Jacobson SH, Sauppe JJ, Sewell EC. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optim.* 2016;19:79–102. https://www.doi.org/10.1016/j.disopt.2016.01.005
2. Taha AH. Operations Research: An Introduction. 8th ed. Upper Saddle River, New Jersey; 2006. https://www.amazon.com/Operations-Research-Introduction-Hamdy-Taha
3. Theurich F, Fischer A, Scheithauer G. A branch-and-bound approach for a vehicle routing problem with customer costs. *EURO J Comput Optim.* 2021;9:100003. https://www.doi.org/10.1016/j.ejco.2020.100003
4. Duman EN, Tas D, Catay B. A bidirectional branch-and-price algorithm with pulse procedure for the electric vehicle routing problem with flexible deliveries. *Transp Res Part C Emerg Technol.* 2024;165:104699. https://www.doi.org/10.1016/j.trc.2024.104699
5. Ozbaygin G, Karasan OE, Savelsbergh M, Yaman H. A branch-and-price algorithm for the vehicle routing problem with roaming delivery locations. *Transp Res Part B Methodol.* 2017;100:115–137. https://www.doi.org/10.1016/j.trb.2017.02.003
6. Li J, Qin H, Baldacci R, Zhu W. Branch-and-price-and-cut for the synchronized vehicle routing problem with split delivery, proportional service time and multiple time windows. *Transp Res Part E Logist Trans Rev.* 2020;140:101955. https://www.doi.org/10.1016/j.tre.2020.101955
7. Laporte G, Nobert Y, Taillefer S. A branch-and-bound algorithm for the asymmetrical distance-constrained vehicle routing problem. *Math Model.* 1987;9(12):857–868. https://www.doi.org/10.1016/0270-0255(87)90004-2

8. Zhang X, Chen L, Gendreau M, Langevin A. A branch-and-cut algorithm for the vehicle routing problem with two-dimensional loading constraints. *Eur J of Oper Res.* 2020;302(1):259–269. https://www.doi.org/10.1016/j.ejor.2021.12.050
9. Morrison DR, Sauppe JJ, Sewell EC, Jacobson SH. A wide branching strategy for the graph coloring problem. *INFORMS J Comput.* 2014;26:704–717. https://www.doi.org/10.1287/ijoc.2014.0593
10. Kolesar PJ. A branch and bound algorithm for the knapsack problem. *Manag Sci.* 1967;13(9):723–735. https://www.doi.org/10.1287/mnsc.13.9.723
11. Fischetti M, Toth P, Vigo D. A branch-and-bound algorithm for the capacitated vehicle routing problem on directed graphs. *Oper Res.* 1994;42:846–859. https://www.doi.org/10.1287/opre.42.5.846
12. Dantzig GB, Ramser JH. The truck dispatching problem. *Manag Sci.* 1959;6:80–91. https://www.doi.org/10.1287/mnsc.6.1.80
13. Toth P, Vigo D. The Vehicle Routing Problem. Philadelphia; 2001. https://www.doi.org/10.1137/1.9780898718515
14. Golden B, Wang X, Wasil E. The Evolution of the Vehicle Routing Problem : A Survey of VRP Research and Practice from 2005 to 2022. Springer Cham; 2023.
15. Achuthan NR, Caccetta L. Integer linear programming formulation for a vehicle routing problem. *Eur J Oper Res.* 1991;52(1):86-89. https://www.doi.org/10.1016/0377-2217(91)90338-V
16. Bula GA, Gonzalez FA, Prodhon C, Afsar HM, Velasco NM. Mixed Integer Linear Programming Model for Vehicle Routing Problem for Hazardous Materials Transportation. *IFAC-PapersOnLine.* 2016;49(12):538-543. https://www.doi.org/10.1016/j.ifacol.2016.07.691
17. Madankumar S, Rajendran C. A mixed integer linear programming model for the vehicle routing problem with simultaneous delivery and pickup by heterogeneous vehicles, and constrained by time windows. Sādhanā. 2019;44:39. https://www.doi.org/10.1007/s12046-018-1048-y
18. Florio AM, Hartl RF, Minner S, Salazar-González J-S. A Branch-and-Price Algorithm for the Vehicle Routing Problem with Stochastic Demands and Probabilistic Duration Constraints. *Transp Sci.* 2020;55(1):1-17. https://www.doi.org/10.1287/trsc.2020.1002
19. Yang W, Ke L, Wang DZW, Lam JSL. A branch-price-and-cut algorithm for the vehicle routing problem with release and due dates. *Transp Res Part E Logist Transp Rev.* 2021;145:102167. https://www.doi.org/10.1016/j.tre.2020.102167
20. Nafstad GM, Desaulniers G, Stålhane M. Branch-Price-and-Cut for the Electric Vehicle Routing Problem with Heterogeneous Recharging Technologies and Nonlinear Recharging Functions. *Transp Sci.* 2025;59(3):628-646. https://www.doi.org/10.1287/trsc.2024.0725
21. Cordeau JF, Gendreau M, Hertz A, Laporte G, Sormany JS. New Heuristics for the Vehicle Routing Problem. In: Langevin, A., Riopel, D. (eds) Logistics Systems: Design and Optimization. Springer, Boston, MA; 2005. https://www.doi.org/10.1007/0-387-24977-X9
22. Ahmed ZH, Yousefikhoshbakht M. An improved tabu search algorithm for solving heterogeneous fixed fleet open vehicle routing problem with time windows. *Alex Eng J.* 2023;64(1):349-363. https://www.doi.org/10.1016/j.aej.2022.09.008
23. Voigt S. A review and ranking of operators in adaptive large neighborhood search for vehicle routing problems. *Eur J Oper Res.* 2015;322(2):357-375. https://www.doi.org/10.1016/j.ejor.2024.05.033
24. French AP, Robinson AC, Wilson JM. Using a Hybrid Genetic-Algorithm/Branch and Bound Approach to Solve Feasibility and Optimization Integer Programming Problems. *J Heuristics.* 2001;7:551–564. https://www.doi.org/10.1023/A:1011921025322
25. Bertacco L, Fischetti M, Lodi A. A feasibility pump heuristic for general mixed-integer problems. *Discrete Optim.* 2007;4(1)63-76. https://www.doi.org/10.1016/j.disopt.2006.10.001
26. Fischetti M, Glover F, Lodi A. The feasibility pump. *Math Program.* 2005;104:91–104. https://www.doi.org/10.1007/s10107-004-0570-3
27. Conforti M, Cornuejols G, Zambelli G. Integer Programming. Switzerland; 2014. https://www.doi.org/10.1007/978-3-319-11008-0
28. Praveen V, Keerthika Dr.P, Sivapriya G, Sarankumar A, Bhasker B. Vehicle Routing Optimization Problem: A Study on Capacitated Vehicle Routing Problem. *Mater Today Proc.* 2022;64(1):670-674. https://www.doi.org/10.1016/j.matpr.2022.05.185
29. Kandakoglu A, Saure A, Michalowski W, Aquino M, Graham J, McCormick B. A decision support system for home dialysis visit scheduling and nurse routing. *Decis Support Syst.* 2020;130. https://www.doi.org/10.1016/j.dss.2019.113224
30. Kulkarni RV, Brave PR. Integer programming formulations of vehicle routing problems. *Eur J Oper Res.* 1985;20(1):58-67. https://www.doi.org/10.1016/0377-2217(85)90284-X
31. Golden B, Raghavan S, Wasil E. The Vehicle Routing Problem: Latest Advances and New Challenges. Springer New York, NY; 2010. https://www.doi.org/10.1007/978-0-387-77778-8
32. Pereira FB, Tavares J. Bio-inspired Algorithms for the Vehicle Routing Problem. Springer Berlin, Heidelberg; 2010. https://www.doi.org/10.1007/978-3-540-85152-3

