International Journal of Pure and Applied Mathematics
Volume 103 No. 4 2015, 797-818
ISSN: 1311-8080 (printed version); ISSN: 1314-3395 (on-line version)
url: http://www.ijpam.eu
doi: http://dx.doi.org/10.12732/ijpam.v103i4.14

SUBGRADIENT OPTIMIZATION PRIOR TO COLUMN GENERATION — A MEANS FOR PREDICTING OPTIMAL COLUMNS

Andreas Westerlund[1], Maud Göthe-Lundgren[2],
Torbjörn Larsson[3] §, Di Yuan[4]
[1]Jeppesen Systems AB
Gothenburg, SWEDEN
[2,4]Department of Science and Technology
Linköping University, SWEDEN
[3]Department of Mathematics
Linköping University, SWEDEN

Abstract: We propose a two-phase combination of two optimization techniques, Lagrangian relaxation and column generation, with the aim of overcoming their respective drawbacks. In a prediction phase, subgradient optimization is used and the Lagrangian relaxed solutions found are used to initialize a master problem. In a solution phase, column generation is performed. We provide a validation of this two-phase method through an asymptotic result for the prediction phase and give guidelines for its truncated usage.

The two-phase method is assessed on a multicommodity network flow problem, for which it performs significantly better than a pure column generation method. We conclude that the subgradient optimization prediction phase can accelerate a column generation method considerably.

AMS Subject Classification: 90C05, 90C08, 90C06, 90-08
Key Words: linear programming, integer linear programming, subgradient

Received: August 16, 2015

§Correspondence author

optimization, column generation, Dantzig–Wolfe decomposition, multicommodity network flows

1. Introduction

A well-known way to approach a structured integer linear optimization problem is to exploit the structure in a price-directive decomposition scheme. Often a Lagrangian dual scheme is used, and we can distinguish between search methods and cutting plane methods for solving the dual problem. A popular search method for finding an approximate dual solution is subgradient optimization. Cutting plane methods for the dual problem are also well established, and they are dually equivalent to Dantzig–Wolfe decomposition and column generation applied to a convexified version of the integer problem. Both subgradient optimization and Dantzig–Wolfe decomposition produce lower bounds to the optimal value of the convexified problem (assuming minimization).

An advantage of subgradient optimization is that the move to the next dual iterate is inexpensive, once the Lagrangian relaxed problem, or subproblem, has been solved. A disadvantage is the lack of a good termination criterion. In practice the method is often terminated after a predetermined number of iterations. Another disadvantage of subgradient optimization is that feasible solutions are usually not easily obtained, neither to the integer problem nor to its convexified version, and therefore upper bounds to their optimal values are not easily available. It is however quite common that a Lagrangian heuristic [e.g., 15] is used to manipulate Lagrangian subproblem solutions into feasible solutions to the integer problem and upper bounds to its optimal value.

Advantages of Dantzig–Wolfe decomposition is that feasible solutions to the convexified problem are obtained and that it has finite convergence. The former fact enables early termination, when the gap between the upper and lower bounds to the optimal value of the convexified problem is small enough. A drawback is that each iteration is quite expensive, since a new dual iterate is computed through the reoptimization of a linear program. Another drawback is the inherent instability [e.g., 10, Ch. 15], in the sense that the values of the dual variables may change significantly between iterations, which may lead to a poor convergence behaviour.

To prevent this phenomenon, a stabilization mechanism can be introduced. One such mechanism can be viewed as the boxstep trust-region method [20] applied in the dual space. This stabilization has been used with good results by e.g. Larsson et al. [16] and Westerlund et al. [27]. A similar stabilization

technique is proposed by du Merle et al. [7]. It is successfully used in three types of applications.

Another possibility to improve the behaviour of a column generation scheme is to heuristically generate high-quality starting columns [e.g., 19, Sect. 4.1.1]. We propose a specific way of generating high-quality starting columns.

We design a two-phase method that benefits from the advantages of both subgradient optimization and column generation. A prediction phase uses subgradient optimization, during which the Lagrangian subproblem solutions are stored. At termination, a lower bound for the convexified problem and a number of subproblem solutions are at hand. In a following solution phase, these subproblem solutions are used to initialize the Dantzig–Wolfe master problem, whereafter column generation is performed as usual.

The prediction phase can be seen as a heuristic for setting up a high-quality initial master problem, so that fewer columns need to be generated in the solution phase. Alternatively, the solution phase can be thought of as to verify the outcome of the prediction phase. If the latter works perfectly, only one master problem needs to be solved for detecting optimality. We prove that with an appropriate choice of steplengths in the subgradient optimization in the prediction phase, optimal columns in the master problem are found asymptotically.

Some procedures similar to our two-phase method can be found in the literature. In [28], subgradient optimization is performed in a first phase, and in each iteration dual cuts are stored and a Dantzig–Wolfe master problem is solved. The master objective value is used as an upper bound in the Polyak steplength formula (which uses the relaxation parameter value one). If the same cut has been generated too many times, the method switches to a second phase, in which the Dantzig–Wolfe method is used. An advantage of this two-phase method is that it is free from parameters. A clear disadvantage is that a master problem is solved in each subgradient iteration, which can be computationally burdensome. The only information used from the master problem of the first phase is its objective value; in particular, the dual solution is not used.

The idea above is further developed by Vera et al. [26], who report on different criteria to decide when to switch from the first to the second phase. Further, the effect of using a bundle method instead of a linear programming approach in the second phase is investigated. A two-phase combination of subgradient optimization with a bundle method is studied also in [13].

Barahona and Jensen [5] apply Dantzig–Wolfe decomposition to a plant location problem, and every few iterations of this method they run a fixed number of subgradient optimization iterations, starting from the dual solution to the master problem. The master problem receives all subproblem solutions

found in these iterations, and the final dual values are used in the Dantzig–Wolfe subproblem. Substantial reductions in computing times are reported, compared to the standard Dantzig–Wolfe method. This line of research is continued in [11], where it is also discussed how to use subgradient optimization for finding approximate dual solutions to Dantzig–Wolfe master problems.

The contributions of our work are as follows. First, an asymptotic result for the prediction phase is shown. This result provides a theoretical basis for a two-phase method. Second, guidelines for practical usage are discussed. Third, computational experiments with multicommodity network flow problems show that the two-phase method can perform favourable compared to the standard Dantzig–Wolfe method.

The remainder of the paper is organized as follows. First, we give notations and outline some known results. In Section 3 the asymptotic result is given together with a summary of the two-phase method and a discussion of truncated usage. The application to multicommodity network flows is described in Section 4. Further, Section 5 presents results of some illustrative experiments. In Section 6, numerical results are given. Conclusions and directions for further research are pointed out in Section 7.

2. Preliminaries

The two-phase method is applicable to linear optimization problems and to convexified integer linear optimization problems. In order to make the presentation as general as possible, we consider the latter, with the problem stated as

$$\begin{array}{lll} [IP] \qquad \min & c^T x & \\ \text{s.t.} & Ax \geq b, & (1) \\ & x \in X \subset R^n, & \end{array}$$

where $b \in R^m$ and X is a finite set, say $X = \{x^1, \ldots, x^N\}$, with N typically being a huge number. The problem is assumed to be easier to solve if constraint (1) is relaxed. Let P be the convexified problem obtained if X is replaced by its convex hull. (If we were considering a linear optimization problem, then X would be assumed to be a polytope, with $\{x^1, \ldots, x^N\}$ being its extreme points.)

The Dantzig–Wolfe master problem of P is given by

$$\begin{aligned}
[MP] \qquad \min \quad & \sum_{j=1}^{N} \left(c^T x^j\right) \lambda_j \\
\text{s.t.} \quad & \sum_{j=1}^{N} \left(Ax^j\right) \lambda_j \geq b, && (2) \\
& \sum_{j=1}^{N} \lambda_j = 1, && (3) \\
& \lambda_j \geq 0, \qquad j = 1, \ldots, N.
\end{aligned}$$

Let u and w be dual variables associated with constraints (2) and (3), respectively. The linear programming dual problem is given by

$$\begin{aligned}
[DMP] \qquad \max \quad & b^T u + w \\
\text{s.t.} \quad & \left(Ax^j\right)^T u + w \leq c^T x^j, \quad j = 1, \ldots N, && (4) \\
& u \geq 0,
\end{aligned}$$

and it is equivalent to the Lagrangian dual problem

$$[LD] \qquad \max_{u \geq 0} \; h(u) = \min_{x \in X} \left\{ c^T x + u^T (b - Ax) \right\}.$$

Let U^* be its set of optimal solutions. Let $v(\cdot)$ be the optimal value for a problem. Then, since problem P is convex, $v(IP) \geq v(P) = v(MP) = v(DMP) = v(LD)$ holds.

A subgradient optimization method for problem LD starts from an arbitrary feasible dual point, u^0, and generates a sequence of dual iterates according to

$$u^{s+1} = \max\{0, u^s + t_s \gamma^s\}, \qquad s = 0, 1, \ldots, \tag{5}$$

where the maximum is taken component-wise, and t_s and γ^s are the steplength and the subgradient, respectively, used in iteration s. Here, $\gamma^s = b - Ax(u^s) \in \partial h(u^s)$, with $x(u^s)$ being the Lagrangian subproblem solution in iteration s. The following convergence result is shown in [14]; see also [6, Ch. 3, Th. 4.5].

Proposition 1. *Let the sequence $\{u^s\}$ be given by the iteration formula (5), where*

$$t_s > 0, \quad \lim_{s \to \infty} t_s = 0, \quad \sum_{s=0}^{\infty} t_s = \infty, \quad \textit{and} \quad \sum_{s=0}^{\infty} t_s^2 < \infty. \tag{6}$$

Then $\{u^s\} \to u^* \in U^*$.

The divergent series requirement (6) is not as restrictive as it might seem, since any steplength selection rule can easily be modified to fulfil (6) by replacing tentative steplengths with their projections onto the intervals $[\alpha/(s+1), \beta/(s+1)]$, where $\beta > \alpha > 0$. Because the projected steplengths are then trapped between two series that fulfil (6), this also holds for the projected steplengths.

Below we give a method for solving problem P by computing a weighted average of the subproblem solutions encountered when applying subgradient optimization to the Lagrangian dual problem. The method is outlined in [24, p. 117]. Larsson and Liu [14] proved that it asymptotically finds optimal primal solutions if the steplengths fulfil the condition (6). Both references consider application of Lagrangian relaxation to a linear optimization problem.

Let

$$\bar{x}(l) = \sum_{s=0}^{l} \mu_l^s x(u^s), \qquad l = 0, 1, \ldots, \tag{7}$$

where

$$\mu_l^s = \frac{t_s}{\sum_{k=0}^{l} t_k}, \qquad s = 0, \ldots, l.$$

The expression (7) defines a convex combination of subproblem solutions. This combination can alternatively be expressed as

$$\bar{x}(l) = \sum_{j=1}^{N} \lambda_j(l) x^j, \qquad l = 0, 1, \ldots, \tag{8}$$

where

$$\lambda_j(l) = \sum_{s \in S_j(l)} \mu_l^s, \qquad j = 1, \ldots, N, \tag{9}$$

with

$$S_j(l) = \{s = 0, \ldots, l : x(u^s) = x^j\}, \quad j = 1, \ldots, N,\ l = 0, 1, \ldots.$$

Results similar to the proposition below have been shown by Larsson and Liu [14] and Larsson et al. [17], for linear and convex optimization, respectively.

