Available online at www.sciencedirect.com
ScienceDirect
Computers & Operations Research 34 (2007) 817–834
computers & operations research
www.elsevier.com/locate/cor

Stabilized column generation for highly degenerate multiple-depot vehicle scheduling problems

Amar Oukil[a], Hatem Ben Amor[b], Jacques Desrosiers[a,*], Hicham El Gueddari[a]

[a]*GERAD and HEC Montréal, 3000, chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7*
[b]*GERAD and École Polytechnique de Montréal, C.P. 6079, Succ. Centre-ville, Montréal, Canada H3C 3A7*

Available online 20 March 2006

Abstract

Column generation has proven to be efficient in solving the linear programming relaxation of large scale instances of the multiple-depot vehicle scheduling problem (MDVSP). However difficulties arise when the instances are highly degenerate. Recent research has been devoted to accelerate column generation while remaining within the linear programming framework. This paper presents an efficient approach to solve the linear relaxation of the MDVSP. It combines column generation, preprocessing variable fixing, and stabilization. The outcome shows the great potential of such an approach for degenerate instances.

Keywords: Column generation; Degeneracy; Stabilization; Multiple-Depot Vehicle Scheduling Problem

1. Introduction

The *multiple-depot vehicle scheduling problem* (MDVSP) consists in assigning a set of time-tabled trips to a set of vehicles that are provided by several depots, such as to minimize a given objective function. Column generation has proven to be efficient in solving the linear programming (LP) relaxation of large scale instances of the MDVSP but difficulties arise with this approach when the instances are highly degenerate. Recent research has been devoted to improve the behavior of the solution process

* Corresponding author.
E-mail addresses: amar.oukil@gerad.ca (A. Oukil), Hatem.Ben.Amor@gerad.ca (H.B Amor), Jacques.Desrosiers@gerad.ca (J. Desrosiers), Hicham.El.Gueddari@gerad.ca (H. El Gueddari).

0305-0548/$ - see front matter © 2005 Published by Elsevier Ltd.
doi:10.1016/j.cor.2005.05.011

through algorithms that stabilize and accelerate column generation while remaining within the LP framework.

The present paper deals with an approach to solve the LP relaxation of highly degenerate MDVSP based on an algorithm that combines column generation, preprocessing variable fixing, and stabilization. The proximal algorithm described in [1] is used to solve the LP relaxation of instances of *long horizon* MDVSPs where the number of trips and depots are fixed while the time length of a route, identified as the horizon, is extended to multiples of the basic period. These are problems with massive degeneracy that can barely be solved by the standard column generation approach. Hence, we apply stabilized column generation and show how large is the potential of such an approach for degenerate instances.

After a literature review on the general MDVSP, we stress the particular aspects of the long horizon MDVSP through an analysis of results provided by solving the LP relaxation of a set of test problems by the standard column generation method. Next, we propose a network reduction approach that provides, in addition to a smaller set of arcs, a lower bound on the (integer) solution of the MDVSP, a primal integer solution which allows to compute an upper bound, and a dual feasible solution. The dual solution is used as the initial stability center for the stabilized column generation. Finally, we present the main ideas of the stabilized column generation with some experimental results that show the extent of the stabilizing and accelerating effects of the proposed approach.

2. The long horizon MDVSP

2.1. Definition

The MDVSP is an important combinatorial optimization problem that arises in the area of time constrained routing and scheduling. In this problem, we are given a set of n trips, $T_1, T_2, \ldots, T_n$, each trip T_j $(j=1,\ldots,n)$ starting at time s_j and ending at time e_j, along with a set of m depots, $D_1, D_2, \ldots, D_m$, in the kth of which v_k vehicles are stationed $(k=1,\ldots,m)$. Let t_{ij} denote the travel time between the ending point of trip T_i and the starting point of trip T_j. An ordered pair of trips (T_i, T_j) is said to be *compatible* if it satisfies the relation $e_i + t_{ij} \leqslant s_j$. The MDVSP can then be described by the multigraph $G=(V, A)$ where the vertex set V is the union of the set of trips and the set of depots, while the arc set A contains m copies of each compatible arc and all the arcs of the form (D_k, T_i) and (T_i, D_k) for $k=1,\ldots,m$ and $i=1,\ldots,n$. Let c_{ij} be the cost incurred if a vehicle performs the trip T_i immediately before the trip T_j. Similarly, let $c_{n+k,j}$ (resp. $c_{j,n+k}$) denote the cost incurred if T_j is the first (resp. the last) trip undertaken by a vehicle housed at depot D_k; these costs include a fixed cost large enough to optimize the number of required vehicles. If R is the arc set denoting a route originating at the depot D_k, the cost of R is $c_{n+k,f} + \sum_{(T_i,T_j)\in R} c_{i,j} + c_{\ell,n+k}$, where T_f and T_ℓ are, respectively, the first and the last trips in R. Therefore, the problem consists of finding a feasible assignment of minimum cost, where the cost of an assignment is the sum of the costs of the routes in the assignment, the number of vehicles leaving from the depot D_k is at most v_k (for all k), each trip is assigned to a single vehicle, and each vehicle used in the solution starts and ends its route at the same depot.

The horizon of a MDVSP is the maximal period of time allowed to accomplish a route. Let us assume that the basic horizon is one day. The *long horizon* MDVSP is addressed when the horizon is extended to multiple days, while keeping the number of trips and depots of the one-day MDVSP unchanged.

2.2. Formulation

The MDVSP has been investigated for more than two decades (see [2]) and we refer the reader to Desrosiers et al. [3] for a survey on the MDVSP and related scheduling problems. Bertossi et al. [4] have proved that the MDVSP is NP-hard for $m \geqslant 2$. Exact algorithms for solving the MDVSP have been proposed by Carpaneto et al. [5], Ribeiro and Soumis [6], Bianco et al. [7], Forbes et al. [8], Löbel [9], and more recently Hadjar et al. [10]. Algorithms that use a column generation scheme to solve the linear relaxation of the MDVSP are based on set partitioning formulations (see [6,10]).