33. Korte B, Vygen J. Combinatorial Optimization: Theory and Algorithm. Germany; 2000. https://www.doi.org/10.1007/978-3-662-56039-6
34. Lin S. Computer solutions of the travelling salesman problem. *Bell Syst Tech J.* 1965;44:2245–2269. https://www.doi.org/10.1002/j.1538-7305.1965.tb04146.x
35. Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to Algorithms. 4th edition. The MIT Press Cambridge, Massachusetts London, England; 2022. https://www.amazon.com/Introduction-Algorithms-fourth-Thomas-Cormen
36. Denardo EV. Dynamic programming: models and applications. Prentice-Hall. Inc; 2003.
37. Art L, Mauch H. Dynamic Programming: A computational tools. Springer Berlin, Heidelberg; 2006. https://www.doi.org/10.1007/978-3-540-37014-7
38. Bellman R. Dynamic Programming. New Jersey, USA; 1957. https://www.doi.org/10.2307/j.ctv1nxcw0f
39. Solomon M. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Oper Res.* 1987;35(2):254–265. https://www.doi.org/10.1287/opre.35.2.254
40. Zhou Y, Hao J-K, Xiao M, Jin Y. An effective branch-and-bound algorithm for the maximum s-bundle problem. *Eur J Oper Res.* 2022;297(1):27–39. https://www.doi.org/10.1016/j.ejor.2021.05.001

Asri Bekti Pratiwi *is a doctoral student in the Department of Mathematics, Faculty Mathematics and Natural Sciences, Universitas Gadjah Mada. She is full time lecturer at Department Mathematics, Faculty of Science and Technology, Universitas Airlangga. She is part of the Operations Research and Computation research group.*
https://orcid.org/0000-0002-8881-3377

Salmah Salmah *is a Full Professor in the Department of Mathematics, Faculty Mathematics and Natural Sciences, Universitas Gadjah Mada in the Applied Mathematics Research Group. Her expertise is in the field of dynamic game, and optimization.*
https://orcid.org/0000-0002-5774-4321

Irwan Endrayanto *is a full time lecturer at Department Mathematics, Faculty Mathematics and Natural Sciences, Universitas Gadjah Mada. His topic interest is in Stochastic Operations Research, Applications of Queueing Theory, Discrete and Stochastic Resource Allocation, Simulation etc. He is actively involved in developing mathematical model for optimizing resource allocation and logistics design in healthcare operations.*
https://orcid.org/0000-0002-4422-7712

An International Journal of Optimization and Control: Theories & Applications
(https://accscience.com/journal/ijocta)