Proposition 2. *Let the steplengths in the dual subgradient procedure (5) applied to LD be chosen so that condition (6) is fulfilled. Let the sequences* $\{\bar{x}(l)\}$ *and* $\{\lambda(l)\}$ *be defined by formulas (7) and (9), respectively. Then each of the sequences* $\{\bar{x}(l)\}$ *and* $\{\lambda(l)\}$ *has at least one accumulation point, and any accumulation points of these sequences are optimal in* P *and* MP*, respectively.*

Since the subgradient optimization scheme is memoryless and any iterate can be considered to be the initial one, the result of this proposition holds also when the generation of the sequences $\{\bar{x}(l)\}$ and $\{\lambda(l)\}$ is delayed for an arbitrary number of iterations, with the formulas (7) and (9) modified accordingly.

A clear drawback of the method implied by the proposition is that when the subgradient optimization scheme is terminated finitely, the averaged primal solution is typically only near-feasible (but also near-optimal, disregarding its infeasibility), which is due to the inherent dual character of the scheme. The practical use of such a solution is thus not immediate. For some applications, small primal infeasibilities may not matter. In others, tailored heuristics can be used to modify the near-feasible solution into a feasible and near-optimal solution; see e.g. [14] on this topic. Some further related works are [23, 17, 4, 22, 1, 9].

Below, a slightly different strategy is employed. Subgradient optimization is performed and Lagrangian subproblem solutions are collected, but instead of assigning predetermined weights to these, as above, a restricted Dantzig–Wolfe master problem is constructed and solved in order to compute optimal weights. (Feasibility of this master problem can be assured by a proper use of starting columns.) If needed, column generation is then performed until overall optimality (or sufficient near-optimality) is reached.

3. The Two-Phase Method

We here develop the use of subgradient optimization to predict optimal basic columns in a Dantzig–Wolfe master problem. First, an asymptotic result is given. Then the method is outlined. Finally we discuss practical issues for truncated usage.

Let $\underline{s}$ be a nonnegative integer, define the set

$$J_{\underline{s}} = \{j = 1, \ldots, N : x(u^s) = x^j \text{ for at least one } s \geq \underline{s}\},$$

and consider the restricted master problem

$$\begin{aligned} [MP(\underline{s})] \qquad \min \quad & \sum_{j \in J_{\underline{s}}} \left(c^T x^j\right) \lambda_j \\ \text{s.t.} \quad & \sum_{j \in J_{\underline{s}}} \left(A x^j\right) \lambda_j \geq b, \\ & \sum_{j \in J_{\underline{s}}} \lambda_j = 1, \end{aligned}$$

$$\lambda_j \geq 0, \qquad j \in J_{\underline{s}}.$$

Theorem 3. *Let the steplengths in the dual subgradient procedure (5) applied to LD be chosen so that condition (6) is fulfilled, and let the sequence* $\{\lambda(l)\}$ *be defined by formula (9). Let* u^* *be the limit point for the sequence* $\{u^s\}$*, according to Proposition 1, and let* w^* *solve DMP for* $u = u^*$*. Define the optimal reduced costs* $\bar{c}_s = (c^T - u^{*T}A)x(u^s) - w^*$*. Then* $v(MP(\underline{s})) = v(P)$ *holds for any nonnegative integer* $\underline{s}$*. Further,* $\bar{c}_s = 0$ *holds for every* s *that is sufficiently large.*

Proof. According to Proposition 2, the sequence $\{\lambda(l)\}$ has an accumulation point, say $\tilde{\lambda}$, which is optimal in MP. If $\tilde{\lambda}_j > 0$, then $x(u^s) = x^j$ holds for at least one $s \geq \underline{s}$, which in turn implies that the problem $MP(\underline{s})$ contains the variable λ_j. Then the solution $\lambda_j = \tilde{\lambda}_j$, $j \in J_{\underline{s}}$, is feasible in $MP(\underline{s})$, and it follows that

$$v(MP(\underline{s})) \leq \sum_{j \in J_{\underline{s}}} \left(c^T x^j\right) \tilde{\lambda}_j = \sum_{j=1}^{N} \left(c^T x^j\right) \tilde{\lambda}_j = v(MP).$$

Further, since $v(MP(\underline{s})) \geq v(MP)$, we obtain that

$$v(MP(\underline{s})) = v(MP) = v(P).$$

From the equivalence between DMP and LD follows that

$$h(u^*) = b^T u^* + w^*. \tag{10}$$

Since the Lagrangian dual objective function h is polyhedral, $x(u^s)$ solves the Lagrangian subproblem for both $u = u^*$ and $u = u^s$ when s is large enough. Hence, both $\gamma^s \in \partial h(u^s)$ and $\gamma^s \in \partial h(u^*)$ hold, so that

$$\begin{aligned} h(u^*) &= h(u^s) + (b - Ax(u^s))^T(u^* - u^s) \\ &= c^T x(u^s) - u^{sT} Ax(u^s) + u^{sT} b + (b - Ax(u^s))^T(u^* - u^s) \\ &= c^T x(u^s) - u^{*T} Ax(u^s) + b^T u^*, \end{aligned}$$

which together with (10) yields that

$$c^T x(u^s) - u^{*T} Ax(u^s) - w^* = 0,$$

which proves the second conclusion. □

According to the theorem, all subproblem solutions needed to construct a master problem that solves problem P will eventually be found by the subgradient optimization, no matter when the construction of the master problem begins

(that is, subproblem solutions from early iterations are never indispensable). Further, in case of dual non-degeneracy, late iterations will only provide subproblem solutions that give optimal basic columns in the master problem. (In case of dual degeneracy, non-basic columns with zero reduced costs can also be obtained.) These results justify the use of subgradient optimization as a means for predicting optimal basic columns, before applying column generation.

