AIMS Mathematics, 9(6): 14618–14639.
DOI: 10.3934/math.2024711
Received: 29 December 2023
Revised: 05 March 2024
Accepted: 08 March 2024
Published: 23 April 2024

http://www.aimspress.com/journal/Math

Research article

Simplicial decomposition of variational inequalities with multiple nonlinear column generation

William Chung*

Department of Management Sciences, City University of Hong Kong, Kowloon, Hong Kong

* **Correspondence:** Email: william.chung@cityu.edu.hk; Tel: +85234427057; Fax: +85234420198.

Abstract: Simplicial decomposition (SD) of variational inequalities experiences the long-tail convergence property. That is, the equilibrium solution rapidly progresses at first but then tails off, making only a tiny amount of progress per column generation iteration, which is a drawback of SD-VI. In the context of Dantzig-Wolfe of LP, it is reported that the more proposals are used to initialize the algorithm, the faster the solution can be found by reducing the number of decomposition steps. Therefore, I proposed to solve multiple nonlinear column generation (mNCG) subproblems in each SD-VI iteration (SD-VI-mNCG) instead of solving only one subproblem as in SD-VI. Generating multiple column generation subproblem solutions in each SD-VI iteration enabled the corresponding convex hull to be rapidly enlarged. Consequently, the number of SD-VI iterations could be greatly reduced. A transportation network equilibrium problem was used to study the performance of the SD-VI-mNCG.

Keywords: column generation; simplicial decomposition; convergence rate; variational inequalities; nonlinear programming
Mathematics Subject Classification: 90C33, 90C59

1. Introduction

Column generation (CG) is a method to solve large-scale mathematical programming problems such as optimization models and variational inequalities (VI) involving a vast number of variables. These models can be found in multi-cloud systems, multi-commodity economic equilibrium models, and transportation models. These models would become too large to be solved due to the rise of big

data. For example, large-scale VI problems can be easily found in stochastic VI settings with large-scale scenario analysis. Even a well-established VI solver (PATH) has been found to terminate when attempting to solve a large-scale VI problem due to a lack of memory [1]. Using column generation algorithms, such large-scale problems can be decomposed into smaller subproblems to be solved on several computers. Another motivation for using CG is to improve model development and maintenance efficiency by joining separately well-developed submodels when a converged solution is needed. Other research groups, such as Murphy et al. [2], have made similar observations. These smaller submodels can even reside on different computers connected by a network. On top of improving model development and maintenance, CG may improve computational efficiency. For example, Chung [3] reported that some computational instances of CG were faster than the reference method, the PATH solver. Chung [3] also reported that the PATH solver terminated when attempting to solve a large-scale VI problem because of insufficient memory.

However, CG has a major drawback which is the long-tail convergence property, called long-tail effect. CG involves solving a subproblem and a master problem iteratively. An example of CG is simplicial decomposition (SD) method. Simplicial decomposition was recently employed and modified for different application aspects, such as Bettiol et al. [4], Uciński [5], Guignard and Ahlatcioglu [6], and Delle Site [7]. For the multiple subproblems aspect, see Morabit et al. [8]. It is assumed that there are efficient solution methods to solve the subproblem and the master problem. However, it is well known that the SD converge rapidly at first but subsequently slowly, with a long tail of near-optimal solutions. As the final optimality gap cannot be closed, this long-tail convergence resulting in poor computational performance becomes a drawback. Other drawbacks are dual oscillations and primal degeneracy, and alternative dual optimal solutions in optimization models. In my experience, CG methods for VI problems inherit this long-tail convergence property. That is, CG-VI is likely to approach the equilibrium solution rapidly at first but then tail off, making only a tiny amount of progress per CG iteration. Consequently, the time savings afforded by the rapid initial convergence tend to be offset by the tailing off. I develop a new method to alleviate the long-tail effect of CG-VI in the current paper.

1.1. Literature review of the methods of resolving the tailing-off effect

1.1.1. Existing methods of resolving the tailing-off effect for LP

Implementation enhancement: Nazareth [9] used a set of LP problem examples consisting of a few constraints and variables to identify the numerical difficulties that can generally occur when applying the DW method. Nazareth showed that even when stable techniques, such as the stable refactorization of the corresponding basis matrix, are used, the columns of the computed master program can differ substantially from those of the true one. Later, Nazareth [10] noted that because changes in primal and dual variables are applied iteratively, a certain amount of "cleaning up" is to be expected. According to Ho [11] explanation of long-tail convergence, implicit error bounds may imply a tolerance for convergence below which further apparent improvements should be considered as noise. However, Lübbecke and Desrosiers [12] remarked that tailing off also occurs when columns are computed exactly, e.g., through the use of combinatorial algorithms.