Let Ω be the set of feasible routes. With c_p denoting the cost of a route p ($p \in \Omega$) define the binary constants a_{ip} and b_{kp} such as: $a_{ip} = 1$ if and only if route p covers trip T_i and $b_{kp} = 1$ if and only if route p starts and ends at depot D_k. Using binary variables θ_p to indicate whether a route p ($p \in \Omega$) is to be used to cover a set of trips, the MDVSP is an integer linear program that can be formulated as follows:

$$\text{(MDVSP)}\ Z_M = \min \sum_{p \in \Omega} c_p \theta_p, \tag{1}$$

$$\sum_{p \in \Omega} a_{ip} \theta_p = 1, \quad i = 1, 2, \ldots, n \tag{2}$$

$$\sum_{p \in \Omega} b_{kp} \theta_p \leqslant v_k, \quad k = 1, 2, \ldots, m \tag{3}$$

$$\theta_p \in \{0, 1\}, \quad \forall p \in \Omega \tag{4}$$

where (1) is the objective, that is minimizing the total cost of the selected routes, (2) is the set of constraints that require each trip is covered exactly once, (3) denotes the availability of vehicles in every depot k, and (4) are the binary conditions. The LP relaxation of the above formulation is called the master problem (*MP*). The corresponding dual problem, denoted *MD*, is formulated as follows:

$$\text{(MD)}\ \max \sum_{i=1}^{n} \pi_i - \sum_{k=1}^{m} v_k \lambda_k, \tag{5}$$

$$\sum_{i=1}^{n} a_{ip} \pi_i - \sum_{k=1}^{m} b_{kp} \lambda_k \leqslant c_p \quad \forall p \in \Omega, \tag{6}$$

$$\lambda_k \geqslant 0 \quad k = 1, 2, \ldots, m. \tag{7}$$

As the number of routes in Ω is huge, the corresponding variables or columns in *MP* are generated dynamically in practice. The columns are generated by solving a classical shortest path problem on the acyclic compatibility network $G = (V, A)$ to compute negative reduced cost variables, if any. The dual multipliers are computed by solving a restricted *MP* that considers only a small subset Ω' of columns at once ($\Omega' \subset \Omega$).

2.3. Results of the standard column generation approach

The standard column generation approach is implemented in the software system GENCOL 4.3 (see [11]) to solve the LP relaxation of the long horizon MDVSP. The maximum size of the restricted *MP* is

Table 1
Average results of the standard column generation

Horizon	Instances	*Arcs*	*CPU* (s)	v^*	n/v^*
1 day	10	277602.7	1770	90.6	5.52
2 days	10	323463.0	9730	49.3	10.14
3 days	5	340108.8	23430	35.8	13.97
4 days	5	348769.2	36653	29.2	17.12
5 days	5	353535.6	51583	26.6	18.80
6 days	2	359674.5	62359	23.5	21.28
7 days	2	362566.5	121346	20.0	25.00

set to 5000 columns. At every column generation iteration, 50 to 100 columns are selected and added to the restricted *MP*. Whenever there are more than 5000 columns, the worst ones are removed on the basis of their reduced costs. The minimum number of columns kept at every iteration is 2500. The restricted *MP* is solved with the primal simplex method.

Test problems are randomly generated to simulate real-life public transport system. The basic generator was designed by Carpaneto et al. [5] and used by other researchers (see [6–8]). The generator is modified in such a way that the horizon could be extended to several days. Since the study focuses only on the degeneracy aspect of the MDVSP, we set the number of trips to $n = 500$ and the number of depots to $m = 3$ whatever is the horizon. With horizons of 1 to 7 days, we generated ten MDVSP instances of type A (see [5]) for each horizon.

Tests with the standard column generation approach were carried out on an Enterprise 1000 Ultra workstation (400 MHz). The average results are summarized in Table 1, where *Arcs* denotes the number of arcs in the network, *CPU* the computing time (in seconds) and v^* the optimal number of vehicles required. The last column contains the column density (n/v^*), that is, the average number of trips covered per route.

Given the computing time trend, the number of instances was limited to five for problems of 3–7-day horizons. The instances of 1–5-day horizon were entirely solved to LP-optimality and only two instances of the 6 and 7-day horizons could be solved within the allocated cpu time (36 h).

Table 1 shows that the average number of arcs increases as the horizon extends. Such a trend is expected since the trips are more spread and, as a result, more pairs of trips are compatible. However, the slope seems flattening over a 3-day horizon with a number of arcs neighbouring $m \times n^2/2$. Meanwhile, the average cpu time per horizon is growing exponentially from horizon 1–7 days. As revealed by the density of service (see Fig. 1), such a behavior was expected because of the enlargement that affects the routes. The longer is the horizon, the larger is the number of trips per route, leading to less required vehicles v^*, as given in Table 1. Hence, the problem is more degenerate. Such a situation is easily understood through the particular case of a single depot ($m = 1$) which is a pure network flow problem. The *MP* is basically composed of $n + 1$ rows, meaning that the basis of the simplex tableau is necessarily a $(n + 1) \times (n + 1)$ matrix. Since only v^* columns are needed to cover all trips, there are $n + 1 - v^*$ basic variables θ_p ($p \in \Omega$) with zero values, i.e., the problem is degenerate. Reducing v^* increases $n + 1 - v^*$ and, consequently, causes more degeneracy.

Some problems in the literature show the same behavior, such as the urban bus driver scheduling problem where the number of trip segments per duty (working day) exceeds more often 30. Other examples

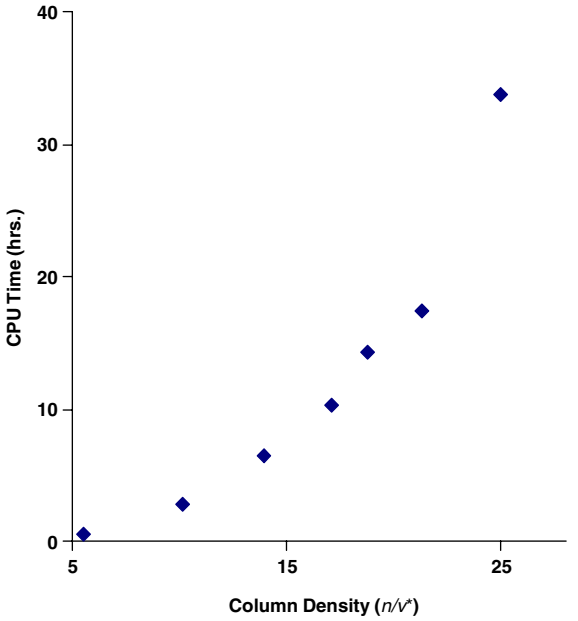

Fig. 1. Impact of column density on standard column generation CPU.

occur in vehicle routing problems for which the scheduling horizon is a week or more, e.g., the aircraft routing problem.

3. Preprocessing

Although the number of arcs in G is $O(mn^2)$, any optimal integer solution of the MDVSP contains only $n + v^*$ arcs. Therefore, our preprocessing aims to achieve as much reduction as possible of G. We first solve a relaxation of the MDVSP, that is a *single depot vehicle scheduling problem* (SDVSP), to compute a lower bound and the optimal number of vehicles v^*. The primal solution of the SDVSP is also used to derive an integer solution to the MDVSP, hence an upper bound on its integer optimal value. The dual solution of the SDVSP is used later to compute an initial stability center for a stabilized column generation algorithm.

3.1. Network reduction

Consider the following relaxation of the MDVSP. We aggregate all depots $D_k (k = 1, \ldots, m)$ within a single virtual depot D. With 0 denoting the index of depot D, we compute the costs of the outer arcs $(0, T_j)$ and the inner arcs $(T_j, 0)$ as $c_{0,j} = \min_{k=1,\ldots,m} c_{n+k,j}$ and $c_{j,0} = \min_{k=1,\ldots,m} c_{j,n+k}$ for $j = 1, \ldots, n$. With $\delta^+(i)$ (resp. $\delta^-(i)$) defining the set of successors (resp. predecessors) of i $(i = 1, \ldots, n)$, the resulting SDVSP, a pure network problem, is formulated as

$$\text{(SDVSP)} \quad Z_S = \min \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{8}$$

$$\sum_{j\in\delta^+(i)} x_{ij} = 1, \quad i = 1, 2, \ldots, n \tag{9}$$

$$\sum_{j\in\delta^-(i)} x_{ji} - \sum_{j\in\delta^+(i)} x_{ij} = 0, \quad i = 0, 1, \ldots, n \tag{10}$$

$$\sum_{j=1}^{n} x_{0j} \leqslant \sum_{k=1}^{m} v_k \tag{11}$$

$$x_{ij} \geqslant 0, \quad \forall (i, j) \in A. \tag{12}$$

In the above formulation, (8) represents the objective function, (9) requires each trip to be covered exactly once, (10) is the set of flow conservation constraints, (11) denotes the availability of the total number of vehicles in depot D, and (12) are the non-negativity conditions.

The objective value Z_S of the SDVSP is a lower bound on the optimal value Z_M of the MDVSP. We have set the fixed costs of the depots inner and outer arcs high enough (a total of 10 000 in our experiments) to allow the minimum cost flow problem determine the optimal number of vehicles v^* required for the whole journey (see [5]).

Given that the primal optimal solution of the SDVSP provides v^* paths, we solve a transportation problem of size $v^* \times m$ to assign each path once to the appropriate depot. The assignment cost of a path R to a depot D_k is given by $c_{n+k,f} + \sum_{(T_i,T_j)\in R} c_{i,j} + c_{\ell,n+k}$ $(k = 1, 2, \ldots, m)$ where T_f and T_ℓ are, respectively, the first and the last trips in the path. Since the transportation problem optimal objective Z_T provides an upper bound on the integer optimal value of the MDVSP, it can be stated that $Z_S \leqslant Z_M \leqslant Z_T$. Let $\bar{c}_{ij}$ denote the reduced cost of the arc (i, j) with respect to the SDVSP solution. As in Hadjar et al. [10], any binary variable x_{ij}, $(i, j) \in A$, which reduced cost is larger than the optimality gap can be removed from the formulation, i.e., if $\bar{c}_{ij} > Z_T - Z_S$. Let A_1 be the reduced set of arcs.

The network reduction algorithm is applied on 7 pairs of test problems, two problems from each horizon, identified respectively as *d.1* and *d.2*, where *d* is the number of days in the horizon. As shown in Table 2, the network reduction $|A_1|/|A|$ is more relevant for problems of the 3-day horizon and over, with a ratio below 25% for one instance.

To assess the impact of such an arc reduction level on the standard column generation procedure, we solved the LP relaxation of the 14 reduced problems. In Table 2, *CPU ini* and *CPU red* refer to the cpu time before and after arc reduction, respectively, while *CPU ratio* is the ratio given in percent. *Avg* denotes the average value on each column of the table. Even if there is a slight improvement of some cpu times, most problems remain very difficult. If the upper bound's value is decreased, more network reduction could be achieved. The optimality gap $Z_T - Z_S$ looks not so large but there is still a possibility to improve the upper bound Z_T.

3.2. Extension of the network reduction approach

A set partitioning problem, formulated as in (1)–(4), can be solved to provide an upper bound Z_P such as $Z_P \leqslant Z_T$. With the v^* routes, solution of the SDVSP, the artificial depot is replaced by each of the m real depots. Hence the initial set Ω' contains $m \times v^*$ columns. To add new feasible columns into Ω',

Table 2
Results of the network reduction

Pb.	Z_S	$Z_T - Z_S$	$\|A\|$	$\|A_1\|/\|A\|$ (%)	*CPU ini* (s)	*CPU red* (s)	*CPU ratio* (%)
1.1	961026	6504	281454	65.5	2498	2078	83.2
1.2	1081809	12278	279606	99.5	877	925	105.5
2.1	619126	5950	324042	84.1	11172	10736	96.1
2.2	615529	5298	326934	87.3	7575	7340	96.9
3.1	474598	1350	344187	39.5	22772	19083	83.8
3.2	580704	4820	343107	58.1	26402	22706	86.0
4.1	436274	2827	351708	39.1	48144	46315	96.2
4.2	464337	1417	352173	40.4	27829	24629	88.5
5.1	458439	1555	356997	30.4	55931	54980	98.3
5.2	472332	2450	355908	28.0	36782	33950	92.3
6.1	401864	1272	360069	23.3	77855	65787	84.5
6.2	454116	3133	359280	29.6	46863	39927	85.2
7.1	440135	1381	362481	33.4	117030	97720	83.5
7.2	441142	1116	362652	35.4	125661	110833	88.2
Avg	564387.9	3667.9	340042.7	49.5	43385.1	38357.8	90.6

the SDVSP (8)–(12) is solved on the reduced network $G_1 = (V, A_1)$. The v_1^* columns that are not identical to the v^* previous ones ($v_1^* \leqslant v^*$), are added to solve a small MDVSP with $(v^* + v_1^*) \times m$ columns. Thus an upper bound Z_P is obtained, and more network reduction is possible. Such a process can be run as long as new columns can be found to enlarge Ω'. Alternatively the stopping criterion could be based on the number of reduction iterations (*phases*), on a running time limit for the reduction process, or on a level of acceptable improvement of the upper bound Z_P. We opted for the first criterion in the present application.

The extended network reduction approach is tested with the above set of 14 instances and we obtained the results of Table 3, where A_r is the resulting reduced set of arcs after r reduction *Phases*. The network reduction $|A_r|/|A|$ is significant, mainly for 3-day horizon problems and over, with figures of less than 4% of the network initial arc number, as the case for the 7-day horizon instances.

The solving process with the standard column generation approach is run on the LP relaxation of the MDVSP using the deeper reduced network. In Table 3, *CPU prep* denotes the preprocessing time (in seconds) while *CPU ratio* is computed as (*CPU red* + *CPU prep*)/*CPU ini*. Once more, no significant improvement of the cpu time is achieved even if the average cpu time required by the standard column generation approach is about 8 h (27 470 s) rather than 12 h (43 385 s). Hence, a deeper network reduction alone remains without any substantial influence on the cpu time. Primal degeneracy, among other factors, is surely responsible for such inconvenience beside instability in the behavior of the values of the dual variables. The next section discusses a methodology to deal with such an issue through a stabilized column generation approach.

4. A stabilized column generation approach

There are two key concepts to define a stabilization approach for the solution of the primal-dual pair of linear problems *MP* and *MD*: the *stability center* and the *penalty function*. The *stability center* is a dual

Table 3
Results of the multiple phase reduction process

Pb.	$Z_P - Z_S$	$\lvert A_r\rvert/\lvert A\rvert$ (%)	*Phases* r	*CPU ini* (s)	*CPU prep* (s)	*CPU red* (%)	*CPU ratio*
1.1	5716	65.2	2	2498	5	2005	80.8
1.2	9538	73.7	11	877	124	856	113.1
2.1	2782	59.7	39	11172	1108	11314	111.4
2.2	2026	53.7	10	7575	149	7517	101.3
3.1	272	9.8	91	22772	4423	12853	75.9
3.2	2828	57.8	7	26402	52	28320	107.5
4.1	684	14.6	41	48144	582	40616	85.6
4.2	493	17.0	61	27829	2001	19270	76.5
5.1	1240	27.5	5	55931	15	50270	90.0
5.2	1327	24.8	11	36782	162	29949	81.9
6.1	290	6.8	60	77855	1937	54069	72.1
6.2	907	16.8	25	46863	300	41982	90.2
7.1	163	3.6	60	117030	1835	43638	38.9
7.2	99	2.7	60	125661	6651	41926	38.7
Avg	2026.1	31.0	34.5	43385.1	1381.7	27470.4	83.1

vector that is adequately chosen while the *penalty function* is the term that is added to the dual objective to penalize any move far from this center. Such penalization aims to control and eventually guide the progress of the dual variables during column generation solution process in a way that better quality columns are generated. The stability center is updated with the current dual point if the dual Lagrangian lower bound is sufficiently better. An extensive theoretical work is carried out on the ground of these ideas since the seventies and several stabilized algorithms are developed (see [12]).

The proposed approach is a proximal type algorithm (see [13]) that combines penalty and trust region concepts. A linear penalty function is used to constrain the dual variables to a box that contains the current dual stability center before the stabilized problems are solved by column generation. The trust region concept is introduced in literature through the Boxstep method (see [14,15]). Schramm and Zowe [16] show that, by adding some features of the trust region philosophy to the bundle concept, a more stable behavior of the cutting plane algorithm is achieved for convex and nonconvex functions. Neame [17] proposes a unified framework for stabilization approaches for unconstrained maximization of piecewise linear concave functions. A stabilization scheme which remains within the linear programming framework is defined by du Merle et al. [18]. The authors use a 3-piecewise linear penalty function to stabilize the column generation procedure. In the following paragraphs, we describe an approach presented by Ben Amor and Desrosiers [1] that is based on a 5-piecewise linear penalty function.

4.1. Formulation of the stabilized primal and dual problems

Given a linear (primal) program P and its dual D, the approach consists in solving P by using a series of linear programming approximations denoted by SP_ℓ ($\ell = 1, 2, ...$) also called stabilized primal problems. The index ℓ refers to the current major iteration which corresponds to the application of the column generation approach to solving the current stabilized problem. Let SD_ℓ be the dual version of SP_ℓ. Every major iteration ℓ is followed with an update of the penalty function coefficients which

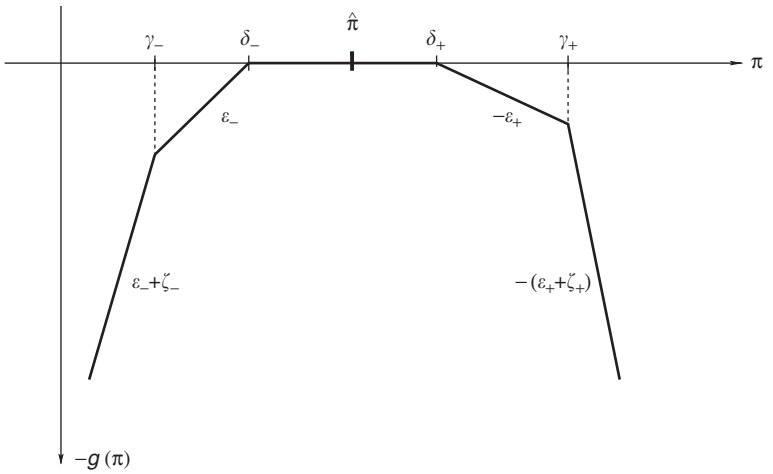

Fig. 2. 5-piecewise linear penalty function.

are identified as stabilization parameters. Convenient updating strategies are used to guarantee the convergence of the series of stabilized pairs of problems (SP_ℓ, SD_ℓ) towards the optimum of the primal and dual problems P and D. Further details on the theoretical aspects of the approach can be found in Ben Amor et al. [19].

At any major iteration ℓ, the proposed proximal point algorithm uses a non penalized trust region that contains the current stability center $\hat{\pi}^\ell$ and a concave penalty function g^ℓ. In order to keep solving linear programs, we use piecewise linear penalty functions. As in Ben Amor and Desrosiers [1], Fig. 2 illustrates an approximation of a quadratic penalty function where no penalty is applied in the trust region defined by $[\delta_-, \delta_+]$. Function $g_i^\ell(\pi)$ consists of the component $\hat{\pi}_i^\ell$ of the stability center $\hat{\pi}^\ell$ with the stabilization parameters $\gamma_-^\ell \leqslant \delta_-^\ell < \delta_+^\ell \leqslant \gamma_+^\ell$ and ζ_-^ℓ, ε_-^ℓ, ε_+^ℓ, $\zeta_+^\ell > 0$.

In our application, we consider the pair of primal and dual problems *MP* and *MD*. We stabilize only the trip partitioning constraints, that is set (2) in the formulation (1)–(4), since the constraints on the vehicle availability are not binding in our test problems and are in a small number ($m = 3$). According to the penalty function illustrated in Fig. 2, at any major iteration ℓ, the stabilized dual problem, identified by SMD_ℓ, can be formulated as follows:

$$(SMD_\ell) \quad \max \sum_{i=1}^{n} \left(\pi_i^\ell - \zeta_{-,i}^\ell v_i^- - \varepsilon_{-,i}^\ell u_i^- - \varepsilon_{+,i}^\ell u_i^+ - \zeta_{+,i}^\ell v_i^+ \right) - \sum_{k=1}^{m} v_k \lambda_k$$

$$\sum_{i=1}^{n} a_{ip} \pi_i^\ell - \sum_{k=1}^{m} b_{kp} \lambda_k \leqslant c_p, \quad \forall p \in \Omega$$

$$\delta_{-,i}^\ell - u_i^- \leqslant \pi_i^\ell \leqslant \delta_{+,i}^\ell + u_i^+, \quad i = 1, \ldots, n$$

$$\gamma_{-,i}^\ell - v_i^- \leqslant \pi_i^\ell \leqslant \gamma_{+,i}^\ell + v_i^+, \quad i = 1, \ldots, n$$

$$u^-, v^-, \lambda, u^+, v^+ \geqslant 0.$$

Table 4
Quality of dual estimates

Pb.	Z_S	$(Z_{LP} - Z_S)/Z_{LP}$ (%)	v^*	$(Z_{LP} - Z_S)/(Z_{LP} - 10000v^*)$ (%)
1.1	961026	0.0625	87	0.6559
1.2	1081809	0.1792	99	2.0720
2.1	619126	0.037	50	0.1922
2.2	615529	0.0316	50	0.1679
3.1	474598	0.0448	34	0.1576
3.2	580704	0.0432	40	0.1386
4.1	436274	0.0454	28	0.1265
4.2	464337	0.0135	30	0.0381
5.1	458439	0.0322	25	0.0707
5.2	472332	0.0229	29	0.0592
6.1	401864	0.0443	22	0.0978
6.2	454116	0.0154	25	0.0343
7.1	440135	0.0034	20	0.0062
7.2	441142	0.0063	20	0.0116

The corresponding stabilized primal problem, identified by SMP_ℓ, can be formulated as follows:

$$(SMP_\ell) \quad \min \sum_{p\in\Omega} c_p\theta_p + \sum_{i=1}^{n} \left(-\gamma^\ell_{-,i} z_i^- - \delta^\ell_{-,i} y_i^- + \delta^\ell_{+,i} y_i^+ + \gamma^\ell_{+,i} z_i^+\right)$$

$$\sum_{p\in\Omega} a_{ip}\theta_p - z_i^- - y_i^- + y_i^+ + z_i^+ = 1, \quad i = 1,\ldots,n$$

$$\sum_{p\in\Omega} b_{kp}\theta_p \leqslant v_k, \quad k = 1,\ldots,m$$

$$\theta \geqslant 0,\ 0 \leqslant z^- \leqslant \zeta_-,\ 0 \leqslant y^- \leqslant \varepsilon_-, \quad 0 \leqslant y^+ \leqslant \varepsilon_+,\ 0 \leqslant z^+ \leqslant \zeta_+.$$

In the above primal problem, y^- and y^+ are vectors of surplus and slack variables, respectively, with upper bound vectors ε_- and ε_+. The cost coefficient of y^+ in the objective function is the upper bound vector δ_+ of the trust region while the coefficient of y^- is the negative of the lower bound vector, i.e., $-\delta_-$. Indeed, the bounded surplus and slack variables may be seen as a perturbation of the covering constraints of *MP*. This is a well known strategy to face degeneracy of linear programs. A further step in the proposed stabilization approach consists in plugging the trust region bounds as cost coefficients for these surplus and slack variables. In a similar manner, z^- and z^+ are bounded surplus and slack variables for which the cost coefficients are based on the outer interval $[\gamma_-, \gamma_+]$.

4.2. Initialization of the stability center

As displayed in Table 4, the gap between the solution of the LP relaxation Z_{LP} and the lower bound Z_S takes values between 0.0034% and 0.1792%. These figures suggest that Z_{LP} gets closer to Z_S as the MDVSPs horizon extends. The same trend is noticed with the last column's gaps, computed on the basis

of the adjusted values of Z_{LP} which exclude the large fixed cost required by using a vehicle. Therefore, the SDVSPs dual solution could be considered as a good estimate for *MD* and, subsequently, a good initialization of the stability center. Referring to the SDVSP formulation (8)–(12), let $\hat{\alpha}_i$ and $\hat{\beta}_i$ be the optimal dual variables associated with constraints (9) and (10), respectively, for $i = 1, \ldots, n$. Hence, the initial dual estimation of the *MD* multipliers is computed as $\hat{\pi}_{Si} = \hat{\alpha}_i + \hat{\beta}_i$, $i = 1, \ldots, n$.

Given that the SDVSPs dual solution is an auxilliary result of the network reduction algorithm, it is important to recall that there is a different dual solution to every reduction level. The stabilized approach is implemented with the first, the second and the last SDVSPs dual solutions identified as $\hat{\pi}_S$, $\hat{\pi}_{P2}$ and $\hat{\pi}_{Pr}$ respectively. Since any convex combination of these solutions remains feasible to *MD*, we also considered the average solutions $\hat{\pi}_A = (\hat{\pi}_S + \hat{\pi}_{P2})/2$, $\hat{\pi}_B = (\hat{\pi}_S + \hat{\pi}_{Pr})/2$ and $\hat{\pi}_C = (\hat{\pi}_S + \hat{\pi}_{P2} + \hat{\pi}_{Pr})/3$.

4.3. Initialization of the stabilization parameters

The choice of the initial values of the stabilization parameters is a relevant step towards more efficiency of the stabilized algorithm. As we assume that all components of the vectors of parameters take the same initial value, δ_+, δ_-, γ_+, γ_-, ζ_+, ζ_-, ε_+ and ε_- denote, in this section, the vectors' components rather than the vector itself, unless otherwise stated. The same assumption applies to $\hat{\pi}$.

Trust region: The trust region is a defined set around the stability center where no penalty is applied, based on the values of δ_+ and δ_-. As the interval is often centered on $\hat{\pi}$, δ_+ and δ_- could be chosen such that $\Delta = \delta_+ - \hat{\pi} = \hat{\pi} - \delta_-$. Practically, the only way to choose a good interval is through a series of trials on relatively easy problems, in addition to simple computing shrewdnesses involving the number of trips as well as the gaps between the available optima. Several values have been tested and, in spite of the problems disparity, all the stabilized processes have been efficient with small values of interval width. Since the gap between the SDVSPs optimum and the MDVSPs LP relaxation optimum is small for the most difficult problems, we set the half-interval width to $\Delta = 0.001$.

Outer region: The set $[\gamma_-, \delta_-[\cup]\delta_+, \gamma_+]$ defines the outer region. It refers to the left and right intervals outside the trust region, where the vectors of penalties ε_- and ε_+ are applied to prevent useless displacements far away from the current dual solution. These intervals are defined with γ_- and γ_+ such that $\Gamma = \gamma_+ - \delta_+ = \delta_- \gamma_- = 0.1$.

Penalties: High penalties restrict the displacements of dual variables outside the trust region both in frequency and in magnitude, while low penalties allow high magnitude displacements. Using very small ε_- and ε_+ barely penalizes values outside the trust region interval; hence, it may be seen as enlarging the trust interval width. Moreover, having $\varepsilon_+ + \zeta_+ \geqslant 1$ (the right-hand side of the covering constraints of the *MP*) provides an initial basic feasible solution and prevents the use of artificial variables in the beginning of the solution process. Given that the penalties can also be interpreted as perturbations of the covering constraints, we set $\varepsilon_- = -\varepsilon_+ = 0.1$ (a small value) and $\zeta_- = -\zeta_+ = 1.0$.

4.4. Updating strategies

To update the stabilization parameters, several strategies can be experimented. The next stability center $\hat{\pi}^{\ell+1}$ is always the optimal solution π^ℓ of SD_ℓ. Hence, the updating strategy focuses on the stabilization parameters. We experiment the *hybrid* strategy described in Ben Amor and Desrosiers [1] and a *dynamic* strategy. The hybrid strategy keeps the (inner) trust region interval width fixed to the initial value 2Δ and the slope parameters ζ_- and ζ_+ fixed to 1.0. At every major iteration, slope parameters ε_- and ε_+ are

Table 5
Average results of the stabilized column generation

Updating strategy	Simplex MP solution	Stabilized column generation cpu time (s)					
		$\hat{\pi}_S$	$\hat{\pi}_{P2}$	$\hat{\pi}_A$	$\hat{\pi}_{Pr}$	$\hat{\pi}_B$	$\hat{\pi}_C$
Hybrid	Primal	1635.9	1597.5	1333.4	556.9	**351.8**	618.1
	Dual	1393.9	1497.9	1277.6	808.6	**375.4**	442.4
Dynamic	Primal	627.8	1057.5	415.1	554.5	**348.0**	620.8
	Dual	428.4	611.9	387.0	808.5	**371.9**	441.5

reduced by a factor of 10. The outer region interval widths are updated whenever the next stability center falls on the boundary or outside $[\gamma^{\ell}_{-}, \gamma^{\ell}_{+}]$. If any component $\pi^{\ell}_i \geqslant \gamma^{\ell}_{+,i}$ (respectively $\pi^{\ell}_i \leqslant \gamma^{\ell}_{-,i}$), the right (respectively the left) interval width of the outer region is enlarged by a factor of 10 for all components.

The dynamic strategy modifies some parameters depending on the position of the current dual value with respect to the trust region and the outer intervals. The trust region and the penalties are either reduced or enlarged. Let $\Delta^{\ell}_{+,i} = \delta^{\ell}_{+,i} - \hat{\pi}^{\ell}_i$ and $\Delta^{\ell}_{-,i} = \hat{\pi}^{\ell}_i - \delta^{\ell}_{-,i}$ at major iteration ℓ. This strategy acts as follows:

- If the ith component π^{ℓ}_i of the current dual solution lies inside the trust region, the box $\left[\delta^{\ell}_{-,i}, \delta^{\ell}_{+,i}\right]$ width is reduced with a higher penality ε in the outer intervals:

$$\text{if } \pi^{\ell}_i \in]\delta^{\ell}_{-,i}, \delta^{\ell}_{+,i}[\text{then} \begin{cases} \Delta^{\ell+1}_{+,i} = \Delta^{\ell}_{+,i}/2 & \text{and} \quad \Delta^{\ell+1}_{-,i} = \Delta^{\ell}_{-,i}/2, \\ \varepsilon^{\ell+1}_{+,i} = \varepsilon^{\ell}_{+,i} \times 2 & \text{and} \quad \varepsilon^{\ell+1}_{-,i} = \varepsilon^{\ell}_{-,i} \times 2. \end{cases}$$

- If π^{ℓ}_i is at the border or outside the trust region, the latter is enlarged while the penalty is reduced:

$$\text{if } \pi^{\ell}_i \geqslant \delta^{\ell}_{+,i}, \text{ then } \Delta^{\ell+1}_{+,i} = \Delta^{\ell}_{+,i} \times 2 \text{ and } \varepsilon^{\ell+1}_{+,i} = \varepsilon^{\ell}_{+,i}/2,$$
$$\text{if } \pi^{\ell}_i \leqslant \delta^{\ell}_{-,i}, \text{ then } \Delta^{\ell+1}_{-,i} = \Delta^{\ell}_{-,i} \times 2 \text{ and } \varepsilon^{\ell+1}_{-,i} = \varepsilon^{\ell}_{-,i}/2.$$

With both strategies, the updated widths of $\Delta_{\pm,i}$ and $\Gamma_{\pm,i}$ are set to a maximum of 1.0 and 10.0 and a minimum of 0.1 and 1.0 respectively. Both values of $\varepsilon_{\pm,i}$ are set to a maximum of 1.0 and a minimum of 10^{-4}. $\zeta_{\pm,i}$ are kept fixed to 1.0.

5. Computational results

The stabilized algorithm is tested with the most reduced versions of the 14 MDVSPs' networks seen in the previous sections. Recall that we solve the LP relaxation of the MDVSP. Two techniques are tested to solve the restricted *MP*: the primal simplex and the dual simplex methods. As initial stability centers, we use $\hat{\pi}_S$, $\hat{\pi}_{P2}$, $\hat{\pi}_{Pr}$, $\hat{\pi}_A$, $\hat{\pi}_B$ and $\hat{\pi}_C$ with both the hybrid and the dynamic updating strategies. All tests are carried out on an Enterprise 1000 Ultra workstation (400 MHz).

Table 5 presents the average values on the cpu times required to solve the 14 instances, while Tables 6–9 report detailed results on the various strategies. Table 10 shows broader results for the

Table 6
Results with the hybrid strategy and the primal simplex

Pb.	Std col. gen. cpu time (s)	Stabilized column generation cpu time (s)						*Best* μ
		$\hat{\pi}_S$	$\hat{\pi}_{P2}$	$\hat{\pi}_A$	$\hat{\pi}_{Pr}$	$\hat{\pi}_B$	$\hat{\pi}_C$	
1.1	2013	**150**	224	220	635	345	403	13
1.2	868	418	199	**192**	1220	1307	966	5
2.1	11340	528	904	**270**	841	692	692	42
2.2	7522	835	856	787	779	538	**486**	15
3.1	12866	449	1682	1916	705	**217**	4040	59
3.2	28329	1339	**243**	1338	555	413	304	117
4.1	40634	1120	324	1025	507	176	**72**	564
4.2	19285	2946	2949	2786	241	173	**142**	136
5.1	50338	2500	2001	2654	494	200	**109**	462
5.2	29977	1803	5303	5157	399	**170**	1091	176
6.1	54187	777	696	214	493	225	**117**	463
6.2	41989	1638	1761	1569	330	263	**124**	339
7.1	43652	5794	4160	390	80	77	**35**	1247
7.2	41971	2605	1063	150	517	129	**72**	583

Table 7
Results with the dynamic strategy and the primal simplex

Pb.	Std col. gen. cpu time (s)	Stabilized column generation cpu time (s)						*Best* μ
		$\hat{\pi}_S$	$\hat{\pi}_{P2}$	$\hat{\pi}_A$	$\hat{\pi}_{Pr}$	$\hat{\pi}_B$	$\hat{\pi}_C$	
1.1	2013	192	158	**147**	608	348	404	14
1.2	868	**422**	471	477	1223	1267	969	2
2.1	11340	587	373	**365**	848	694	690	31
2.2	7522	**239**	299	288	773	542	488	31
3.1	12866	671	2033	1471	716	**212**	4065	61
3.2	28329	317	307	**199**	559	396	308	142
4.1	40634	489	527	239	484	168	**72**	564
4.2	19285	232	257	181	238	182	**144**	134
5.1	50338	346	334	170	499	198	**110**	458
5.2	29977	437	1944	1310	400	**170**	1097	176
6.1	54187	923	1740	440	498	225	**115**	471
6.2	41989	239	185	187	333	263	**123**	341
7.1	43652	418	5159	180	82	77	**34**	1284
7.2	41971	3277	1018	158	502	130	**72**	583

configuration using the dynamic strategy and the dual simplex with $\hat{\pi}_B$ as initial stability center. Finally, Figs. 3 and 4 display the convergence behavior of both the standard and the stabilized column generation approaches.

Table 8
Results with the hybrid strategy and the dual simplex

Pb.	Std col. gen. cpu time (s)	Stabilized column generation cpu time (s)						*Best* μ
		$\hat{\pi}_S$	$\hat{\pi}_{P2}$	$\hat{\pi}_A$	$\hat{\pi}_{Pr}$	$\hat{\pi}_B$	$\hat{\pi}_C$	
1.1	2013	**148**	233	241	553	367	465	14
1.2	868	396	218	**198**	1371	1165	1016	4
2.1	11340	518	1135	**261**	843	696	751	43
2.2	7522	1050	1045	1024	868	564	**478**	16
3.1	12866	516	1313	1142	639	**340**	854	38
3.2	28329	1967	**292**	1858	695	400	353	97
4.1	40634	1928	384	1706	595	154	**84**	484
4.2	19285	2152	1742	1733	396	288	**209**	92
5.1	50338	2839	4383	2650	775	230	**163**	309
5.2	29977	2997	2961	4756	535	**209**	1340	143
6.1	54187	863	833	247	1010	322	**181**	299
6.2	41989	1643	1764	1571	572	220	**152**	276
7.1	43652	1912	3850	377	1308	112	**41**	1065
7.2	41971	585	817	122	1161	189	**107**	392

Table 9
Results with the dynamic strategy and the dual simplex

Pb.	Std col. gen. cpu time (s)	Stabilized column generation cpu time (s)						*Best* μ
		$\hat{\pi}_S$	$\hat{\pi}_{P2}$	$\hat{\pi}_A$	$\hat{\pi}_{Pr}$	$\hat{\pi}_B$	$\hat{\pi}_C$	
1.1	2013	179	148	**141**	545	361	460	14
1.2	868	483	**351**	389	1376	1157	1008	2
2.1	11340	554	422	**294**	846	692	742	39
2.2	7522	**285**	327	287	867	559	475	26
3.1	12866	992	1630	1579	638	**330**	836	39
3.2	28329	299	317	**182**	669	403	372	156
4.1	40634	313	506	216	589	159	**87**	467
4.2	19285	244	264	220	391	274	**218**	88
5.1	50338	397	294	172	766	223	**169**	298
5.2	29977	360	1293	1232	536	**209**	1334	143
6.1	54187	835	825	264	1022	321	**177**	306
6.2	41989	248	207	168	577	216	**153**	274
7.1	43652	262	1222	155	1322	112	**42**	1039
7.2	41971	547	761	119	1175	191	**108**	389

5.1. The accelerating effect of the stabilized algorithm

All results show that the stabilized column generation approach clearly outperforms the standard version. In fact, it is seen in Section 3.2 that the average cpu time of the standard version is approximately 8 h (27 470 s). Meanwhile, in the worst case, the stabilization speeds up the solving process 17 times,

Table 10
Detailed results with the dynamic strategy and the dual simplex

Pb.	MP cpu (s)	CG cpu (s)	Total cpu (s)	Gen Col Nbr	Col Gen Itr Nbr	Major Itr Nbr
Standard column generation approach						
1.1	1735	270	2005	15286	410	1
1.2	677	179	856	13954	238	1
2.1	10668	647	11314	25304	902	1
2.2	7039	479	7517	28042	777	1
3.1	12813	40	12853	33498	1127	1
3.2	27227	1092	28320	37043	1497	1
4.1	40470	146	40616	58522	2423	1
4.2	19175	96	19270	38673	1419	1
5.1	49797	473	50270	59061	2569	1
5.2	29700	248	29949	53459	2108	1
6.1	53982	87	54069	66680	2939	1
6.2	41801	182	41982	65416	2527	1
7.1	43584	55	43638	68284	3174	1
7.2	41890	36	41926	60471	3234	1
Standard column generation approach						
1.1	55	298	353	2169	503	225
1.2	98	1035	1133	2893	1492	911
2.1	84	594	678	2912	917	451
2.2	145	404	549	3862	717	247
3.1	302	22	324	8291	1000	208
3.2	109	289	398	4348	445	80
4.1	135	20	155	4461	563	108
4.2	230	38	268	5372	807	249
5.1	167	53	220	4278	361	56
5.2	142	61	203	4414	659	155
6.1	309	10	319	6757	608	48
6.2	169	40	209	4675	879	267
7.1	109	2	111	3517	260	12
7.2	187	2	189	3481	305	38

with a corresponding figure of only 27 min (1636 s) average cpu time, as shown in Table 5. It can also be observed from the same table that $\hat{\pi}_B$ performs, in average, better than the other stability centers. In this case, the speed-up factor is approximately 75 times faster. The values in the last three columns assert that both updating strategies contribute equally to the solving process with $\hat{\pi}_{Pr}$, $\hat{\pi}_B$ or $\hat{\pi}_C$ as stability center depending on the technique used to solve the restricted *MP*. This is to stress that the high quality of the dual estimates plays an important role in the impressive factors we obtained.

The analysis of the effect of the initial dual solution on the quality of the stabilized process leaves no doubt that the convex combinations $\hat{\pi}_A$, $\hat{\pi}_B$ and $\hat{\pi}_C$ are very good approximations of the optimal solution. Nonetheless, on the basis of the results in Tables 6–9, the best cpu times are more often achieved with $\hat{\pi}_C$, which could partly be explained by the fact that $\hat{\pi}_C$ is a convex combination of three rather than two SDVSP dual solutions. Let the accelerating factor μ denotes the ratio between the total cpu time of

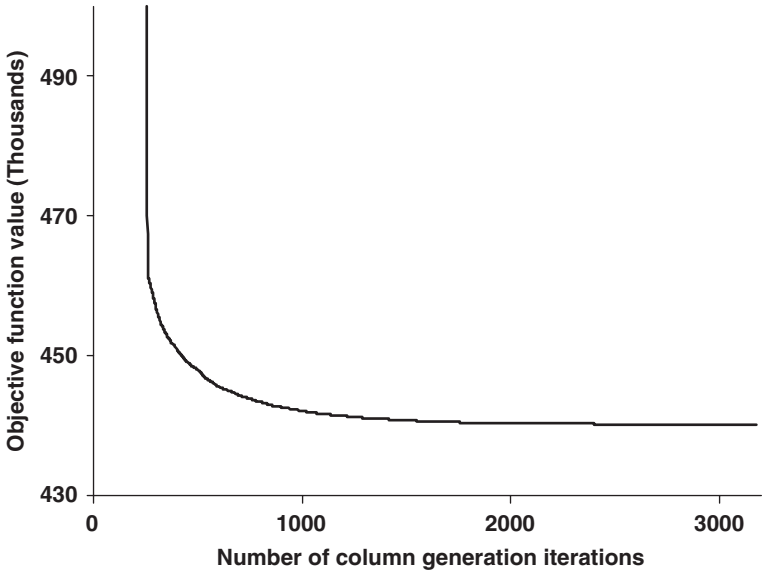

Fig. 3. Tail effect of the standard column generation.

the standard column generation over the total cpu time of the stabilized column generation, both on the most reduced networks. In Tables 6–9, the $Best\mu$ is the value computed with the cpu times provided by the best initial stability center. The values in bold are the best corresponding cpu times. Whatever is the updating strategy or the restricted *MP* solving technique, the stabilized column generation solves all test problems within less than 100 min cpu time (5794 s). The accelerating factor μ reaches its highest levels with problems of the 3-day horizon and over, i.e., the ones that the standard approach could barely solve. While μ does not exceed 42 with the problems 1.1 – 2.2, its value with problems of the 7-day horizon shows that stabilized column generation is 1280 times faster than the standard version. The stabilized approach manages to solve in 35 s the problem 7.1 that requires more than 32 h by the standard version.

For a better illustration of the outcome of the stabilized column generation approach, detailed results are presented in Table 10 for the configuration using the dynamic strategy and the dual simplex with $\hat{\pi}_B$ as the initial stability center. The columns contain the following information, given in that order: the time spent solving the restricted *MP* (in seconds), the time spent generating columns (in seconds), the total computing time (in seconds), the number of columns generated, the number of column generation iterations, and the number of major iterations required to the whole solving process of the LP relaxation of the long horizon MDVSP. The first observation is that there is a unique major iteration to solve *MP* by the standard column generation approach. On the other hand, the number of major iterations required by the stabilized approach varies between 12 and 911. The average number of column generation iterations per major iteration (or stabilized problems solved) is clearly very small (from 1.6 to 21.7 iterations). The most relevant impact of the proposed method appears on the MP cpu times (from 7 to 400 times faster) which influence the total cpu times. Indeed the dual estimates that we provide to the *MP* are good enough to restrict the dual space.

Finally, Figs. 3 and 4 illustrate the behavior of the standard and the proposed stabilized column generation approaches, respectively, on the instance 7.1. Fig. 3 shows a typical tail effect of the standard

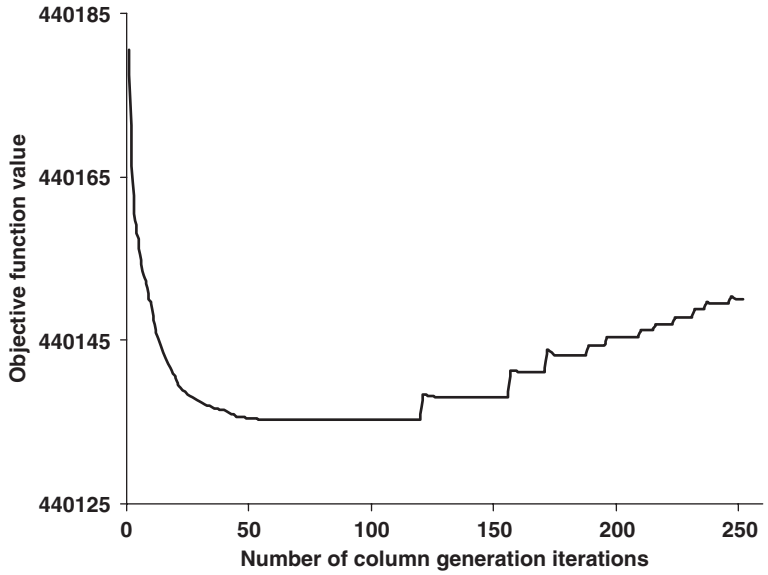

Fig. 4. Convergence behavior of the stabilized column generation.

column generation with more than 3000 iterations. In Fig. 4, the objective function value is already close to the optimum even with the first approximation of *MP* by a stabilized problem. This value is slightly below the optimum and 11 additional major iterations are required to reach optimality over a total of 260 column generation iterations only.

5.2. Concluding remarks

Through this paper, a cyclic logic is easily derived; the longer is the horizon, the more degenerate is the linear relaxation of the MDVSP tackled with the standard column generation approach but, at the same time, the closer it is to its SDVSP approximation. The closer is the SDVSP to its corresponding MDVSP, the better is its dual information. With a good dual information, degeneracy is overcome thanks to the stabilized column generation approach. The results we presented illustrate the great potential of such an approach in solving large scale problems. Prior to the application of the stabilizing approach, we investigated a network reducing approach that allows not only to compute an initial stability center but also to find a good approximation of the long horizon MDVSP integer solution. Even if such a contribution is specific to the MDVSP, the ideas can be extended to other combinatorial problems such as the bus driver scheduling and the airline crew pairing problems.

References

[1] Ben Amor H, Desrosiers J. A proximal trust-region algorithm for column generation stabilization. Computers & Operations Research, 2004, to appear.

[2] Bodin L, Rosenfield D, Kydes A. UCOST: a micro approach to a transit planning problem. Journal of Urban Anal. 1978;5: 46–69.

[3] Desrosiers J, Dumas Y, Solomon MM, Soumis F. Time constrained routing and scheduling. In: Ball MO., et al., editor. Handbooks in Operations Research and Management Science, Network Routing, Vol. 8. Amsterdam: Elsevier; 1995. p. 35–139.
[4] Bertossi AA, Carraresi P, Gallo G. On some matching problems arising in vehicle scheduling models. Networks 1987;17: 271–81.
[5] Carpaneto G, Della'Amico M, Fischetti M, Toth P. A branch and bound algorithm for the multiple vehicle scheduling problem. Networks 1989;19:531–48.
[6] Ribeiro CC, Soumis F. A column generation approach to the multiple depot vehicle scheduling problem. Operations Research 1994;42(1):41–52.
[7] Bianco L, Mingozzi A, Ricciardelli S. A set partitioning approach to the multiple depot vehicle scheduling problem. Optimization Methods and Software 1994;3:163–94.
[8] Forbes MA, Holt JN, Watts AM. An exact algorithm for multiple depot bus scheduling. European Journal of Operational Research 1994;72:115–24.
[9] Löbel A. Vehicle scheduling in public transit and lagrangean pricing. Management Science 1998;44:1637–49.
[10] Hadjar A, Marcotte O, Soumis F. A branch-and-cut algorithm for the multiple-depot vehicle scheduling problem. Operations Research, 2001, to appear.
[11] Desaulniers G, Desrosiers J, Ioachim I, Solomon MM, Soumis F, Villeneuve D. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In: Crainic TG, Laporte G, editors. Fleet management and logistics. Boston: Kluwer; 1998. p. 57–93.
[12] Lemaréchal C. Lagrangian relaxation. In: Jünger M, Naddef D, editors. Computational Combinatorial Optimization. Heidelberg: Springer; 2001. p. 115–60.
[13] Rockafellar RT. Monotone operators and the proximal point algorithm. SIAM Journal on Control and Optimization 1976;14:877–98.
[14] Martsen RE, Hogan WW, Blakenship JW. The Boxstep method for large-scale optimization. Operations Research 1975;23:389–405.
[15] Martsen RE. The use of Boxstep method in discrete optimization. Mathematical Programming Study 1975;3:127–44.
[16] Schramm H, Zowe J. A version of the bundle idea for minimizing a nonsmooth function: conceptual idea convergence analysis, numerical results. SIAM Journal on Optimization 1998;2:121–52.
[17] Neame P. Nonsmooth methods in integer programming, PhD dissertation, University of Melbourne, Australia, 1999.
[18] du Merle O, Villeneuve D, Desrosiers J, Hansen P. Stabilized column generation. Discrete Mathematics 1999;194: 229–37.
[19] Ben Amor H, Desrosiers J, Frangionni A. Stabilization in column generation. Les Cahiers du GERAD, G-2004-62, HEC Montréal, Canada, 2003.