Proposition 2 provides solution procedures that asymptotically solve problem P or MP. Finitely generated solutions are however typically infeasible in the respective problems. In contrast, a finitely generated problem $MP(\underline{s})$ can be used for finding feasible and also near-optimal solutions to problem MP (and P). The important difference between the averaged solution $\bar{x}(l)$ and the solution to P obtained from a finitely generated problem $MP(\underline{s})$, is, clearly, that the former is defined by the expression (8) with the convexity weights given a priori by formula (9), while in the latter case the values of the convexity weights are optimized with respect to the objective in P and the constraint (1).

Our two-phase method works as follows. In the prediction phase of the method, $\bar{s}$ subgradient optimization iterations are performed, with steplengths computed according to a rule such that (6) holds. Subproblem solutions are collected from iteration $\underline{s} \leq \bar{s}$. On termination of the prediction phase, a lower bound $\underline{v}(LD)$ to $v(LD)$ and a collection of subproblem solutions have been obtained. First in the solution phase, an initial master problem is constructed, using the subproblem solutions already collected. (Other starting columns may of course also be included.) Thereafter, column generation in a Dantzig–Wolfe fashion is performed, until the relative gap between upper and lower bounds on the optimal value does not exceed $\varepsilon \geq 0$. [The lower bound is initialized to $\underline{v}(LD)$.]

We last discuss practical considerations of a truncated use of the solution procedure suggested by Theorem 3. Ideally, one would like to find all optimal basic columns in MP, which corresponds to finding all constraints (4) that are fulfilled with equality at an optimal solution to DMP, that is, the segments of the Lagrangian dual function that are active at an optimal solution. (We ignore the issue of degeneracy.) In order to find these constraints (or most of them) in a finite number of iterations, the subgradient steplengths need to ensure that the dual iterates reach near-optimality, that is, come close to u^* (as defined in Proposition 1), while they also oscillate around u^*. Below we discuss the consequences of failing to meet either of these two requirements. In this respect there are two pitfalls: the steplengths can be too small or too large.

The two pitfalls are illustrated on the fictitious Lagrangian dual problem in Figure 1. Suppose first that the subgradient optimization steplengths decrease

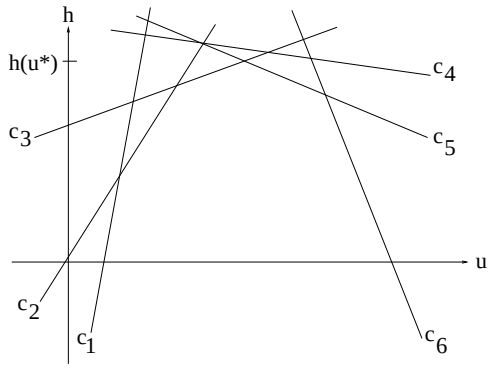

Figure 1: A one-dimensional example of DMP with six constraints

too rapidly. Then the dual iterates may never reach the vicinity of u^* or they may approach this point without oscillating around it, and therefore all constraints (4) needed to solve problem DMP may not be found. If for example $u^0 = 0$ and the steplengths are too small, then maybe only the segments c_1 and c_2 have been visited at termination (in which case $MP(\underline{s})$ becomes infeasible). Suppose instead that the steplengths decrease too slowly. Then the dual iterates may never come close to u^* and the constraints needed to solve DMP may therefore never be found. If again $u^0 = 0$, it can in this case in the example happen that only the segments c_1 and c_6 have been visited at termination.

In either of these cases the quality of the problem $MP(\underline{s})$ may be poor. In order to make an accurate truncated prediction of optimal basic columns, the subgradient optimization steplengths must thus be chosen so that the iterates reach the vicinity of u^*, while they also need to decrease slowly enough to cause the iterates to oscillate around u^*. The first requirement coincides with the usual one in subgradient optimization, when the goal is to compute a strong lower bound, while the second one is a supplement.

4. The Multicommodity Network Flow Problem

The two-phase is expected to be advantageous in applications where the master problem is computationally burdensome, since the prediction phase should reduce the number of master problems to be solved. If the subproblems are inexpensive, it also favours the two-phase method, since high-quality columns can be found if more subgradient iterations can be allowed. With subproblems that are instead costly, it is likely that too much computing time will be used

in the prediction phase and the two-phase method is not promising.

We perform experiments with a linear optimzation problem with the former characteristics, namely a linear minimum-cost multicommodity network flow (MCNF) problem where each commodity is given by an origin–destination (OD) pair. The problem can be described by an arc-node formulation or by a path formulation.

Consider a directed graph $G = (N, A)$, with N and A being the sets of nodes and directed arcs, respectively. The sets of arcs emanating and terminating at node i are denoted by A_i^+ and A_i^-, respectively. Let K be the set of commodities. Let $o(k)$ and $d(k)$ be the origin and the destination, respectively, for commodity k, and let d_k be the demand. Further, c_{ka} denotes the unit cost of sending flow of commodity k on arc a and u_a denotes a total flow capacity on the arc. With variable x_{ka} denoting the flow of commodity k on arc a, the arc-node formulation is as follows.

$$
\begin{aligned}
\min \quad & \sum_{k\in K}\sum_{a\in A} c_{ka}x_{ka} \\
\text{s.t.} \quad & \sum_{a\in A_i^+} x_{ka} - \sum_{a\in A_i^-} x_{ka} = \begin{cases} d_k \text{ if } \ i = o(k), \\ -d_k \text{ if } \ i = d(k), \\ 0 \text{ otherwise,} \end{cases} \quad i \in N,\ k \in K, \\
& \sum_{k\in K} x_{ka} \le u_a, \qquad a \in A, \\
& x_{ka} \ge 0, \qquad a \in A,\ k \in K
\end{aligned}
$$

The number of flow variables is clearly $|A| \cdot |K|$, and it typically grows fast with the size of the network. If the flow cost does not vary by commodity, the number of variables can be reduced considerably by aggregation of commodities with respect to their source (or, alternatively, destination) nodes. Such aggregation may however impair the performance of some methods, and in particular Dantzig–Wolfe decomposition [12].

In the path formulation of MCNF, we let P_k be the set of elementary paths from $o(k)$ to $d(k)$, and let the parameter $\delta_{kpa} = 1$ if path $p \in P_k$ uses arc a and $\delta_{kpa} = 0$ otherwise. With c_{kp} being the cost of path p, that is, $c_{kp} = \sum_{a\in A} \delta_{kpa} c_{ka}$, and the variable h_{kp} being the flow on path $p \in P_k$, the problem can be stated as below.

$$
\begin{aligned}
\text{[MCNF–P]} \qquad \min \quad & \sum_{k\in K}\sum_{p\in P_k} c_{kp}h_{kp} \\
\text{s.t.} \quad & \sum_{p\in P_k} h_{kp} = d_k, \qquad k \in K,
\end{aligned}
$$

$$\sum_{k\in K}\sum_{p\in P_k} \delta_{kpa} h_{kp} \le u_a, \qquad a \in A,$$
$$h_{kp} \ge 0, \qquad p \in P_k,\ k \in K$$

The two formulations are tightly related through Dantzig–Wolfe decomposition. When applying this method to the arc-node formulation, the subproblem separates over commodities. For commodity k, an optimal subproblem solution is an extreme point of the set defined by the flow conservation and nonnegativity constraints for the commodity, and it describes a flow of strength d_k along a path in P_k that is shortest with respect to the subproblem cost function. This generation of columns to the Dantzig–Wolfe master problem is equivalent to a direct application of column generation to the formulation MCNF–P. In fact, the master problem is obtained from MCNF–P by introducing new variables that equal h_{kr}/d_k. The equivalence between MCNF–P and a Dantzig–Wolfe reformulation of the arc-node formulation of MCNF was observed in [25].

It was early recognized [e.g., 2] that price-directive decomposition is a promising approach for designing efficient solution methods for the MCNF problem. Straightforward Dantzig–Wolfe decomposition, that is, a cutting plane approach to the Lagrangian dual problem, is however not competitive, and several researchers have proposed other strategies for solving the Lagrangian dual problem. A bundle method is presented in [8], and in [18] it is proposed to use an augmented Lagrangian function to improve the rate of convergence in the dual space. In [3], encouraging results of applying a proximal analytic center cutting plane method are reported.

We have used two sets of test problems, which are among those used in [18] and [3]. For both sets, the unit flow cost varies by arc but not by commodity. The first set comprises planar networks, with nodes randomly generated in the plane and arc costs that are the Euclidean distances. Arcs are created in an ascending order of Euclidean distance, and so that the network becomes planar. The origins and destinations of commodities are randomly chosen pairs of nodes. The arc capacities and commodity demands are uniformly distributed. The networks in the second problem set have a grid topology. For an $n \times n$ grid, the network has n^2 nodes and $4n(n-1)$ directed arcs. Capacities and commodities are generated as for the planar networks, but the arc costs have a uniform distribution.

Table 1 gives properties of the problem instances. The first two columns give the name of the instance and the numbers of nodes, arcs and OD-pairs. The optimal value is z^*. We also compute the optimal value that is obtained if all arc capacity constraints are relaxed, denoted $\tilde{z}$. The ratio $\tau = (z^* -$

$\tilde{z})/z^*$ measures how restrictive the arc capacities are. Further, let ρ be the portion of the capacity constraints that have non-zero dual variables when the final master problem has been solved. This value is another measure of how restrictive the arc capacities are. Finally, ς is the number of paths used in the final master problem solution divided by the number of OD-pairs, that is, the average number of paths used per OD-pair in an optimal solution. We

Table 1: Characteristics of original instances

name	$\|N\|, \|A\|, \|K\|$	z^*	$\tau\,(\%)$	$\rho\,(\%)$	ς
planar100	100, 532, 1085	2.3134×10^8	7.10	16.35	1.07
planar150	150, 850, 2239	5.4809×10^8	15.91	27.76	1.10
planar300	300, 1680, 3584	6.8998×10^8	2.99	7.68	1.03
planar500	500, 2842, 3525	4.8198×10^8	1.21	1.90	1.01
grid8	625, 2400, 500	4.1711×10^7	0.92	12.25	1.59
grid9	625, 2400, 1000	8.2653×10^7	1.15	16.88	1.41
grid10	625, 2400, 2000	1.6411×10^8	1.16	16.92	1.21
grid11	625, 2400, 4000	3.2926×10^8	0.73	10.75	1.07
grid12	900, 3480, 6000	5.7719×10^8	0.39	5.95	1.03
grid13	900, 3480, 12000	1.1593×10^9	0.50	7.87	1.02

have also made experiments on a set of modified instances, obtained from the instances planar500 and grid11. The modified instances are constructed by scaling the demand in each OD-pair by a given factor. For example, the instance planar500_2.5 is obtained from the instance planar500 by scaling all demands by the factor 2.5. With larger demands, the flow will be more congested, more arc capacities will be saturated and the optimal objective value will increase, as can be seen in Table 2.

5. An Illustration of Truncated Performance

Below we report on experiments that illustrate truncated usage of subgradient optimization as a means for predicting optimal columns, and especially how the quality of the columns found depends on the choice of parameter values. For this purpose, the instance grid8 is used. All input and output parameters are given in Table 3. (Here, D–W stands for Dantzig–Wolfe. Some of the parameters are not used until later.) The subgradient steplength in iteration s

Table 2: Characteristics of modified instances

name	z^*	$\tau\,(\%)$	$\rho\,(\%)$	ς
planar500_1.5	7.3904×10^8	3.36	4.79	1.04
planar500_2.0	1.0137×10^9	6.06	8.27	1.06
planar500_2.5	1.3036×10^9	8.69	11.82	1.09
planar500_3.0	1.6107×10^9	11.32	15.31	1.12
planar500_3.5	1.9355×10^9	13.90	18.26	1.15
planar500_4.0	2.2809×10^9	16.50	21.71	1.17
grid11_0.75	2.4596×10^8	0.34	5.29	1.03
grid11_1.25	4.1346×10^8	1.19	17.08	1.10
grid11_1.5	4.9872×10^8	1.69	25.21	1.15

is $t_s = a/s$, where a is a positive parameter. For the experiments reported in this section, $a = 1.0$ is used.

In a first experiment, the quality of the prediction is measured in terms of the number of columns that are necessary to generate in the solution phase in order to reach optimality. Figure 2 shows how the quality of the column prediction depends of $\underline{s}$ and σ. Each curve in the figure corresponds to a given value of σ. Two patterns can be seen. First, for a given value of $\underline{s}$, a more accurate prediction is observed for higher values of σ. Second, for a given value of σ it is typically better to do many subgradient iterations before beginning to collect columns.

In another experiment, the value of σ was fixed to 20 and four runs were made with $\underline{s} = 1$, 100, 200, and 400, respectively. The objective value of the first master problem was compared with the optimal value, and a relative gap was computed. The resulting relative gaps were 0.25%, 0.03%, 0.007%, and 0.002%, respectively. In a similar experiment, $\underline{s}$ was fixed to 150 and five runs were made with $\sigma = 5$, 10, 15, 20, and 25, which gave relative gaps of 0.14%, 0.04%, 0.02%, 0.01%, and 0.007%, respectively.

We conclude that the quality of the predicted columns improves with increasing values of $\underline{s}$ and σ, as expected. There is of course a trade-off between the quality of the prediction and its computational cost. With a large value of $\underline{s}$, many subgradient iterations are performed, which can become costly. Also, with a large value of σ, the first master problem will contain more columns, which can make the solution phase more costly.

Table 3: Input and output parameters

a	Controls the subgradient steplength, $t_s = a/s$.
$\bar{s}$	Number of subgradient iterations to perform.
$\underline{s}$	Controls from which iteration columns are collected.
σ	Number of subgradient iterations to collect columns.
ε	Relative gap tolerance.
ρ	Percentage of binding arc capacity constraints.
DW	Number of D–W iterations performed.
cg	Number of columns generated in solution phase.
$cg/\lvert K\rvert$	cg divided by the number of OD-pairs.
$time$	CPU time of two-phase method versus D–W.
$p1/p2$	Ratio of the CPU times of the two phases.
ς	Average number of paths used per OD-pair.

6. Numerical Experiments

We here compare the performance of the two-phase method to that of the standard Dantzig–Wolfe method on the MCNF. The aim is only to show that the prediction phase can significantly accelerate the column generation, and not to compare with other solution approaches for this application.

After some initial experiments, the parameter values $\bar{s} = 200$ and $\underline{s} = 190$ were fixed, giving $\sigma = 10$. The value of the steplength parameter a was calibrated for different instaces. For all grid instances, $a = 0.01$ is used. For all planar instances, except for planar500, $a = 10.0$ is used. For planar500 and for the modified planar500 instances, $a = 1.0$ and $a = 5.0$ are used, respectively.

The results are presented in Tables 4 to 7. The contents of the columns in the tables are given in Table 3. In Tables 4 and 5, numerical results with solution accuracy $\varepsilon = 10^{-6}$ are presented. In Tables 6 and 7, the solution accuracy is $\varepsilon = 10^{-3}$.

From the CPU time ratios in Table 4 we conclude that the two-phase method performs well compared to the standard Dantzig–Wolfe method, especially for the larger instances. Further, it is the solution phase of the two-phase method that is most computationally costly, again especially for the larger instances.

To study the effect of the tightness of the arc capacities, we used the set of modified instances. As can be seen in Tables 2 and 5, for higher demands

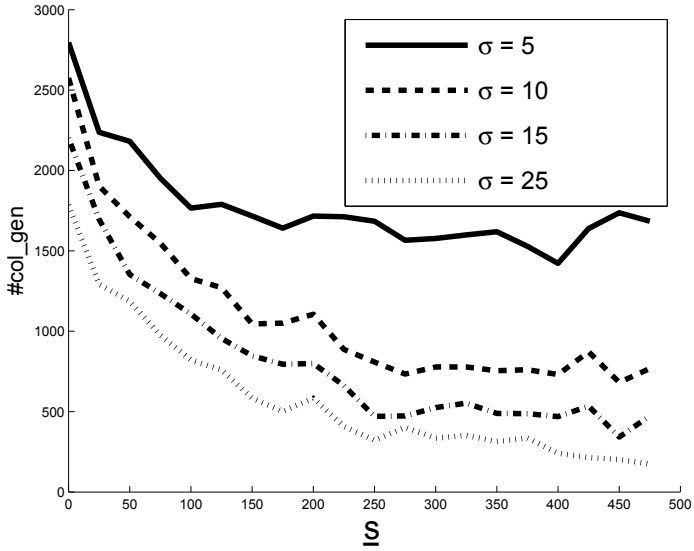

Figure 2: Number of columns generated in the solution phase as a function of $\underline{s}$ and σ for accuracy $\varepsilon = 10^{-6}$

in the OD-pairs, the value of ρ increases, the number of paths needed in each OD-pair increases, the advantage of using the two-phase method becomes more and more evident, and the fraction of the CPU time spent in the second phase increases. (The last effect is due to that the CPU time in the first phase is almost unchanged while the CPU time in the second phase grows.)

Table 6 shows results for the solution accuracy $\varepsilon = 10^{-3}$. When comparing with Table 4, some differences can be seen. Considering the values of *cg* for the planar instances, somewhat smaller numbers are obtained in Table 6. For the grid instances, however, the values of *cg* are significantly smaller. For the instances grid11, grid12 and grid13, all columns that are needed for solving the problems to an accuracy of $\varepsilon = 10^{-3}$ are found in the prediction phase.

When comparing the values of *cg* in Tables 5 and 7, it can be seen that there are no large differences for the modified planar500 instances. For the grid11_1.25 and grid11_1.5 instances, however, the prediction phase was successful in finding the columns needed to solve the problem for the solution accuracy $\varepsilon = 10^{-3}$.

Table 4: Results for original instances, with accuracy $\varepsilon = 10^{-6}$

name	*method*	*DW*	*cg*	$cg/\lvert K\rvert$	*time* (%)	$p1/p2$ (%)
planar100	D–W	9	3336	3.07		
	2-phase	4	1057	0.97	62	40
planar150	D–W	14	14764	6.59		
	2-phase	6	2824	1.26	21	10
planar300	D–W	9	13922	3.88		
	2-phase	5	2402	0.67	53	22
planar500	D–W	8	7468	2.12		
	2-phase	4	2039	0.58	98	48
grid8	D–W	16	3750	7.50		
	2-phase	8	1516	3.03	50	26
grid9	D–W	16	8068	8.07		
	2-phase	7	2993	2.99	33	13
grid10	D–W	14	14170	7.08		
	2-phase	6	4971	2.49	25	6.8
grid11	D–W	11	18936	4.73		
	2-phase	3	3562	0.89	25	7.6
grid12	D–W	10	23161	3.86		
	2-phase	4	5160	0.86	25	13
grid13	D–W	10	52887	4.41		
	2-phase	3	7616	0.63	17	3.3

7. Conclusions and Further Research

We have presented a two-phase method for optimization problems that benefit from price-directive decomposition. The method comprises a prediction phase, using subgradient optimization, and a solution phase, using column generation in a Dantzig–Wolfe environment. The combination is founded on an asymptotic result for the prediction phase. A truncated usage of the prediction phase can yield an initial Dantzig–Wolfe master problem of high quality, resulting in a shorter computing time for the overall method. In order to benefit from the use of the two-phase method, the subproblems must be computationally inexpensive compared to the master problems, since a relatively large number of subgradient optimization iterations needs to be performed.

Table 5: Results for modified instances, with accuracy $\varepsilon = 10^{-6}$

name	*method*	*DW*	*cg*	$cg/\lvert K\rvert$	*time* (%)	$p1/p2$ (%)
planar500_1.5	D–W	11	13892	3.94		
	2-phase	5	2747	0.78	49	30
planar500_2.0	D–W	12	21188	6.01		
	2-phase	5	3338	0.95	36	14
planar500_2.5	D–W	15	29305	8.31		
	2-phase	7	7830	2.22	25	7.1
planar500_3.0	D–W	16	36247	10.28		
	2-phase	6	8196	2.33	18	4.5
planar500_3.5	D–W	18	44730	12.69		
	2-phase	8	10237	2.90	17	2.8
planar500_4.0	D–W	19	51006	14.47		
	2-phase	8	19041	5.40	18	1.7
grid11_0.75	D–W	9	11483	2.87		
	2-phase	5	3009	0.75	39	20
grid11_1.25	D–W	13	26304	6.58		
	2-phase	5	6390	1.60	20	3.7
grid11_1.5	D–W	15	35115	8.78		
	2-phase	7	11177	2.79	19	1.5

The two-phase method is applied to a multicommodity network flow problem, which has the characteristics that should be advantageous for the method, and our numerical experiments show that it indeed performs favourable compared to standard Dantzig–Wolfe decomposition, in terms of computing times. Of particular interest is that the reduction in computing time tends to increase with the size of the instance and the tightness of the arc capacities, that is, for instances that are more computationally demanding.

A drawback of the two-phase method is its several control parameters. Suitable values for most of these can, however, be chosen according to simple rules. Our experience is that only a single parameter needs more careful calibration, namely the one that controls the steplengths in the subgradient optimization scheme. A topic for further research would be to try to incorporate into the framework of our two-phase method a subgradient algorithm without any control parameter that needs to be calibrated.

Table 6: Results for original instances, with accuracy $\varepsilon = 10^{-3}$

name	*method*	*DW*	*cg*	$cg/\|K\|$	*time* (%)	$p1/p2$ (%)
planar100	D–W	6	3278	3.02		
	2-phase	3	1057	0.97	62	40
planar150	D–W	10	14246	6.36		
	2-phase	3	2703	1.21	21	11
planar300	D–W	7	13763	3.84		
	2-phase	3	2357	0.66	52	22
planar500	D–W	5	6780	1.92		
	2-phase	1	1770	0.50	102	48
grid8	D–W	9	2913	5.83		
	2-phase	1	395	0.79	35	44
grid9	D–W	9	6133	6.13		
	2-phase	1	851	0.85	26	21
grid10	D–W	8	10726	5.36		
	2-phase	1	1733	0.87	21	9.4
grid11	D–W	6	14853	3.71		
	2-phase	0	0	0.00	18	10
grid12	D–W	5	17613	2.94		
	2-phase	0	0	0.00	26	15
grid13	D–W	5	38769	3.23		
	2-phase	0	0	0.00	16	4.2

References

[1] K.M. Anstreicher, L.A. Wolsey, Two "well-known" properties of subgradient optimization, *Mathematical Programming, Series B*, **129** (2009), 213–220.

[2] A. Assad, Solving linear multicommodity flow problems, in: *Proceedings of IEEE International Conference on Circuits and Computers*, N.G. Rabbat, ed., vol. **1** (1980), pp. 157–161.

[3] F. Babonneau, O. du Merle, J.-P. Vial, Solving large scale linear multicommodity flow problems with an active set strategy and proximal-ACCPM, *Operations Research*, **54** (2006), 184–197.

Table 7: Results for modified instances, with accuracy $\varepsilon = 10^{-3}$

name	*method*	*DW*	*cg*	$cg/\lvert K\rvert$	*time* (%)	$p1/p2$ (%)
planar500_1.5	D–W	8	13337	3.78		
	2-phase	3	2572	0.73	50	30
planar500_2.0	D–W	9	20276	5.75		
	2-phase	3	3320	0.94	34	14
planar500_2.5	D–W	11	26845	7.62		
	2-phase	4	6931	1.97	27	7.1
planar500_3.0	D–W	12	31245	8.86		
	2-phase	3	6211	1.76	18	4.2
planar500_3.5	D–W	14	37849	10.74		
	2-phase	4	7036	2.00	15	3.0
planar500_4.0	D–W	16	45482	12.90		
	2-phase	4	9981	2.83	16	1.8
grid11_0.75	D–W	4	8509	2.13		
	2-phase	1	1625	0.41	36	23
grid11_1.25	D–W	7	19374	4.84		
	2-phase	0	0	0.00	15	5.4
grid11_1.5	D–W	9	26463	6.62		
	2-phase	0	0	0.00	11	3.0

[4] F. Barahona, R. Anbil, The volume algorithm: producing primal solutions with a subgradient method, *Mathematical Programming, Series A*, **87** (2000), 385–399.

[5] F. Barahona, D. Jensen, Plant location with minimum inventory, *Mathematical Programming*, **83** (1998), 101–111.

[6] V.F. Dem'yanov, L.V. Vasil'ev, *Nondifferentiable Optimization*, Optimization Software, Inc., New York, NY (1985).

[7] O. du Merle, D. Villeneuve, J. Desrosiers, P. Hansen, Stabilized column generation, *Discrete Mathematics*, **194** (1999), 229–237.

[8] A. Frangioni, G. Gallo, A bundle type dual-ascent approach to linear multicommodity min-cost flow problems, *Informs Journal on Computing*, **11** (1999), 370–393.

[9] E. Gustavsson, M. Patriksson, A.-B. Strömberg, Primal convergence from dual subgradient methods for convex optimization, *Mathematical Programming*, **150** (2015), 365–390.

[10] J.-B. Hiriart-Urruty, C. Lemaréchal, *Convex Analysis and Minimization Algorithms II: Advanced Theory and Bundle methods*, Springer–Verlag, Berlin (1993).

[11] D. Huisman, R. Jans, M. Peeters, A.P.M. Wagelmans, Combining column generation and Lagrangian relaxation, in: *Column Generation*, G. Desaulniers, J. Desrosiers, M.M. Solomon, eds., Springer, New York, NY (2005), Ch. 9, pp. 247–270.

[12] K.L. Jones, I.J. Lustig, J.M. Farvolden, W.B. Powell, Multicommodity network flows: The impact of formulation on decomposition, *Mathematical Programming*, **62** (1993), 95–117.

[13] N. Kohl, O.B.G. Madsen, An optimization algorithm for the vehicle routing problem with time windows based on Lagrangian relaxation, *Operations Research*, **45** (1997), 395–406.

[14] T. Larsson, Z. Liu, A Lagrangean relaxation scheme for structured linear programs with applications to multicommodity network flows, *Optimization*, **40** (1997), 247–284.

[15] T. Larsson, M. Patriksson, Global optimality conditions for discrete and nonconvex optimization-with applications to lagrangian heuristics and column generation, *Operations Research*, **54** (2006), 436–453.

[16] T. Larsson, M. Patriksson, C. Rydergren, A column generation procedure for the side constrained traffic equilibrium problem, *Transportation Research Part B*, **38** (2004), 17–38.

[17] T. Larsson, M. Patriksson, A.-B. Strömberg, Ergodic, primal convergence in dual subgradient schemes for convex programming, *Mathematical Programming*, **86** (1999), 283–312.

[18] T. Larsson, D. Yuan, An augmented Lagrangian algorithm for large scale multicommodity routing, *Computational Optimization and Applications*, **27** (2004), 187–215.

[19] M.E. Lübbecke, J. Desrosiers, Selected topics in column generation, *Operations Research*, **53** (2005), 1007–1023.

[20] R.E. Marsten, W.W. Hogan, J.W. Blackenship, The boxstep method for large-scale optimization, *Operations Research*, **23** (1975), 389–405.

[21] B.T. Polyak, Minimization of unsmooth functionals, *USSR Computational Mathematics and Mathematical Physics*, **9** (1969), 14–29.

[22] A. Ruszczyński, A merit function approach to the subgradient method with averaging, *Optimization Methods and Software*, **23** (2008), 161–172.

[23] H. Sherali, G. Choi, Recovery of primal solutions when using subgradient optimization methods to solve Lagrangian duals of linear programs, *Operations Research Letters*, **19** (1996), 105–113.

[24] N.Z. Shor, *Minimization Methods for Non-Differentiable Functions*, translated from the Russian by K.C. Kiwiel and A. Ruszcyński, Springer–Verlag, Berlin (1985).

[25] J.A. Tomlin, Minimum-cost multicommodity network flows, *Operations Research*, **14** (1966), 45–51.

[26] J. Vera, A. Weintraub, M. Koenig, G. Bravo, M. Guignard, F. Barahona, A Lagrangian relaxation approach for a machinery location problem in forest harvesting, *Pesquisa Operacional*, **23** (2003), 111–128.

[27] A. Westerlund, M. Göthe-Lundgren, T. Larsson, A stabilized column generation scheme for the traveling salesman subtour problem, *Discrete Applied Mathematics*, **154** (2006), 2212–2238.

[28] S. Zhu, *Hybrid Methods for Solving Lagrangean Duals and Applications in Mixed Integer Programming*. Ph.D. thesis, University of Pennsylvania (1995).