Column dropping: One may anticipate that dropping columns from the master problem will improve computational efficiency. Dantzig [13] proposed some methods for deciding how long

proposals should be kept in the master problem, such as deleting all non-basic proposals immediately (apart from those just generated). Indeed, it is not entirely clear when to delete proposals from the master problem. Beale et al. [14] reported that when using Dantzig's [13] above-mentioned methods, for a problem with 450 constraints, the DW method performed marginally better than the simplex method. For NLP, Murphy [15] provided two conditions under which columns may be dropped from the restricted master problem. However, O'Neill's [16] computational results revealed virtually no differences in the measures of computational efficiency when all columns were retained and when only basic columns were retained.

Stabilized column generation methods: Another approach is to use stabilized column generation methods to ensure that the dual variable values of the linking constraints smoothly converge to their respective optima without vehement oscillation. From the literature, three stabilization techniques of optimization models can be found. These techniques are the proximity of a stability center, smoothing techniques, and centralized prizes for stabilizing the iterative dual solutions from the master problems. There are some well-known stabilization principles, such as the Boxtep method developed by Marsten et al. [17]. These stabilization methods are widely used in branch-and-bound algorithms for mixed integer programs. It should be noted that these techniques rely on solving dual master optimization problems. It would become a big challenge when one tries to use the dual master VI problems.

1.1.2. Existing methods of resolving the tailing-off effect for VI

To resolve the tailing-off problem afflicting simplicial decomposition of VI (SD-VI), Larsson and Patriksson [18] considered the use of subproblem column generations of nonlinear simplicial decomposition (NSD) to reduce the number of decomposition iterations. The generalized NSD method is obtained from the restricted simplicial decomposition method by replacing the linear column generation subproblem with a nonlinear column generation (NCG) subproblem. Generating columns based on the NCG subproblem requires fewer columns to describe an optimal solution, resulting in fewer decomposition iterations. There are three kinds of NSD column generation subproblems addressed by adopting Newton's method, the diagonalized Newton algorithm, and the projection method.

Chung et al. [19] extended the application of the DW procedure from linear programming (LP) to VI problems. The master problem and the subproblems of the decomposed VI are VIs. To shorten the computation time, Chung and Fuller [20] derived an approximation method to solve the subproblem VIs. Chung [3] integrated the results of Çelebi and Fuller [1] and Chung and Fuller [20] to develop an approximation method to solve DW-VI, in which both master VI and subproblem VI can be solved approximately as DW for LP. Chung [21] derived a truncated DW-VI method, in which the subproblem VI was approximately solved by one iteration of DW. Although they did not discuss the tailing-off effect, their computational results showed that the tailing-off effect for VI was alleviated.

In recent years, in the context of optimization problems, some research works concentrated on using machine learning to predict the stabilization technique for column generation and the patterns of different existing columns for re-optimization, see Morabit et al. [22] and Kraul et al. [23]. Although these approaches do not focus on column generation, their results can help to overcome the tailing-off effect in the re-optimization processes.

1.2. Approaches and contributions

In the context of Dantzig-Wolfe of LP, Dirickx and Jennergren [24] reported that, in general, the more proposals are used to initialize the algorithm, the faster the solution can be found. Therefore, in the current paper, I propose to solve multiple column generation subproblems in each SD-VI iteration instead of solving only one subproblem. By solving multiple column generation subproblems in each SD-VI iteration, I obtain two advantages, as follows: (1) Different columns will be generated and stored in the master problem to improve the representativeness of the corresponding convex hull. (2) Generating multiple column generation subproblem solutions in each SD-VI iteration will enable the corresponding convex hull to be rapidly enlarged. Consequently, the number of SD-VI iterations can be greatly reduced. The idea of generating multiple columns at a given iteration is not a new strategy in nonlinear programming, see Morabit et al. [8]. In short, I propose a new method, simplicial decomposition with multiple subproblems for VI, that solves multiple column generation subproblem in each SD-VI iteration to reduce the number of decomposition steps and thereby ease the tailing-off effect. One of the challenges of the new method is how to derive "multiple" subproblems. According to my knowledge, there is no research on solving multiple subproblems in SD-VI. It is a research gap in the literature. Moreover, the proposed method can also apply to NLP since the VI framework includes NLP.

While employing multiple subproblems (multiple column generation) in SD of optimization models, convergence properties and theories may be easily derived based on the existing results of the monotonic convergence properties. However, the convergence property of the SD-VI is non-monotone. Hence, the convergence properties and theories of multiple column generation of SD-VI is derived, and empirical tests are given.

In brief, the main contribution of the current paper is to derive a new method, simplicial decomposition with multiple subproblems for VI. The major challenges are (1) to derive "multiple subproblems" in SD-VI in the Subsection 3.2 and (2) to prove the convergence of the new method.

Section 2 provides the background of the VI problems, SD-VI methods, and SD of NLP with nonlinear column generation (NCG). Section 3 derives SD-VI with NCG (SD-VI-NCG) and with multiple NCGs (SD-VI-mNCG). SD-VI-NCG and SD-VI-mNCG are SD with one subproblem and multiple subproblems for VI, respectively. Convergence properties of the SD-VI-NCG and SD-VI-mNCG are also included. Section 4 is used to report computational performance of SD-NLP-mNCG, SD-VI-mNCG, and their re-optimization processes. Section 5 provides the conclusion with further research topics.

2. Background

The class of the VI problems I will study can be described as follows. $VI(G,K)$: Find a vector $x^* \in K \subseteq R^n$ such that $G(x^*)^T(x-x^*) \geq 0\ \forall x \in K$, where G is given continuous mapping from $K\ to\ R^n$, superscript T denotes the transpose, and all vectors are considered to be column vectors. K is a nonempty, closed, and convex set. Applications of $VI(G,K)$ can be found in user equilibrium traffic assignment problems and energy equilibrium problems. For large-scale VI, see Murphy et al. [2]. Harker and Pang [25] provided standard conditions for the existence and uniqueness of solutions to this class of VI problems. There are different methods of solving $VI(G,K)$, such as PIES-like methods or Newtonian methods.

2.1. Column generation of VI methods (CG-VI)

There are two popular column generation of VI methods, simplicial decomposition of VI (see Lawphongpanich and Hearn [26]) and Dantzig-Wolfe of VI (see Fuller and Chung [27]). Since the my results are derived from simplicial decomposition of VI (SD-VI), a brief description of SD-VI is given below. Moreover, multiple subproblems are generated from SD with nonlinear column generation (NCG) of nonlinear programming (Larsson et al. [28]), NCG is given in this subsection.

2.2. Simplicial decomposition of VI (SD-VI)

Lawphongpanich and Hearn [26] presented a type of simplicial decomposition of VI method for the user equilibrium traffic assignment problem, in which K is a closed convex set of feasible flow patterns, and G is a cost mapping. The following SD-VI is not exactly the one presented in Lawphongpanich and Hearn [26]. The major differences are that there is no pre-set convergence sequence (ε^k) for convergent of their SD-VI and no column dropping consideration for improving the computational performance of the master-VI problems. It is because my results of employing multiple subproblems do not require the pre-set convergence sequence and column dropping procedure.

Note that SD-VI consists of linear programming subproblem, $Sub-LP^k$, and an equilibrium master problem, $\text{Master}-\text{VI}^k$, where superscript k represents the iteration number of CG-VI. $Sub-LP^k$ is defined as follows:

$$\boldsymbol{Sub-LP^k\left(G\left(x_M^k\right),K\right)}\text{: find } x_S^k \in K, \text{ such that } G\left(x_M^k\right)^T\left(x-x_S^k\right)\geq 0\ \forall x\in K,$$

where x_M^k equals the solution of master problem ($\text{Master}-\text{VI}^k$), described below.

The feasible set for the master problem at iteration k is restricted to all convex combinations of the k proposals (solutions of the subproblem). Let X^k be the $n\times k$ matrix whose columns are the solutions x_S^i of the subproblems solved at iterations $i=1,\dots,k$; i.e., $X^k=\left[x_S^1,x_S^2,\dots,x_S^k\right]$. The weights on the proposals in the convex combination are contained in the vector $\lambda\in R^k$. The feasible set for the master problem is defined as

$$\Lambda^k=\{\lambda\in R^k|\ e^{kT}\lambda=1,\lambda\geq 0\},$$

where $e^k\in R^k$ is a vector with 1 for every entry. The master problem at iteration k is defined as

$$\boldsymbol{Master-VI^k\left(G,\Lambda^k\right)}\text{: find } \lambda^k\in\Lambda^k, \text{ such that } G(X^k\lambda^k)^T(X^k\lambda-X^k\lambda^k)\geq 0\ \forall\ \lambda\in\Lambda^k.$$

Alternatively, let $conv(X^k)$ represent the set of all convex combinations of the columns of X^k. