TRANSPORT
ISSN 1648-4142 / eISSN 1648-3480
2016 Volume 31(4): 389–407
doi:10.3846/16484142.2013.819814

A HYBRID COLUMN GENERATION ALGORITHM BASED ON METAHEURISTIC OPTIMIZATION

Wenbin Hu, Bo Du, Ye Wu, Huangle Liang, Chao Peng, Qi Hu

School of Computer, Wuhan University, Wuhan City, Hubei Province, China

Submitted 15 October 2012; accepted 3 April 2013;
first published online 25 October 2013

Abstract. The exact solution and heuristic solution have their own strengths and weaknesses on solving the Vehicle Routing Problems with Time Windows (VRPTW). This paper proposes a hybrid Column Generation Algorithm with Metaheuristic Optimization (CGAMO) to overcome their weaknesses. Firstly, a Modified Labelling Algorithm (MLA) in the sub-problem of path searching is analysed. And a search strategy in CGAMO based on the demand of sub-problem is proposed to improve the searching efficiency. While putting the paths found in the sub-problem into the main problems of CGAMO, the iterations may fall into endless loops. To avoid this problem and keep the main problems in a reasonable size, two conditions on saving the old paths in the main problem are used. These conditions enlarge the number of constraints considered in the iterations to strengthen the limits of dual variables. Through analysing the sub-problem, we can find many useless paths that have no effect on the objective function. Secondly, in order to reduce the number of useless paths and improve the efficiency, this paper proposes a heuristic optimization strategy of CGAMO for dual variables. It is supposed to accelerate the solving speed from the view of on the dual problem. Finally, extensive experiments show that CGAMO achieves a better performance than other state-of-the-art methods on solving VRPTW. The comparative experiments also present the parameters sensitivity analysis, including the different effects of MLA in the different path selection strategies, the characteristics and the applicable scopes of the two path-keeping conditions in the main problem.

Keywords: vehicle routing problems with time windows; column generation algorithm; labelling algorithm; sub-problem; simplex method; dual problem.

Introduction

Nowadays, with the increasing demand of better services and tighter requirements for the users, the Vehicle Routing Problem (VRP) has played a very important role in the logistics and supply chain management and many other areas. The VRP is defined as a problem of minimizing the total travel distance of a number of vehicles with some constraints, including every customer must be served at least once by a vehicle. Among many kinds of VRP, the Vehicle Routing Problem with Time Windows (VRPTW) has been widely studied and occupies an important place in the field of operations research and combinatorial optimization (Hashimoto *et al.* 2008; Lau *et al.* 2003; Teodorovic *et al.* 1995; Vaidyanathan *et al.* 2007). The VRPTW has a wide range of applications such as supermarket distribution, expressage, bank deliveries, postal deliveries, school bus routing and so no. The VRPTW has become a subject of great interest in VRP fields.

Scholars mainly concentrate on two kinds of algorithms: exact and heuristic algorithms. Exact algorithms are solved by precise mathematical models and reasoning, and can get the optimal solution. But since the VRPTW is NP-hard, solving large-scale problems using the exact algorithm is very time-consuming (Achuthan *et al.* 2003; Kohl 1995; Laporte *et al.* 1992; Christiansen, Lysgaard 2007; Gutiérrez-Jarpa *et al.* 2009; Cheung, Hang 2003). The gradual optimization of the heuristic algorithm for feasible solutions is promising, but when the problem is of a smaller scale, the solution presents the large deviations (Sungur *et al.* 2008; Chakroborty, Mandal 2005; Hiquebran *et al.* 1993; Li *et al.* 2010;

Corresponding author: Wenbin Hu
E-mail: *hwb@whu.edu.cn*

Taylor & Francis
Taylor & Francis Group

Cheng, Wang 2009; Gajpal, Abad 2009; Balseiro *et al.* 2011). Since both kinds of algorithms have their own strengths and weaknesses for solving VRPTW, combining these two kinds of algorithms to attain even better solution for VRPTW becomes a treading research direction (Tavakkoli-Moghaddam *et al.* 2011; Hashimoto *et al.* 2008; Garcia-Najera, Bullinaria 2011; Hong 2012). A good combined algorithm should have the least time consuming and the smallest solution deviations.

Dantzig and Wolfe (1960) proposed the Column Generation (CG) method to solve the linear programs with decomposable structures. CG has been applied successfully to solve many optimization problems and becomes a leading optimization technology to solve VRPTW (Rousseau *et al.* 2007). However, original CG often shows a very slow convergence speed, partly because of heavy degeneracy problems. The convergence becomes even slower when the multiple dual solutions are associated with each primal solution. The dual solution is the crucial part of CG to solve VRPTW.

This paper focuses on the combination of CG and heuristic algorithms, and proposes a hybrid column generation algorithm based on metaheuristic optimization (CGAMO). It establishes a mixed-integer programming model. Dantzig–Wolfe (D–W) is used to divide the problem into a partitioning set problem and a shortest path problem with resource constraints (Dantzig, Wolfe 1960). CGAMO is finally used to get the optimal solution. The main contributions are as follows:

1) Labelling Algorithm (Desrochers, Soumis 1988) is used to search the paths. According to VRPTW, a modified labelling algorithm (MLA) is proposed to satisfy the path total demands of selecting and extending policies for path extensive searching. MLA can reduce the iteration number and memory space in solving VRPTW.
2) A method with the conditionally retained history path of the main problem is proposed. It can effectively avoid endless loops because of the height degradation of VRPTW and the limitations of the sub-problem.
3) The metaheuristic optimization policy of the dual variables is employed in order to upgrade the CGAMO's solving ability. It selects an appropriate optimization step and a local optimal value to search the optimal value of the dual variables. In this way, the phenomenon of endless loops in the process of CG iteration is avoided and the corresponding optimization speed is obviously accelerated.

The rest of this paper is organized as follows. Section 1 describes the related works and typical solving algorithms on VRPTW. Section 2 describes the VRPTW model and the decomposition methods. Section 3 analyses the VRPTW solving steps by CGAMO. It also analyses three key technologies in CGAMO in detail. Section 4 analyses the performance of CGAMO and compares it with other state-of-the-art algorithms. Final Sections draw a conclusion and show some perspectives.

1. Related Works

To solve VRPTW problems, many methods have been proposed, mainly divided into the exact methods and the heuristic methods. As for the exact methods, they include branch and bound method (Laporte *et al.* 1986), integer programming (Foster, Ryan 1976), tree search (Christofides *et al.* 1981), cutting plane method (Gomory 1958), branch and cut method (Padberg, Rinaldi 1987) and CG algorithm (Dantzig, Wolfe 1960) and so on. The exact solution-based methods mathematically formulate VRPTW. Laporte *et al.* (1992) proposed a branch and bound algorithm for VRPTW. Christiansen and Lysgaard (2007) presented a branch-and-price algorithm for the capacitated VRPTW with stochastic demands. Because VRPTW is a NP-hard problem (Savelsbergh 1985), when the size of the problem is large, it cannot be solved within an acceptable time. Desrochers *et al.* (1992) proposed an exact approach to solve a VRPTW, and found that it was inefficient on time costs. Kolen *et al.* (1987) proposed a branch and bound method to solve VRPTW with the node number ranging from 6 to 15. He found that when the node number of VRPTW was 6, the computer (VAX11/785) took nearly one minute to find the solution; and when the node number of VRPTW reached 12, the computer was unable to solve the VRPTW. Through reviewing the literature, the exact methods can solve the small scale VRPTW very well. But they are poor for solving the large scale VRPTW.

In order to solve the above problem, many scholars studied different heuristic algorithms to solve VRPTW. The typical heuristic algorithms are local searching (Zachariadis, Kiranoudis 2010; Hashimoto *et al.* 2008), Simulated Annealing (SA) (Tavakkoli-Moghaddam *et al.* 2011), Genetic Algorithm (GA) (Ghoseiri, Ghannadpour 2010; Cheng, Wang 2009), Tabu Search (TS) (Li *et al.* 2010; Lau *et al.* 2003), ant system (Reimann *et al.* 2004; Balseiro *et al.* 2011; Vaidyanathan *et al.* 2007), Large Neighbour Searching (LNS) algorithm (Hong 2012), Particle Swarm Optimization Algorithm (PSOA) (Chen *et al.* 2006; Ai, Kachitvichyanukul 2009), etc. Heuristic-based methods are classified into the constructed heuristic algorithms and the smart optimization algorithms. The smart optimization algorithms derive from the simulation and the nature learning, simulation optimization on the existing viable solutions. But due to the large difference between the real problem and the ideal model, initial solutions are difficult to construct, and the convergence speed in early stage is slow with the local optimal solutions.

Both exact methods and heuristic methods have their own weaknesses. Combining these two kinds of algorithms to solve VRPTW is a promising choice. Many researchers used the integer programming to transfer VRPTW into a set partition problem (Mautor, Naudin 2007; Lorenz, Raz 2001; Dumitrescu, Boland 2003; Sellmann *et al.* 2007). But since the path space tends to be huge, it is difficult to find such path. The common integer programming can't fit the problem scale. However,

CG algorithm has a good ability to fit the problem scale, so it has been widely used. After using CG to decompose VRPTW, the solution is actually a subprocess to find the basic restrictions to the shortest path, and the main problem to find the optimal integer solution (Feillet *et al.* 2004; Irnich, Villeneuve 2006; Zhu, Wilhelm 2012; Qureshi *et al.* 2009; Jepsen *et al.* 2008). CG algorithm is constructed on the exact solution of the integer programming model. It expands the solution range based on a guaranteed accuracy. But due to the inherent defects and VRPTW's characteristics, there are still some non-negligible problems:

1) when the time windows between the client nodes vary greatly, a distributed aggregation phenomenon problem in the paths of the sub-problem causes the objective function converge slowly;
2) since VRPTW is degenerated obviously, there are many invalid paths with the zero increment, which hardly improves the objective function;
3) due to the tailing effect of original CG in VRPTW, the resulting basic variables is worse, so the improvement of the objective function decreases and the time cost sharply increases with the increment of the iteration times.

To solve the above defects, this article focuses on Vehicle Routing Problem with Hard Time Window (VRPHTW) that is it cannot be allowed to serve for the customer after the required time. A path selecting method based on the total needs is proposed. The purpose is to extend the path search range, and enlarge the number of valid paths in the results. Besides, in order to avoid the endless loop phenomenon, a method with conditionally preserved main problem in the historic path is also proposed. The two different preserving conditions are set to compare the different effects. A dual variable of the heuristic optimization policy is also used to accelerate the convergence speed of the dual variable, and reduce the number of the variable repetition endless loops.

2. VRPTW Model

2.1. Model Formulation

In general, VRPTW mainly consists of several factors including client the node, the distribution centre and the road in the road network. It involves some constraints including the vehicle's maximum carrying capacity, the time window of clients' available time and the vehicle travelling circuit. The target of VRPTW is to make an overall minimum cost of the distribution; here it is the shortest travelling distance of the vehicles.

In VRPTW, the vehicle's starting point and client nodes distribute discretely in a two-dimensional space and two neighbour nodes with the same vehicle's service tying to each other. Therefore, the VRPTW model can be built using a directed graph $G = (V, A)$, where V represents the set $C = \{1,2,3,\cdots,n\}$ which is a set of the vehicle's starting point and client nodes. Arc set A contains all of $arc(i,j)$ which link each two nodes that are valid in the time, where $i,\ j \in V$.

Symbols used in the model are defined as following:

d_i – the cargo demand of client node i;
q – the vehicle's maximum cargo capacity, not considering the differences between vehicles;
s_{jk} – the time when vehicle k reaches client node j and starts the service, when $j = 0$, it represents the time when vehicle k returns to the starting point;
t_{ij} – the time required for the vehicle travelling from node i to node j, including the vehicle service time at node i;
c_{ij} – the consuming value of vehicles travelling from node i to the node j, which is proportional to the travelling time;
x_{ijk} – whether vehicle k travels through $arc(i, j)$, when $x_{ijk} = 1$, it indicates that vehicle k travels through $arc(i, j)$, otherwise $x_{ijk} = 0$;
$[a_i, b_i]$ – the time window of client node i.

Constraints of the vehicle service process in the model are as following:

1) each client node only needs one vehicle to service, namely that the in-degree and out-degree of any client node is 1 for the same vehicle (as Formula (2));
2) the vehicle's capacity has an upper limitation (as Formula (3));
3) the vehicle route is a loop that contains the starting point, and each client node's in-degree and out-degree for the same vehicle are equal, except that both of them are 1 for the starting point (as Formula (4) to (6));
4) the constraint for a client node's time window, when the vehicle arrives at the node in advance, it needs to wait for the client until the client node can accept service, and the vehicle can't be late (as Formula (4) to (6)).

Based on the above constraints, the sum of all vehicles' distribution cost makes up the objective function, and the mixed integer programming model of VRPTW can be built as follows:

$$\min \sum_{k\in K}\sum_{(i,j)\in A} c_{ij}x_{ijk}; \tag{1}$$

$$\sum_{k\in K}\sum_{j\in C} x_{0jk} = 1,\ \forall i \in C; \tag{2}$$

$$\sum_{i\in C} d_i \sum_{j\in V} x_{ijk} \le q,\ \forall k \in K; \tag{3}$$

$$\sum_{j\in C} x_{0jk} = 1,\ \forall k \in K; \tag{4}$$

$$\sum_{i\in V} x_{ihk} - \sum_{j\in V} x_{hjk} = 0,\ \forall h \in C,\ \forall k \in K; \tag{5}$$

$$\sum_{i\in C} x_{i0k} = 1,\ \forall k \in K; \tag{6}$$

$$x_{ijk}\left(s_{ik} + t_{ij} - s_{jk}\right) \le 0,\ \forall (i,j) \in A,\ \forall k \in K; \tag{7}$$

$$a_i \le s_{ik} \le b_i,\ \forall i \in V,\ \forall k \in K; \tag{8}$$

$$x_{ijk} \in \{0,1\},\ \forall (i,j) \in A,\ \forall k \in K. \tag{9}$$

Formula (7) expresses that when vehicle k travels through $arc(i,j)\left(x_{ijk}=1\right)$, the vehicle's arrival time and travelling time satisfies the inequalities $s_{ik}+t_{ij}\le s_{jk}$, $\forall(i,j)\in A,\ \forall k\in K$, which means that the vehicle travelling through node i should not reach node j before $s_{ik}+t_{ij}$. Due to the presence of time windows, the vehicle needs to wait until the client node can accept service when it arrives at the client node in advance. The model set the upper limit of the service vehicles, namely the size of set K. When $|K|$ is large enough $(\ge n)$, the upper limited number of the vehicles does not exist.

2.2. Model Decomposition and Relaxation

In this paper, the constraints related with the vehicle route will be extracted as a sub-problem by the D–W decomposition method, and the constraints of the main problem will be made by retaining the single service limit of client node in the problem. Sub-problem is the basically shortest path problem with the resource constraints, and it searches the shortest vehicle travelling path satisfying the Formula (2) to (8), while the main problem is the set partition problem and it uses a feasible path founded in the sub-problem to classify the client node set. The main problem remains linear after its decomposition and can be solved by the simplex method (Nelder, Mead 1965). The sub-problem with nonlinear constraints of the time window can be solved by dynamic planning method. They are shown in Formula (10) to (12):

$$\min\sum_{p\in P}c_p y_p; \tag{10}$$

$$\sum_{p\in P}a_{ip}y_p=1,\forall i\in C; \tag{11}$$

$$y_p\in\{0,1\},\forall p\in P, \tag{12}$$

where: p represents the feasible path of the vehicle service node; P is the set of all feasible paths; c_p in the objective function represents the consuming value of path p; y_p represents the path selection, and when y_p = 1 it means that there is a vehicle to complete distribution, otherwise y_p = 0; a_{ip} represents the times of path p visiting to the client node i.

In order to improve the stability of the main problem solution, Formula (13) will replace Formula (11), the set partition problem will be relaxed as the Set Covering (SC) problem, and a non-negative constraint will be added to the dual variables:

$$\sum_{p\in P}a_{ip}y_p\ge 1,\forall i\in C. \tag{13}$$

2.3. Sub-Problem Model

After the decomposition of the VRPTW model, the sub-problem uses the test number of the paths in the main problem as objective function, and it needs to find the feasible path with the smallest test number. The test number of the paths can be provided by subtracting the dual variable corresponding to the visited node from travelling cost c_p. Since the vehicle can just serve each node once, the path consuming value should subtract the corresponding dual variable when the vehicle goes through each node. Therefore, the obtained travelling cost is the test number of the path in the main problem when the travelling cost is modified by the dual variable according to Formula (14):

$$\bar{c}_{ij}=c_{ij}-u_j,\forall j\in V, \tag{14}$$

where: $\bar{c}_{ij}$ is the modified cost value; c_{ij} is the cost value of vehicles travelling from node i to node j, which is proportional to the travelling time; u_j is the starting point and client node set of the vehicle route is $C=\{1,2,3,\cdots,n\}$.

The sub-problem is simplified as the problem of searching the feasible path by modifying the travelling cost of the arc. The isolated sub-model is shown as Formula (15) to (22):

$$\min\sum_{(i,j)\in A}\bar{c}_{ij}x_{ij}; \tag{15}$$

$$\sum_{i\in C}d_i\sum_{j\in V}x_{ij}\le q; \tag{16}$$

$$\sum_{j\in V}x_{0j}=1; \tag{17}$$

$$\sum_{i\in V}x_{ih}-\sum_{j\in V}x_{hj}=0,\ \forall h\in C; \tag{18}$$

$$\sum_{i\in V}x_{i0}=1; \tag{19}$$

$$s_i+t_{ij}-s_j\le\left(1-x_{ij}\right)M_{ij},\ \forall(i,j)\in A; \tag{20}$$

$$a_i\le s_i\le b_i,\ \forall i\in V; \tag{21}$$

$$x_{ij}\in\{0,1\},\ \forall(i,j)\in A, \tag{22}$$

where: M_{ij} is the order of service when x_{ij} is 0 or 1. In order to initialize it, we use the basic path that each vehicle serves one node as an initial solution and its consuming value is taken as the dual variable of the corresponding node. Since the initialization does not limit the number of the vehicles, the dual variable of the start point u_0 is 0.

3. CGAMO for VRPTW

3.1. CGAMO

When the scale of VRPTW is large, we must use an efficient method to find the solution of the main problem after decomposition. Simplex method and Interior Point (IP) method (Karmarkar 1984) are all well-known to solve the large-scale liner programming problems. They are all suitably used to find the solution of the main problem after decomposition. When they get close to the local optimal solution, the ending condition of simplex method is easy to be determined; IP method doesn't perform well in this aspect. In the coding process, IP method is more complicated than simplex method. It has to do the matrix inversions and logarithmic calculations and it cannot guarantee the precision when calculating the decimals. Although IP method has a lower complexity in theory, simplex method has a quicker processing speed for most of the problems when the prob-

lem scales are not very big. So we can use CG algorithm to decompose the problem and solve the sub-problems and the main problem by iterative solution and simplex method. In CG algorithm, some base intake variables obtained in each iteration for the sub-problem will be substituted into the main problem. When the problems degenerate, a part of variables are searched many times and get into base repeatedly due to the limitation of the sub-problem solution. While the degradation situation is serious or the sub-problem performance is poor, it is possible to fall into endless loops.

In order to avoid the endless loops, we can conditionally save the historical paths of the main problem and strictly restrict the dual variable when adding the search result into the main problem. In order to reduce the variables repeatedly and get them into the base, we apply metaheuristic optimization to the dual variable in the iterations to accelerate the speed of converging.

Algorithm I: CGAMO for VRPTW

Input: client node information, the number of vehicles, the maximum cargo capacity, cost matrix.

Output: optimal travelling route set:

1) Initialization, use the route cost for one vehicle serving one client node as the dual variable of the corresponding node, the dual value of the starting point is 0;
2) Use the dual variable to modify the consuming value of the arc in the cost matrix;
3) Solve the sub-problem (see Section 3.2), and get a group of paths whose test numbers are negative. If you can't find these paths, then go to step 7;
4) Add the paths found in the sub-problem to the main problem, conditionally saving the original paths of the main problem (see Section 3.3);
5) Solve the main problem, and update the dual variables;
6) Optimize the dual variables by the metaheuristic function (see Section 3.4), go to step 2;
7) If the current optimal solution is an integer solution, then output it; otherwise, use the branch division to obtain the optimal integer solution according to the largest branch weight of the arcs.

3.2. Sub-Problem Solution

3.2.1. Expansion Search of the Path

In order to find the base intake path with a smaller test number and control the scale of the main problem and reduce the storage space required by the algorithm, this paper uses MLA to find a part or all of *Pareto* optimal paths and eliminate the poor paths. The main data structures used in the MLA are as following:

1) $Label = \{pr\ \ L), t(L), q(L), c(L), Label, no\}$, used to represent the state when the vehicle arrives the current client node *pr*, including the service start time *t*, the total demand *q* of client nodes which have been served, the travelling cost *c* when reaching the current node. We mark the label with a unique serial number *no*.
2) $Path = \{vis, Label\ list, length\}$, the vector *vis* records the next node which can be extended by the path, and it can avoid repeatedly visiting client node in the path search, wherein 1 means the next node that can be extended and 0 means the node that can't be reached. In the path extension, the vehicles' expansion range is limited by the visiting time of the current node and the total demand of the path. By the path expanding, the range of the extensible nodes get smaller and smaller , so *vis* doesn't increase, but decrease instead. Label list is used to save the label of each client node's vehicle visiting. *lenght* is used to record the number of accessible nodes. The information of accessible nodes is stored in the path instead of the label, which can avoid storing the visiting state of the non-current nodes' label. When the poor paths are eliminated, the storage space is filled with some labels which are no longer cited by the path, thus it can reduce the storing consumption.
3) Path queue *Q*, used to store the paths which have not yet been extended.
4) Path set $U(v)$, used to store the paths which can reach the specified nodes and have been extended. It is used to eliminate the selected route.

MLA is an extended search algorithm for the graph traversal. The main steps are shown as follows.

Algorithm II: MLA

Input: client node information, the number of the vehicles, the maximum cargo capacity, cost matrix.

Output: a certain number of paths with negative test number:

1) Initialize the path queue *Q*, add p_0 to *Q*, p_0 is a path beginning from the starting point and has only one label of the point 0, $U(v) = \Phi$, Φ expresses the empty set;
2) Path selection: find path p with the shortest travelling time from *Q*, recording node v and time $t(p)$ when it reaches node v;
3) Path extraction: extract the paths from *Q* which can reach node v and consume less than $t(p) + \min(t_{ij})$ to build the path set $P(v)$, and then delete the paths in $P(v)$ from *Q*;
4) Path elimination: eliminate the paths in $P(v)$ using all of the paths in $U(v) \bigcup P(v)$, delete the poor paths in $P(v)$ according to the elimination rules, add the paths to $U(v)$ which are in $P(v)$ and not eliminated;
5) Path expansion: extend each path in $P(v)$ to all accessible nodes and obtain a set of paths $S(v)$;
6) For the path in $S(v)$, if it doesn't return to the starting point, it will be added to queue *Q*, otherwise, judging whether its cost is less than 0, if so, add it to $U(0)$;
7) If the number of the paths in $U(0)$ reaches or exceeds the set limit, exit; otherwise, go to step 2 to continue to find and expand the paths.

In the expanding search of the path, every time we select a group of paths with the same ending point in

order to speed up the search and reduce the branches to be searched. Since the path extension involves time, demand and cost, each of these three resources can be regarded as the basis of path selecting:

1) According to the nodes of the path whose traveling time is the shortest, choose the time of the shortest arc in cost matrix as the selected width;
2) According to the node of the path whose current demand is the smallest, choose the minimum cargo demand as the selected width;
3) According to the nodes of the path whose current cost is the smallest, choose the traveling cost of the shortest arc in cost matrix as the selected width.

Since time and demand are monotonically increasing in the path expansion, time and demand-based path selection and expansion are actually similar to the breadth-first search. The nodes in the found paths are relatively homogeneously distributed, and there are many effective paths getting into the base. But because its breadth-first feature, the length of the path found by the algorithm in the early stages is shorter, and the convergence speed of vehicles' number is relatively slow. While selection and expansion based on consuming value are the depth-first search based on greedy strategy, it is able to find the longer and accessible paths. But the driving costs are not monotonic increasing, so the found paths are often gathered, and the similarity degree between paths is high. As a result, the number of effective paths which can improve the objective function may be relatively less than the breadth-first search.

3.2.2. Path Elimination Rule

When several different paths are selected, the pruning operation can be done in the expansion trees of the paths. Then the part of apparently poor paths and their extensions will be eliminated by comparing the extended scope and the costs of the paths. For two paths p_1 and p_2 with the same current nodes, first of all, compare the consuming values (Formula (23)), and to estimate the merits degree of the path. Then compare time (Formula (24)), the total demand (Formula (25)) and the range of the extensible subsequent nodes of the path (Formula (26)). Only when both the consuming time and the demand of the path are small, we can guarantee that it can do a longer extension. The expansion path of $p_1$contains all expansion paths of p_2 and the costs of the extended paths in p_1 are smaller when both p_1 and p_2 satisfy the following requirements. Then p_2 can be deleted to decrease the path search range:

$$c(p_1) \le c(p_2); \tag{23}$$

$$t(p_1) \le t(p_2); \tag{24}$$

$$q(p_1) \le q(p_2); \tag{25}$$

$$vis(p_1) \le vis(p_2). \tag{26}$$

3.3. The Connection Between the Sub-Problem and the Main Problem

During the process of solving the VRPTW degradation problem by simplex method, there is always an item with the value of zero based on the feasible solution. The increment of the base intake variable may be 0, and the base intake variable does not change the objective function. The convergence of the algorithm stop, and what's worse, the base intake variables enter the loop state. VRPTW is a seriously degraded problem, where path search in the sub-problem using CG algorithm only considers the size of the test number without involving the size of the increment getting into the base, thus it may find an invalid variable whose increment value is 0 and reduce solving efficiency. When the search results of the sub-problem are all the invalid paths, the iterative solution does not optimize the objective function; repeating invalid iterations may lead to an endless loop.

During the iteration, the dual variable is continuously restricted by the corresponding constraints of the base intake variable. The algorithm will converge to the optimal solution at last. If all of the variables in a few iterations are considered at the same time, we can get the dual variable satisfying all the constraints and avoid circulating into the base. But saving all the found paths will increase the scale of the main problem and take up more storage space. This is contrary to the purpose of using CG. Therefore, we need to set the proper conditions of saving the historical paths to control the scale of the main problem. Due to the trailing effect generated by the column, the closer the objective function is to the optimal solution, the smaller the increment of the found base intake variables is, most of which are 0. The more the times of the iteration is, the more the occurrence possibility of circulating into the base is, and the more the need to save the historical paths of the main problem is. Thus two judgment conditions of saving the historical paths should be constructed:

1) When the objective function comes to Convergence Stagnation (CS), which means that the solution of the main problem does not optimize the objective function, the increments of the base intake variable in the sub-problems are 0. The solution of the original problem is also unchanged, only the dual variables are changed, and it is likely to be added into the iteration. At this time, saving the historical paths can reduce the delay time of the objective function.
2) When the sub-problem becomes Global Search (GS) in the process of searching path, the number of the found paths is less than the specified limit. The found paths where the nodes are relatively homogeneously distributed contain the path with the smallest test number and the dual variables are strongly restricted. The upper limit of the found paths number in the sub-problem affects the algorithm greatly in determining whether becomes the global search or not.

3.4. Metaheuristic of the Dual Problem

3.4.1. Heuristic Function for Optimization

During the process of solving linear programming by the simplex method, the base's feasible solution of the original problem will change with the base intake operation. Due to the constraints of corresponding hyper plane restrictions and the dual variables gradually converging to the optimal solution, the dual problem will still converge to its optimal solution even when the iteration convergence of original problem begins to stagnation. The change of the base variables is continuous and the dual variable smoothly converges to the optimal solution in the simplex method. While in the CG algorithm, because the decomposition of the original problem separates the selection of the base intake variables from the base intake operation, only part of variables are considered during iteration. The invalid variables may repeatedly get into the base, which results in that the convergence of dual variable to the optimal solution is no longer smooth but is constantly oscillating near the optimal solution instead. So the convergence of the dual problem solution is relatively slow, and in the worst case, it may become an endless loop (Lübbecke, Desrosiers 2005; Nazareth 1988).

We subjectively set the value of objective function Z^* for the dual problem. When the convergence of the original problem begins to stop, the dual variable value remains in the corresponding hyper plane of the objective function. It is shown in Formula (27):

$$\sum_{i \in V} u_i = Z^*. \tag{27}$$

Large-scale linear programming problems can be solved through a number of iterations as the CG algorithm applies to the simplex method and the principle of locality. But the locality not only lessens the optimization range of the objective function but also reduce the algorithm's speed. This phenomenon becomes more and more obvious as objective function is gradually approaching to the optimal value. Therefore, we can apply an appropriate method to optimize the dual problem to reduce the oscillation amplitude of the dual variables and to accelerate the algorithm's convergence.

This paper adopts the function shown in Formula (28) to do metaheuristic optimization for the dual variable:

$$u'_m = w_m u_m + \left(1 - w_m\right) u_{m-1}^{best}, \tag{28}$$

where: u'_m represents the dual variables used for the next solving for sub-problem; u_m represents the dual variables obtained by the current (m-th) iteration; u_{m-1}^{best} is a relatively better dual variable obtained in the former $m-1$ iterations, that it is the local optimal value of the dual variable; w_m is a parameter, and $0 < w_m < 1$. The above formulas can be expressed as Formula (29):

$$u'_m = u_{m-1}^{best} + w_m \left(u_m - u_{m-1}^{best}\right). \tag{29}$$

After iteration, the dual variable chooses the current local optimal value as the starting point and moves forward to the dual variables obtained in this iteration actually. Each forward step is w_m. w_m is an important parameter for metaheuristic optimization, and its value may be a fixed value or a dynamic value according to different iterations. In general, we set a lower limitation greater than zero for w_m in order to ensure that the current dual variables are involved in the optimization and avoid the local optimization convergence of the problem.

3.4.2. The Selection of the Local Optimal Solution

During the based intake or out-take operation of the original problem, the dual variable is driven by the corresponding constraints and continuously approximate to the optimal solution. The optimal value of the dual variable satisfies all the constraints and matches the optimal value of the objective function. During the solution, the objective function gradually decreases, the constraints for the dual variables gradually increases, and the dual variables do not get worse generally. So we can simply select the dual variables obtained in the latest iteration as the current optimal value, which is shown as Formula (30):

$$u_{m-1}^{best} = u_{m-1}. \tag{30}$$

In the degradation problem, dual variable will oscillate around the optimal solution in the corresponding hyper plane of the objective function when the original problem stops the optimization with the increment of the base intake variable being zero. Dual variable occurs to be cyclic when the original problem loops into the base. When the dual variable oscillates around the optimal solution, we can employ the average of the dual variables obtained in the recent several iterations to estimate the location of the optimal dual variables; it is shown as Formula (31):

$$u_{m-1}^{best} = \frac{1}{m - lo} \sum\nolimits_{i=lo}^{m-1} u_i, \tag{31}$$

where: *lo* (last optimized) represents iterations in the latest optimization, and m is the current iterations.

We can use the average to estimate the approximate location of the oscillation centre of the dual variables and to accelerate the convergence of the dual variables. Meanwhile, a simple arithmetic average can guarantee that the optimum value is still in the corresponding hyper plane of the objective function.

3.4.3. The Calculation of Heuristic Function Parameter

Applying heuristic function to optimize the dual problem, the dual variable starts from the optimal solution, and the current obtained dual variable steps forward w_m each time. In order optimize the dual problem continuously; our method imposes some constraints on each step shown in Formula (32):

$$const \le w_m \le 1, \tag{32}$$

where: *const* is a constant greater than zero, indicating the lower limit of step w_m. It's used to ensure that the dual variable obtained in the current iteration can participate in the optimization process rather than stay at the

local optimal solution. The upper limit 1 avoids the dual variable's variation due to the excessive optimization.

Since the dual problem is oscillating near the optimal solution and continuously approximates to the optimal solution in iterations, the form of the oscillation is attenuated, and the obtained dual variable will be better each time. When u_m and u_{m-1}^{best} are in different sides of the optimal solution, the optimized step less than 1 can obtain an optimized value u'_m, which is closer to the optimal solution, and it can accelerate the convergence of the dual variable. When the step size is larger than 1, u'_m is much farer away from the optimal solution and its value may be worse than u_m, which may make the dual variable degrade. Therefore, w_m has the upper limit 1. Considering the different oscillations of the dual variables, the specific values of w_m can be divided into the following situations:

1) The constant *const* is fixed on the lower limit, in the case of continuous damp oscillation. A fixed reduction of the step can accelerate the oscillation's attenuation and make the dual variable reach the optimal solution faster;
2) The function with the lower limit of *const*, such as $w_m = \max\left\{const, \frac{lo}{m}\right\}$. When the objective function is convergent, the optimized dual variable should be close to the iteration's result. The convergence is in stagnation. The longer the stagnation is, the worse the qualities of local optimal results are.

4. Experimental Results and Analysis

This section is devoted to the performance evaluation of CGAMO for VRPTW. The adopted benchmarks and experimental conditions will be described in Section 4.1. Section 4.2 describes the comparison results of CGAMO with other state-of-the-art methods. Section 4.3 details the performance of CGAMO for path searching in VRPTW. Section 4.4 presents the performance of CGAMO for avoiding endless loops in resolving the VRPTW. Section 4.5 analyses the performance of CGAMO for metaheuristic of dual problem in VRPTW.

4.1. Benchmark Description and Experimental Conditions

In order to compare our algorithms with other approaches on VRPTW, the basic data of our testing problems adopt Solomon's benchmark (Vehicle Routing Problems… 2012). The Solomon's benchmark contains 56 instance each with a size of $N = 100$. These instances are categorised as C1 and C2, where customers are located in geographical clusters, R1 and R2, and the customers are randomly distributed, and RC1 and RC2, which have a mix of random locations and clusters. Different types of problems differ in the distribution of the nodes, the service time of each node, and the width of time windows. The Solomon's benchmark did not define the computing method of travelling cost and time for VRPTW. We defines the travelling cost of VRPTW as Formula (33), and set (x_i, y_i), (x_j, y_j) as coordinates of node i and j:

$$c_{ij} = \frac{\left\lfloor 10\sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2} \right\rfloor}{10}, \tag{33}$$

where: $\lfloor . \rfloor$ expresses the integer part of real number a. The travelling time can be computed by $t_{ij} = c_{ij} + st_i$. st_i expresses the service time of customer node i, and can be obtain from the Solomon's benchmark problems.

Experiments are carried out under the configuration of *Windows Vista TM Home Premium*, with *AMD Turion (tm) 64 X2 Mobile Technology TL-62 2.10 GHz* and *2GB RAM*. We realize the CGAMO on platform MyEclipse 6.6. The solution of main problem invokes the linprog function of *Matlab 7.0*. The initial solution of VRPTW is that each vehicle services a node. The upper bound routing searching number of R1 is set as 500, and RC1 and C1 are both set as 1000. The connection of sub-problem and main-problem reserves the history route of main-problem in the condition of CS.

4.2. Comparison Results and Analysis

Our comparison experiment consists of two parts. First, the computational efficiency of the proposed approach CGAMO was tested, and compared with other state-of-the-art methods. Then, the path searching computational efficiency of the proposed approach MLA in CGAMO was tested, and compared with a powerful optimization software used in solving integer programming problems – CPLEX (*http://www.cplex.com*) and an Extended Label-Setting Algorithm (EMLSA) (Zhu, Wilhelm 2012).

4.2.1. Computational Efficiency Analysis

Comparison Results of CGAMO with the Original CG and IP Method

In order to extend the range of the path search and enlarge the number of valid paths in CGAMO, so that a path selecting method is proposed. Besides, in order to avoid the endless loop phenomenon, a method with conditionally preserved main problem in the historic path is also proposed. A dual variable of heuristic optimization policy is also used to accelerate the convergence of the dual variable, and reduce the number of variable repetitions to the base. To verify the results, this section carries out extensive experiments to compare the CGAMO with the original CG and IP method. The results are shown in Table 1. Table 1 compare the performance produced by the proposed CGAMO, the original CG and IP method in the set C1, R1 and RC1. The Runtime in the Table 1 presents the solving time of problems by CGAMO, CG or IP method. The Rate in Table 1 shows the ration of different values of distance between the best known result and CGAMO, CG and IP method results. If the Rate value is negative, it means that the Distance value of CGAMO, CG or IP method is shorter than the best known results. If the Rate value is a positive number, it means that the Distance value

Table 1. Comparison of the performance of the CGAMO, CG and IP method

Instance	Best known results (Tavakkoli-Moghaddam *et al.* 2011)		CGAMO				CG				IP method			
	Vehicle	Distance	Vehicle	Distance	CPU [s]	Rate [%]	Vehicle	Distance	CPU [s]	Rate [%]	Vehicle	Distance	CPU [s]	Rate [%]
R101	19	1645.79	19	1633.51	185.69	–0.75	19	1896.36	255.63	15.23	19	1689.59	184.97	2.66
R102	17	1486.12	16	1471.96	165.39	–1.00	18	1678.64	241.68	12.96	16	1496.38	157.96	0.69
R103	13	1292.68	12	1276.99	142.95	–1.23	12	1396.97	219.55	8.07	13	1296.31	164.35	0.28
R104	9	1007.24	9	1007.31	138.54	0.00	10	1084.32	200.13	7.65	9	1007.31	142.68	0.01
R105	14	1377.11	14	1377.11	165.23	0.00	14	1489.63	236.39	8.17	15	1377.11	169.68	0.00
R106	12	1251.98	12	1251.98	151.11	0.00	12	1296.21	218.74	3.53	12	1328.97	159.61	6.15
R107	10	1104.66	10	1104.66	135.24	0.00	10	1154.25	200.15	4.49	10	1174.54	141.01	6.33
R108	9	960.88	9	960.88	124.17	0.00	9	996.35	185.68	3.69	10	960.88	132.10	0.00
R109	11	1194.73	11	1194.73	138.99	0.00	11	1301.21	201.97	8.91	11	1204.36	140.14	0.81
R110	10	1118.59	10	1118.59	134.65	0.00	10	1196.39	218.67	6.96	10	1186.96	138.74	6.11
R111	10	1096.72	10	1096.72	138.96	0.00	10	1152.32	222.32	5.07	10	1102.94	142.03	0.57
R112	9	982.14	9	982.14	142.32	0.00	9	1021.35	196.39	3.99	9	982.14	152.31	0.00
C101	10	828.94	10	828.94	75.96	0.00	10	828.94	135.96	0.00	10	828.94	96.32	0.00
C102	10	828.94	10	828.94	78.65	0.00	10	841.20	140.29	1.48	10	828.94	91.21	0.00
C103	10	828.06	10	828.06	87.45	0.00	10	828.06	163.21	0.00	10	828.06	92.38	0.00
C104	10	824.78	10	824.78	84.25	0.00	10	824.78	164.17	0.00	10	824.78	100.21	0.00
C105	10	828.94	10	828.94	81.20	0.00	10	828.94	158.24	0.00	10	828.94	106.32	0.00
C106	10	828.94	10	828.94	79.69	0.00	10	828.94	171.01	0.00	10	828.94	94.21	0.00
C107	10	828.94	10	828.94	86.32	0.00	10	828.94	159.68	0.00	10	828.94	104.65	0.00
C108	10	828.94	10	828.94	84.14	0.00	10	828.94	167.96	0.00	10	828.94	100.01	0.00
C109	10	828.94	10	828.94	81.28	0.00	10	828.94	164.17	0.00	10	828.94	100.65	0.00
RC101	14	1696.94	14	1670.98	204.36	–1.55	15	1821.39	305.14	7.33	14	1721.97	187.98	1.48
RC102	12	1554.75	12	1446.08	186.39	–7.52	13	1596.32	296.37	2.68	12	1574.98	169.68	1.30
RC103	11	1261.67	11	1261.67	158.65	0.00	12	1301.27	268.34	3.14	11	1287.64	165.24	2.06
RC104	10	1135.48	10	1135.48	154.21	0.00	11	1193.97	253.19	5.15	10	1164.87	164.98	2.59
RC105	13	1629.44	13	1549.69	197.14	–5.15	13	1963.21	319.67	20.51	13	1687.96	197.01	3.59
RC106	11	1424.73	11	1412.39	178.65	–0.87	11	1542.12	286.39	8.24	11	1475.41	169.63	3.56
RC107	11	1230.48	11	1226.39	149.69	–0.33	11	1354.69	268.97	10.09	11	1424.73	154.32	15.79
RC108	10	1139.82	10	1142.97	136.55	0.28	10	1197.85	235.96	5.09	10	1139.82	142.65	0.00

of CGAMO, CG or IP method is bigger than the best known results. Based on these results in Table 1, we can make the following conclusions on the performance of CGAMO, CG and IP method on solving the VRPTW:

1) CGAMO can get very promising result, and sometimes it gets even better result than the best known results, such as for problems R101, R102, R103, RC101, RC102, RC105, RC106, RC107 and RC108. Meanwhile, basic CG and IP method can hardly converge to the best known results for almost all the problems of set R1 and RC1. Besides, CGAMO has better performance than basic CG;
2) CGAMO converges to the best known results at nearly 2÷3 minutes for R1, one minute for C1 and at most three minutes for RC1. CG converges to its best results at least 3÷4 minutes for R1, 2÷3 minutes for C1 and 5÷6 minutes for RC1. IP method needs the similar time to converge to the best known results with CGAMO in set RC1 and R1, but IP method needs more time than CGAMO in set C. We can see that CGAMO can effectively accelerate the searching in solution space and the performance coverage speed of the CGAMO is higher than the original CG and IP method.

Comparison Result of CGAMO with Other State-of-the-Art Methods:

In order to compare the efficiency of the different algorithms on VRPTW, GPGA (Goal Programming with Genetic Algorithm) (Ghoseiri, Ghannadpour 2010), LNS (Hong 2012) and ACO (Ant Colony Optimization) (Balseiro *et al.* 2011) are used to compare with CGAMO. The results of comparison experiments of VRPTW are shown in Tables 2–4. The rate in Tables 2–4 expresses the ratio of the distance between CGAMO and other typical methods. If the Rate value is negative, it means that the distance value of CGAMO is shorter than the other typical methods. If the Rate value is positive, it means that the Distance value of CGAMO is bigger than the other typical methods. Based on these results of Tables 2–4, we can make the following conclusions on the effectiveness of the CGAMO approach for solving VRPTW:

1) The solutions quality computed by CGAMO can well approximate to the best known results. If the travel distances and the number of used vehicles are taken into account properly, the CGAMO almost can get the best results of C1, R1 and RC1;
2) Compared with the results generated by GPGA, LNS, ACO, the results of CGAMO keeps better than them. The average CPU (Central Processing Unit) time of CGAMO is less than 3 minutes. The results are hard to systematically compare with other typical methods because there are big differences in the run environments.

Table 2. Comparison results of different methods for VRPTW of R1

Instance	CGAMO			Best known results (Tavakkoli-Moghaddam *et al.* 2011)			GPGA			LNS			ACO		
	Vehicle	Distance	CPU [s]	Vehicle	Distance	Rate [%]	Vehicle	Distance	Rate [%]	Vehicle	Distance	Rate [%]	Vehicle	Distance	Rate [%]
R101	19	1633.51	185.69	19	1645.79	–0.75	19	1677.00	–2.66	18	1612.29	1.30	19	1650.80	–1.06
R102	16	1471.96	165.39	17	1486.12	–1.00	18	1511.80	–2.71	16	1473.41	–0.10	17	1486.12	–0.96
R103	12	1276.99	142.95	13	1292.68	–1.23	14	1287.00	–0.78	12	1279.37	–0.19	13	1292.68	–1.23
R104	9	1007.31	138.54	9	1007.24	0.00	10	974.24	3.28	10	1025.47	–1.80	9	1007.31	0.00
R105	14	1377.11	165.23	14	1377.11	0.00	15	1424.60	–3.55	14	1377.95	0.00	14	1377.11	0.00
R106	12	1251.98	151.11	12	1251.98	0.00	13	1270.30	–1.46	12	1276.48	–1.96	12	1252.03	0.00
R107	10	1104.66	135.24	10	1104.66	0.00	11	1108.80	–0.38	11	1153.86	–4.45	10	1104.66	0.00
R108	9	960.88	124.17	9	960.88	0.00	10	971.91	–1.15	10	990.82	–3.12	9	960.88	0.00
R109	11	1194.73	138.99	11	1194.73	0.00	12	1212.30	–1.47	12	1179.73	1.26	11	1194.73	0.00
R110	10	1118.59	134.65	10	1118.59	0.00	12	1156.5	–3.39	11	1113.10	0.49	10	1118.84	0.00
R111	10	1096.72	138.96	10	1096.72	0.00	11	1111.9	–1.38	11	1155.39	–5.35	10	1096.73	0.00
R112	9	982.14	142.32	9	982.14	0.00	10	1036.9	–5.58	10	981.46	0.07	9	985.28	0.32

Table 3. Comparison results of different methods for VRPTW of C1

Instance	CGAMO			Best known results (Tavakkoli-Moghaddam *et al.* 2011)			GPGA			LNS			ACO		
	Vehicle	Distance	CPU [s]	Vehicle	Distance	Rate [%]	Vehicle	Distance	Rate [%]	Vehicle	Distance	Rate [%]	Vehicle	Distance	Rate [%]
C101	10	828.94	75.96	10	828.94	0.00	10	828.94	0.00	10	828.94	0.00	10	828.94	0.00
C102	10	828.94	78.65	10	828.94	0.00	10	828.94	0.00	10	828.94	0.00	10	828.94	0.00
C103	10	828.06	87.45	10	828.06	0.00	10	828.06	0.00	10	839.37	–1.37	10	828.06	0.00
C104	10	824.78	84.25	10	824.78	0.00	10	824.78	0.00	10	838.98	–1.72	10	824.78	0.00
C105	10	828.94	81.20	10	828.94	0.00	10	828.94	0.00	10	828.94	0.00	10	828.94	0.00
C106	10	828.94	79.69	10	828.94	0.00	10	828.94	0.00	10	842.10	–1.59	10	828.94	0.00
C107	10	828.94	86.32	10	828.94	0.00	10	828.94	0.00	10	828.94	0.00	10	828.94	0.00
C108	10	828.94	84.14	10	828.94	0.00	10	828.94	0.00	10	832.74	–0.46	10	828.94	0.00
C109	10	828.94	81.28	10	828.94	0.00	10	828.94	0.00	10	828.94	0.00	10	828.94	0.00

Table 4. Comparison results of different methods for VRPTW of RC1

Instance	CGAMO			Best known results (Tavakkoli-Moghaddam *et al.* 2011)			GPGA			LNS			ACO		
	Vehicle	Distance	CPU [s]	Vehicle	Distance	Rate [%]	Vehicle	Distance	Rate [%]	Vehicle	Distance	Rate [%]	Vehicle	Distance	Rate [%]
RC101	14	1670.98	204.36	14	1696.94	–1.55	15	1690.60	–1.17	15	1671.54	–0.03	14	1696.94	–1.55
RC102	12	1446.08	186.39	12	1554.75	–7.52	14	1509.40	–4.38	13	1447.14	–0.07	12	1554.75	–7.52
RC103	11	1261.67	158.65	11	1261.67	0.00	12	1331.80	–5.56	11	1313.79	–4.13	11	1262.02	–0.03
RC104	10	1135.48	154.21	10	1135.48	0.00	11	1177.20	–3.67	11	1163.54	–2.47	10	1135.48	0.00
RC105	13	1549.69	197.14	13	1629.44	–5.15	15	1611.50	–3.99	13	1502.48	3.05	13	1629.44	–5.15
RC106	11	1412.39	178.65	11	1424.73	–0.87	13	1437.60	–1.79	12	1406.25	0.44	11	1424.73	–0.87
RC107	11	1226.39	149.69	11	1230.48	–0.33	11	1222.10	0.35	11	1278.96	–4.29	11	1230.48	–0.33
RC108	10	1142.97	136.55	10	1139.82	0.28	11	1156.50	–1.18	11	1172.83	–2.61	10	1139.82	0.28

4.2.2. Path Searching Computational Efficiency Analysis

In order to compare the efficiency of the different methods to path searching for VRPTW solution, CPLEX and a Three-Stage Approach (TSA) (Zhu, Wilhelm 2012) are used to compare with MLA proposed in this paper. The results of comparison experiments of VRPTW are shown in Table 5. Table 5 reveals the average, minimal and maximum runtime time of path searching of MLA and other typical methods. The rate in Table 5 expresses the ratio of runtime time between CGAMO and other typical methods. If the Rate value is negative, it means that the average runtime times of MLA in CGAMO are less than the other typical methods. Based on these results in Table 5, we can make the following conclusions on the effectiveness of the MLA approach in CGAMO for solving path searching of VRPTW:

1) The path searching speed of MLA in CGAMO is similar to TSA method for R1. In C1 and RC1, the average Rates of TSA is almost 20%, which proves that the MLA has obviously better performance than TSA for path searching in C1 and RC1;
2) Compared with CPLEX, the path searching speed of MLA in CGAMO is far better than the other methods for R1, C1 and RC1, and the average rates of CPLEX is more than 80%.

4.3. Performance of CGAMO for Path Searching

In Section 3.2.2, we presented three kinds of path selecting methods based on resources. In order to compare the efficiency of MLA for these three selecting methods, we use R1 as experiment case. The experiment compares the total number of searching path and iteration of MLA

Table 5. Comparison results of different methods for path searching in VRPTW (runtime [s])

Instance	MLA			TSA				CPLEX			
	Average	Min	Max	Average	Min	Max	Rate [%]	Average	Min	Max	Rate [%]
R101	0.41	0.29	0.65	0.44	0.25	0.74	–7.32	0.84	0.76	0.98	–104.88
R102	0.37	0.25	0.61	0.35	0.21	0.59	5.41	0.76	0.68	0.89	–78.38
R103	0.31	0.21	0.57	0.32	0.22	0.39	–3.23	0.64	0.59	0.75	–106.45
R104	0.29	0.21	0.51	0.33	0.19	0.45	–13.79	0.56	0.48	0.68	–93.10
R105	0.35	0.23	0.59	0.41	0.29	0.77	17.14	0.67	0.57	0.72	–91.43
R106	0.30	0.20	0.61	0.34	0.25	0.65	–16.67	0.59	0.44	0.66	–96.67
C101	0.15	0.09	0.24	0.19	0.12	0.29	–20.00	0.24	0.17	0.31	–60.00
C102	0.16	0.09	0.23	0.22	0.12	0.33	–37.50	0.33	0.24	0.43	–106.25
C103	0.19	0.11	0.31	0.18	0.08	0.28	5.26	0.35	0.24	0.45	–84.21
C104	0.22	0.12	0.38	0.25	0.14	0.49	–14.29	0.39	0.29	0.51	–77.27
C105	0.21	0.10	0.34	0.25	0.12	0.39	–19.05	0.37	0.26	0.49	–76.19
RC101	0.51	0.44	0.79	0.66	0.55	0.87	–29.41	0.92	0.79	1.11	–80.39
RC102	0.49	0.41	0.72	0.64	0.54	0.84	–30.61	0.87	0.75	1.02	–77.55
RC103	0.32	0.19	0.43	0.45	0.23	0.76	–40.63	0.57	0.49	0.71	–78.13
RC104	0.30	0.14	0.45	0.36	0.18	0.57	–20.00	0.52	0.43	0.68	–73.33
RC105	0.45	0.34	0.65	0.55	0.36	0.69	–22.22	0.83	0.72	0.99	–84.44

in the condition of these three selecting methods. To avoid the interference of the other factors, we set the upper number bound of the searching path in the sub-problems to 500 and keep the main problem in the condition of CS (Lübbecke, Desrosiers 2005). As the sizes of different problems are different, we use the minimum time as the reference to compare the relative number on different conditions. The results are shown in Fig. 1.

Based on the results of Fig. 1, we can make the following conclusions on the effectiveness of three kinds of path searching methods:

1) The methods based on the minimum demand perform better than the methods based on the minimum time in path searching and iteration. The reason is that the range of the total demand in the path extension is unified, and the extension based on the demands is closer to the breadth-first strategy. Because there are time windows in client notes, the notes selected in the extended search and the starting time of the time windows are all different. The results of the methods based on the minimum time are relatively worse;
2) When we use the methods based on the minimum cost, the paths that are not Pareto optimal paths are relatively less than other two methods, because the costs are not monotonically varying. When the negative cost arcs become a little more after the price is set, the method based on the minimum cost is like the depth-first strategy, so the total number of path searching and iteration are bigger than the other two methods.

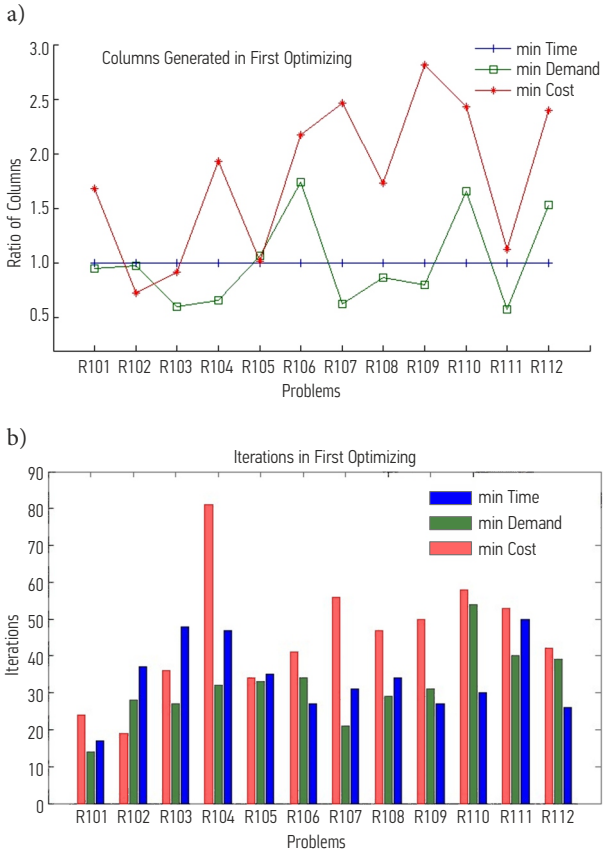

Fig. 1. Total number of searching path and iteration number in the condition of different path selecting methods: a – the total number of searching number; b – the number of iteration

4.4. Performance of CGAMO for Avoiding Endless Loops

When the paths found in sub-problems are included into the main problem, to avoid the endless loops caused by the problematic degenerate and the lack of path searched, we keep the paths in the former iteration conditionally to accelerate the iteration. In Section 3.3, we present two judgment conditions of preserving the historical path. This section we analyse the impact on the optimization of the CS and GS when the numbers of paths searching procedures are different in the sub-problems by comparing the total path searching, the number of iterations and the maximum number of paths in the main problem. At the same time, we compare the optimization performance of CS and GS on the same condition. The less the number of path searching procedures in sub-problems is, the more effective the solution is. The less the iterations are, the quicker the calculating speed is. The decreasing maximum vehicles in main-problem cause the less resource needed to solve main-problem. And the size of the solvable algorithm becomes bigger. We also use 500 as the search number to analyse the relative number of path searching (Lübbecke, Desrosiers 2005) with ours. The comparison of total path searching number under CS and GS with different searching number is shown in Fig. 2. Based on these results of Fig. 2, we can draw the following conclusions on the effectiveness of path searching under CS and GS with different searching number:

1) Under the CS condition, the impacts on the different problems caused by the different number of path searching procedures are different, because the state of road network and path distribution are different in different problems, and the proportion of the total number of path searching changes within the range of 0.4 to 2.2.
2) Under the GS condition, the smaller the number of path searching is, the larger the number of path searching would be. And the less the upper number of global search is, the larger the actual number of path searching would be. When the upper number of path searching is too small, the actual total number of path searching may have an explosive growth. For example, when the number of path searching is 300, the total number of path searching will be expanded more than 8 times. And when the number of path searching is much smaller, there may be a iteration endless loops.
3) Overall, the GS is more easily to be affected by the upper bound of the number of path searching set by the sub-problems. In the comparison of CS and GS that the history information will be kept, the GS performs better when the size

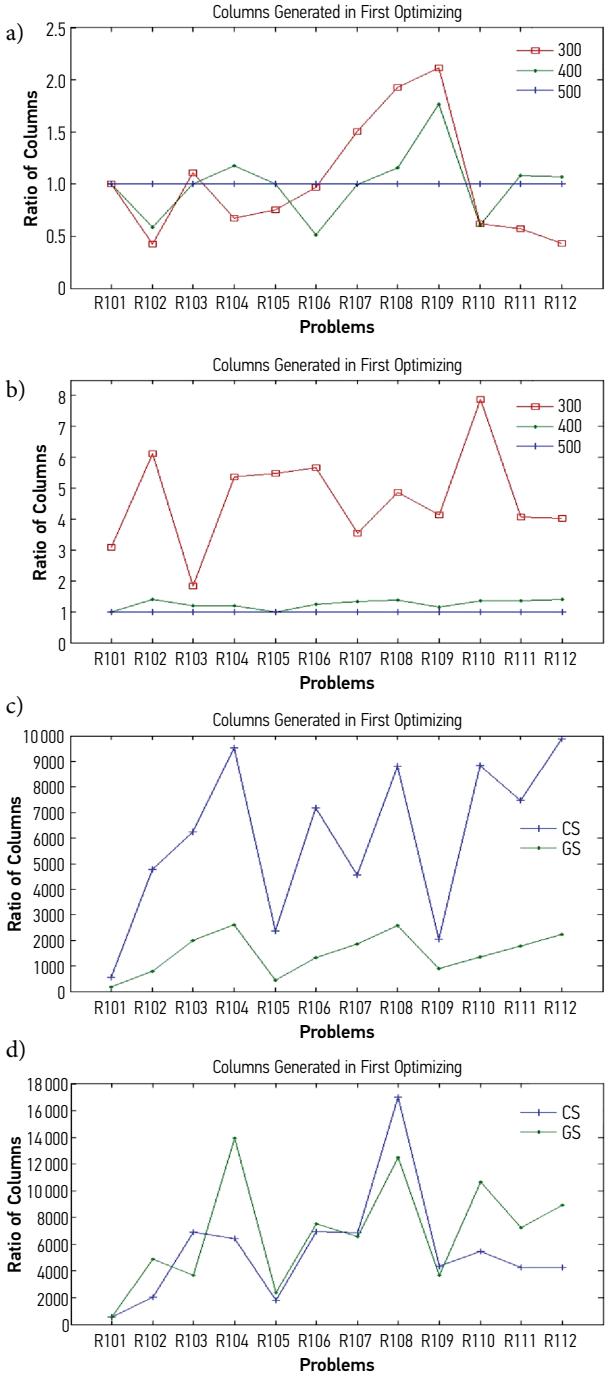

Fig. 2. Comparison of total number of path searching under different condition and searching number: a – total number of vehicle under CS; b – total number of vehicle under GS; c – vehicle number under searching time 500; d – vehicle number under searching time 300

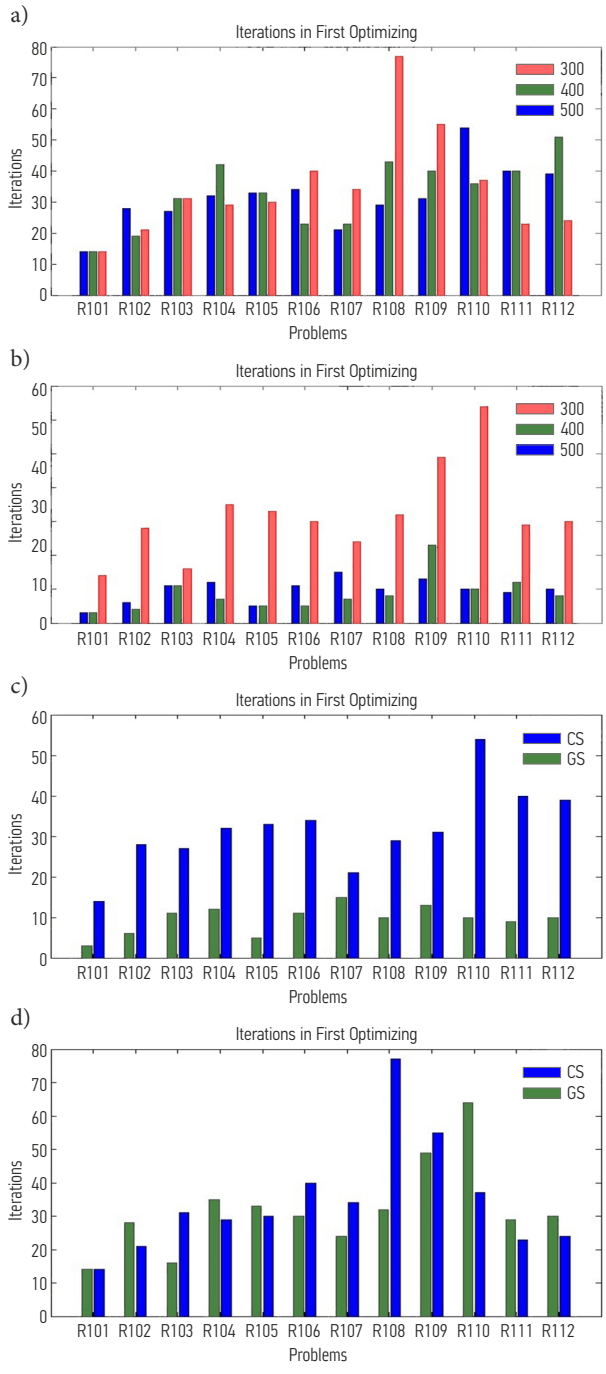

Fig. 3. Comparison of iteration number under different condition and searching number: a – iteration number under CS; b – iteration number under GS; c – iteration number under searching time 500; d – iteration number under searching time 300

of sub-problems is bigger. But GS is sensitive to the data when the number of searching becomes smaller. The total numbers of paths are all slightly different.

The comparison of the iteration number under CS and GS with different searching number is shown in Fig. 3. Based on these results of Fig. 3, we can make the conclusion that CS is not affected so much by the searching number, yet GS is affected sharply. When the upper number of path searching is big, the actual number of iterations will be relative small, but when the number of path searching becomes smaller, the difference of the number of iteration under CS and GS is smaller.

The comparison of the main-problem scale under CS and GS with different searching numbers is shown in Fig. 4. Based on these results of Fig. 4, we can make the

following conclusions on the effectiveness of the main-problem scale under CS and GS with different searching numbers:

1) Under the CS condition, the size of the main-problem will be smaller when the upper bound of the path searching number is smaller. When the upper bound is too small, the size of main-problem may increase. The size of main-problem will be larger and larger under GS condition. Similarly, under the GS condition, the size of main-problem is not big when the size of path searching is large. The difference between GS and CS is not big when the size of path searching is small, and the CS performs better;
2) GS performs better on the number of path searching procedure and the number of iteration than CS when the size of path searching defined as a large number (500). GS and CS perform analogously when the size of search is small (300), while the main-problem scale under GS is larger. In the experiment, an iteration endless loop occurs in most of the problems under GS condition, but the convergence and optimal solution under CS condition can be obtained.

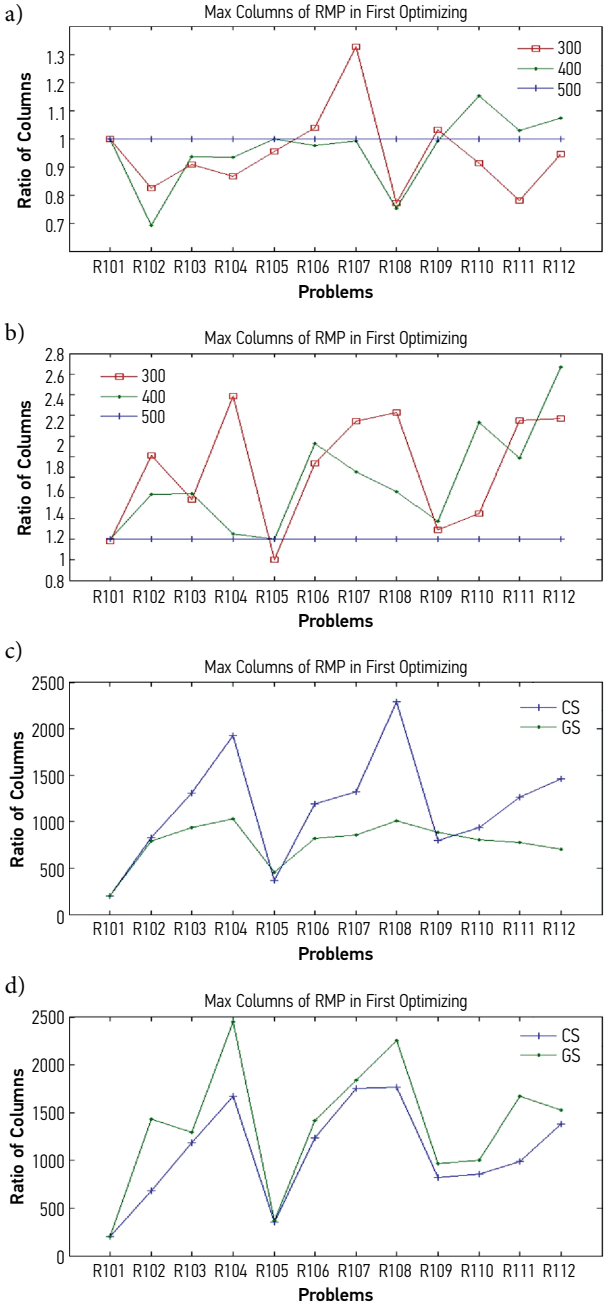

Fig. 4. Comparison of main-problem scale under different condition and searching number: a – main-problem scale under CS; b – main-problem scale under GS; c – main-problem scale under searching time 500; d – main-problem scale under searching time 300

Then, we come to the conclusions as following by the comparing the total number of path searching procedures, the number of iterations and the main-problem scale under the CS and GS with different upper bounds of searching path number:

1) The CS is steadier when the upper bound of path searched in sub-problems changes;
2) The GS performs better on the speed of calculating, the efficiency and the main-problem scale when the number of path searching is big enough;
3) The CS can guarantee an optimal solution when the number of path searching is very small, while the GS may raise an iteration endless loop.

According to the conclusions above, if we can set the number of path searching procedures to be large in the sub-problems in the solving VRPTW, the history path can be kept in the global searching, the iteration cycling can be avoided effectively, the speed and the efficiency of calculating will be higher, and the resource consumed by solving of the main problem will be less. If the proper number of path searching cannot be estimated, or the resource for solving the main-problem is not enough, the CS method is steadier and can guarantee an optimal solution.

4.5. Performance of CGAMO for Metaheuristic of Dual Problem

In Section 3.4, we have presented the metaheuristic optimization of dual problem of VRPTW as Formula (28) to (32). In order to evaluate the performance of CGAMO for metaheuristic of dual problem, we adapt four different kinds of heuristic methods to optimize the solution of VRPTW. The four kinds of heuristic methods are listed in the Table 6.

In the process of VRPTW, the constant *const* can be set as 0.4, 0.5, 0.6, 0.7 and 0.8. We make 20 kinds of heuristic functions by different combination with 4 kinds of heuristic methods in the Table 5. According to the experiments, we get the best and the worst optimization methods and their parameters, which are showed in Table 7. The first two columns of Table 7 represent the serial number of problems and the number of iterations without optimization, columns 3÷5 represent the minimum number of iteration and its heuristic function, columns 6÷8 represent the worst optimization, the column 9 represents the average number of iterations needed by

Table 6. Parameter combinations of four type's heuristic optimization

Optimal method	w_k	u_{k-1}^{best}
1	const	u_{k-1}
2	const	$\frac{1}{k-lo}\sum_{i=lo}^{k-1}u_i$
3	$\max\left\{const,\frac{lo}{k}\right\}$	u_{k-1}
4	$\max\left\{const,\frac{lo}{k}\right\}$	$\frac{1}{k-lo}\sum_{i=lo}^{k-1}u_i$

the 20 heuristic functions, the column 10 represents the number of schemes for reducing the iterations. Based on these results of Table 6, we can make the following conclusions on the effectiveness of different heuristic methods for the dual problem in VRPTW:

1) The optimization effects of one heuristic function are different in different problems. The proper heuristic methods and the corresponding parameters can make a large decrease of the iterations;
2) On the contrary, the improper parameters for a problem may make a low convergence speed of the problem and an increase of the iterations. In the 12 problems of Table 7, most of the problems have a half or nearly a half of optimization by the 20 heuristic optimization functions, except a few ones.

In order to compare the performances of different heuristic methods for dual problem in VRPTW, we carry out the experiments to compare the total numbers of path searching by different heuristic methods in Table 6. The experiments results are shown in Fig. 5. Based on these results of Fig. 5, we can make the following conclusions on the effectiveness of different heuristic methods for the dual problem in VRPTW:

1) In R101, the efficiency of the problem solution decreases when the *const* is a little bigger. In R102, the efficiency of 3rd optimization is low when the *const* is a little smaller. Among R103~R106 and R110~R112, the optimization method with the fixed step size is better, and the method of vibrational step size performs worse. But the effects in R107 and R109 are opposite, and the method with varying step sizes in this situation is better. In R07, the optimization is effected easily by a constant parameter, which may be too large or too small. In R08, the optimizations of all functions are effected much by the change of parameter;
2) On the aspect of optimization functions, there are respectively 13, 9, 18, 16 kinds of situation for the 4 kinds of heuristic functions with a total of 12 problem and 5 kinds of parameter values. Then we can say that the functions with a fixed step size is better than the functions with a varying step size, and the way with the average dual variable value at the stop of convergence for the estimate of the local optimal solution is better than the way of recording the last solution;
3) Considering the optimization of all the problems in Fig. 5, the solution process of the problem is improved with the most functions and parameter, and the optimization performance of the different problems in different ways are different. A decrease of efficiency and speed also exit for a part of the ways of optimization. So the heuristic function with a 0.6 or 0.7 step size and using the methods of using the average dual variable value at the stop of convergence as the local optimal solution can increase the efficiency of the problem solution.

Table 7. Results of different heuristic optimization

Instance	Its.	Best optimal solutions			Worst optimal solutions			Average its.	Better solutions
		Its.	Const.	Method	Its.	Const.	Method		
R101	14	10	0.6	2	24	0.6	3	14.8	11
R102	28	14	0.8	2	34	0.6	1	22.05	15
R103	27	16	0.4	1	47	0.6	3	32.85	4
R104	34	18	0.4	1	44	0.8	1	34.1	8
R105	33	18	0.8	1	56	0.4, 0.5	4	33.45	12
R106	32	19	0.8	1	34	0.6	1	25.8	17
R107	23	17	0.7	2	42	0.4	1	24.35	11
R108	29	15	0.5	4	61	0.4	2	32.6	9
R109	31	13	0.6	4	41	0.5	2	23.65	17
R110	69	23	0.8	1	79	0.4	2	40.65	19
R111	40	15	0.4	2	51	0.7	3	32.55	18
R112	41	18	0.7	2	42	0.6	1	30.75	15

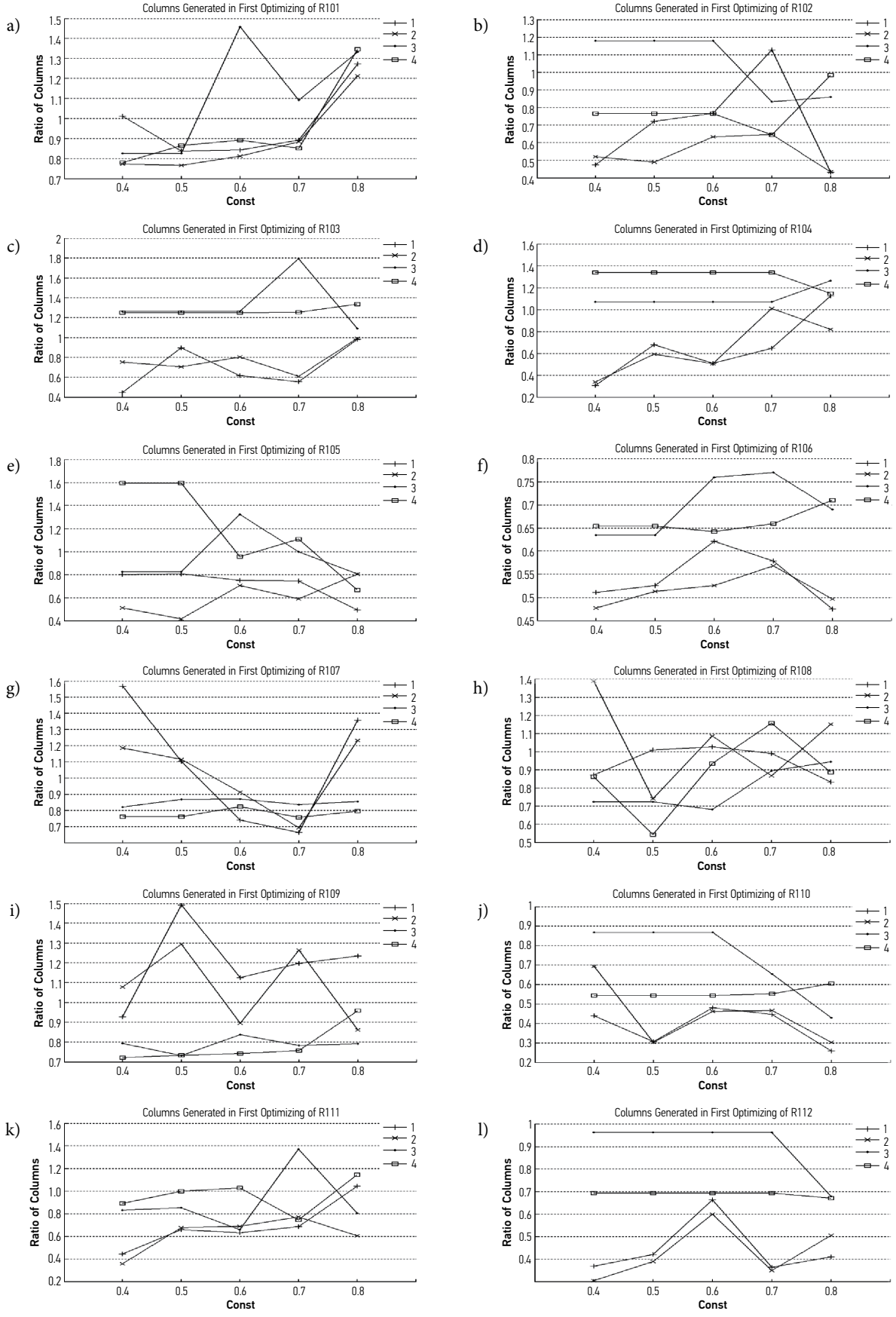

Fig. 5. Comparison of path searching number for R1: a – R101; b – R102; c – R103; d – R104; e – R105; f – R106; g – R107; h – R108; i – R109; j – R110; k – R111; l – R112

Conclusions

State-of-the-art VRPTW algorithms are usually susceptible to the scale of the VRPTW problem. They would present a high computation burden and the local optimal solutions on a large dataset. In order to alleviate the above problems, this paper proposes a hybrid method by combining exact and heuristic algorithms. It constructs a hybrid CGAMO algorithm to solve VRPTW. CGAMO uses a MLA to search the sub-problem. In order to improve the path searching efficiency, it uses a new search strategy based on the sub-problem demand. CGAMO adopts two conditions on reserving the old paths in the main problem to avoid iterations endless loops and keep the main problems in a reasonable size, and it also uses a new heuristic optimization strategy for dual variable.

The experiments mainly analyse the comparison of CGAMO with the original CG and other state-of-the-art methods, including the performance of MLA of CGAMO for path searching in VRPTW, the performance of CS and GS condition and metaheuristic optimization in CGAMO for dual problem. From the extensive experiments, it concludes that:

1) CGAMO can effectively accelerate the searching in solution space and the CGAMO has a better performance coverage speed than the original CG and other state-of-the-art methods;
2) MLA in CGAMO shows an approximate path searching speed with TSA method, and it is far better than CPLEX method. The method based on the minimum demand in MLA performs better than the method based on the minimum time in path searching and iteration. When we use the method based on the minimum cost, the paths that are not Pareto optimal paths are relatively less than other two methods, because the cost is not monotonic changing;
3) The CS is steadier than GS, but GS performs better on the speed of calculating, the efficiency and the main-problem scale than CS does. The GS may raise an iteration endless loop with the increment of path searching number;
4) The solution process of VRPTW is improved with the four types of metaheuristic optimization methods, and the optimization performances of different problems in different ways are different. A decrease of efficiency and speed also exits for a part of the way of optimization. Different VRPTW problems should choose different heuristic optimization methods and parameters.

Although CGAMO for VRPTW obtains satisfactory achievements, there are still some room for improvement:

1) How to keep convergence speed steady of CGAMO for the distribution of path is different and the number of iteration is different;
2) How to choose the different metaheuristic optimization functions and the parameters. These would be the focus of our future work.

Acknowledgements

This work is partially supported by National Natural Science Foundation, China (No. 70901060); Hubei Province Natural Science Foundation (No. 2011CDB461); State Key Lab of Software Engineering Open Foundation (No. SKLSE2010-08-15); Youth Plan Found of Wuhan City (No. 201150431101) and the Fundamental Research Funds for the Central Universities. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

References

Achuthan, N. R.; Caccetta, L.; Hill, S. P. 2003. An improved branch-and-cut algorithm for the capacitated vehicle routing problem, *Transportation Science* 37(2): 153–169. http://dx.doi.org/10.1287/trsc.37.2.153.15243

Ai, T. J.; Kachitvichyanukul, V. 2009. Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem, *Computers & Industrial Engineering* 56(1): 380–387. http://dx.doi.org/10.1016/j.cie.2008.06.012

Balseiro, S. R.; Loiseau, I.; Ramonet, J. 2011. An ant colony algorithm hybridized with insertion heuristics for the time dependent vehicle routing problem with time windows, *Computers & Operations Research* 38(6): 954–966. http://dx.doi.org/10.1016/j.cor.2010.10.011

Chakroborty, P.; Mandal, A. 2005. An asexual genetic algorithm for the general single vehicle routing problem, *Engineering Optimization* 37(1): 1–27. http://dx.doi.org/10.1080/03052150410001721468

Chen, A.-L.; Yang, G.-K.; Wu, Z.-M. 2006. Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem, *Journal of Zhejiang University SCIENCE A* 7(4): 607–614. http://dx.doi.org/10.1631/jzus.2006.A0607

Cheng, C.-B.; Wang, K.-P. 2009. Solving a vehicle routing problem with time windows by a decomposition technique and a genetic algorithm, *Expert System with Applications* 36(4): 7758–7763. http://dx.doi.org/10.1016/j.eswa.2008.09.001

Cheung, R. K.; Hang, D. D. 2003. Multi-attribute label matching algorithms for vehicle routing problems with time windows and backhauls, *IIE Transacitons* 35(3): 191–205. http://dx.doi.org/10.1080/07408170304371

Christiansen, C. H.; Lysgaard, J. 2007. A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands, *Operations Research Letters* 35(6): 773–781. http://dx.doi.org/10.1016/j.orl.2006.12.009

Christofides, N.; Mingozzi, A.; Toth, P. 1981. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations, *Mathematical Programming* 20(1): 255–282. http://dx.doi.org/10.1007/BF01589353

Dantzig, G. B.; Wolfe, P. 1960. Decomposition principle for linear programs, *Operations Research* 8(1): 101–111. http://dx.doi.org/10.1287/opre.8.1.101

Desrochers, M.; Desrosiers, J.; Solomon, M. 1992. A new optimization algorithm for the vehicle routing problem with time windows, *Operations Research* 40(2): 342–354. http://dx.doi.org/10.1287/opre.40.2.342

Desrochers, M.; Soumis, F. 1988. A generalized permanent labelling algorithm for the shortest path problem with time windows, *INFOR: Information Systems and Operational Research* 26(3): 191–212.

Dumitrescu, I.; Boland, N. 2003. Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem, *Networks* 42(3): 135–153. http://dx.doi.org/10.1002/net.10090

Feillet, D.; Dejax, P.; Gendreau, M.; Gueguen, C. 2004. An exact algorithm for the elementary shortest path problem with resource constraints: application to some vehicle routing problems, *Networks* 44(3): 216–229. http://dx.doi.org/10.1002/net.20033

Foster, B. A.; Ryan, D. M. 1976. An integer programming approach to the vehicle scheduling problem, *Journal of the Operational Research Society* 27(2): 367–384. http://dx.doi.org/10.1057/jors.1976.63

Gajpal, Y.; Abad, P. L. 2009. Multi-ant colony system (MACS) for a vehicle routing problem with backhauls, *European Journal of Operational Research* 196(1): 102–117. http://dx.doi.org/10.1016/j.ejor.2008.02.025

Garcia-Najera, A.; Bullinaria, J. A. 2011. An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows, *Compters & Operstions Research* 38(1): 287–300. http://dx.doi.org/10.1016/j.cor.2010.05.004

Ghoseiri, K.; Ghannadpour, S. F. 2010. Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm, *Applied Soft Computing* 10(4): 1096–1107. http://dx.doi.org/10.1016/j.asoc.2010.04.001

Gomory, R. E. 1958. Outline of an algorithm for integer solutions to linear programs, *Bulletin of the American Mathematical Society* 64(5): 275–278. http://dx.doi.org/10.1090/S0002-9904-1958-10224-4

Gutiérrez-Jarpa, G., Marianov, V.; Obreque, C. 2009. A single vehicle routing problem with fixed delivery and optional collections, *IIE Transacitons* 41(12): 1067–1079. http://dx.doi.org/10.1080/07408170903113771

Hashimoto, H.; Yagiura, M.; Ibaraki, T. 2008. An iterated local search algorithm for the time-dependent vehicle routing problem with time windows, *Discrete Optimization* 5(2): 434–456. http://dx.doi.org/10.1016/j.disopt.2007.05.004

Hiquebran, D. T.; Alfa, A. S.; Shapiro, J. A.; Gittoes, D. H. 1993. A revised simulated annealing and cluster-first route-second algorithm applied to the vehicle routing problem, *Engineering Optimization* 22(2): 77–107. http://dx.doi.org/10.1080/03052159308941327

Hong, L. 2012. An improved LNS algorithm for real-time vehicle routing problem with time windows, *Computers & Operations Research* 39(2): 151–163. http://dx.doi.org/10.1016/j.cor.2011.03.006

Irnich, S.; Villeneuve, D. 2006. The shortest-path problem with resource constraints and k-cycle elimination for k≥3, *INFORMS Journal on Computing* 18(3): 391–406. http://dx.doi.org/10.1287/ijoc.1040.0117

Jepsen, M.; Petersen, B.; Spoorendonk, S.; Pisinger, D. 2008. Subset-row inequalities applied to the vehicle-routing problem with time windows, *Operations Research* 56(2): 497–511. http://dx.doi.org/10.1287/opre.1070.0449

Karmarkar, N. 1984. A new polynomial-time algorithm for linear programming, *Combinatorica* 4(4): 373–395. http://dx.doi.org/10.1007/BF02579150

Kohl, N. 1995. *Exact Methods for Time Constrained Routing and Related Scheduling Problems*: PhD Thesis. Technical University of Demark. 234 p.

Kolen, A. W. J.; Kan, A. H. G. R.; Trienekens, H. W. J. M. 1987. Vehicle routing with time windows, *Operations Research* 35(2): 266–273. http://dx.doi.org/10.1287/opre.35.2.266

Laporte, G.; Mercure, H.; Nobert, Y. 1992. A branch and bound algorithm for a class of asymmetrical vehicle routeing problems, *Journal of the Operational Research Society* 43(5): 469–481. http://dx.doi.org/10.2307/2583566

Laporte, G.; Mercure, H.; Nobert, Y. 1986. An exact algorithm for the asymmetrical capacitated vehicle routing problem, *Networks* 16(1): 33–46. http://dx.doi.org/10.1002/net.3230160104

Lau, H. C.; Sim, M.; Teo, K. M. 2003. Vehicle routing problem with time windows and a limited number of vehicles, *European Journal of Operational Research* 148(3): 559–569. http://dx.doi.org/10.1016/S0377-2217(02)00363-6

Li, X. Y.; Tian, P.; Leung, S. C. H. 2010. Vehicle routing problems with time windows and stochastic travel and service times: models and algorithm, *International Journal of Production Economics* 125(1): 137–145. http://dx.doi.org/10.1016/j.ijpe.2010.01.013

Lorenz, D. H.; Raz, D. 2001. A simple efficient approximation scheme for the restricted shortest path problem, *Operations Research Letters* 28(5): 213–219. http://dx.doi.org/10.1016/S0167-6377(01)00069-4

Lübbecke, M. E.; Desrosiers, J. 2005. Selected topics in column generation, *Operations Research* 53(6): 1007–1023. http://dx.doi.org/10.1287/opre.1050.0234

Mautor, T.; Naudin, E. 2007. Arcs-states models for the vehicle routing problem with time windows and related problems, *Computers & Operations Research* 34(4): 1061–1084. http://dx.doi.org/10.1016/j.cor.2005.05.024

Nazareth, J. L. 1988. *Computer Solution of Linear Programs*. Oxford University Press. 254 p.

Nelder, J. A.; Mead, R. 1965. A simplex method for function minimization, *The Computer Journal* 7(4): 308–313. http://dx.doi.org/10.1093/comjnl/7.4.308

Padberg, M.; Rinaldi, G. 1987. Optimization of a 532-city symmetric traveling salesman problem by branch and cut, *Operations Research Letters* 6(1): 1–7. http://dx.doi.org/10.1016/0167-6377(87)90002-2

Qureshi, A. G.; Taniguchi, E.; Yamada, T. 2009. An exact solution approach for vehicle routing and scheduling problems with soft time windows, *Transportation Research Part E: Logistics and Transportation Review* 45(6): 960–977. http://dx.doi.org/10.1016/j.tre.2009.04.007

Reimann, M.; Doerner, K.; Hartl, R. F. 2004. D-ants: savings based ants divide and conquer the vehicle routing problem, *Computers & Operations Research* 31(4): 563–591. http://dx.doi.org/10.1016/S0305-0548(03)00014-5

Rousseau, L.-M.; Gendreau, M.; Feillet, D. 2007. Interior point stabilization for column generation, *Operations Research Letters* 35(5): 660–668. http://dx.doi.org/10.1016/j.orl.2006.11.004

Savelsbergh, M. W. P. 1985. Local search in routing problems with time windows, *Annals of Operations Research* 4(1): 285–305. http://dx.doi.org/10.1007/BF02022044

Sellmann, M.; Gellermann, T.; Wright, R. 2007. Cost-based filtering for shorter path constraints, *Constraints* 12(2): 207–238. http://dx.doi.org/10.1007/s10601-006-9006-4

Sungur, I.; Ordóñez, F.; Dessouky, M. 2008. A robust optimization approach for the capacitated vehicle routing problem with demand uncertainty, *IIE Transactions* 40(5): 509–523. http://dx.doi.org/10.1080/07408170701745378

Tavakkoli-Moghaddam, R.; Gazanfari, M.; Alinaghian, M.; Salamatbakhsh, A.; Norouzi, N. 2011. A new mathematical model for a competitive vehicle routing problem with time windows solved by simulated annealing, *Journal of*

Manufacturing Systems 30(2): 83–92. http://dx.doi.org/10.1016/j.jmsy.2011.04.005

Teodorovic, D.; Krcmar-Nozic, E.; Pavkovic, G. 1995. The mixed fleet stochastic vehicle routing problem, *Transportation Planning and Technology* 19(1): 31–43. http://dx.doi.org/10.1080/03081069508717556

Vaidyanathan, B. S.; Matson, J. O.; Miller, D. M.; Matson, J. E. 2007. A capacitated vehicle routing problem for just-in-time delivery, *IIE Transctions* 31(11): 1083–1092. http://dx.doi.org/10.1023/A:1007631726356

Vehicle Routing Problems with Time Windows (VRPTW): Solomon Benchmark Problems. 2012. Available from Internet: http://www.idsia.ch/~luca/macs-vrptw/problems/welcome.htm

Zachariadis, E. E.; Kiranoudis, C. T. 2010. A strategy for reducing the computational complexity of local search-based methods for the vehicle routing problem, *Compters & Operstions Research* 37(12): 2089–2105. http://dx.doi.org/10.1016/j.cor.2010.02.009

Zhu, X.; Wilhelm, W. E. 2012. A three-stage approach for the resource-constrained shortest path as a sub-problem in column generation, *Computers & Operations Research* 39(2): 164–178. http://dx.doi.org/10.1016/j.cor.2011.03.008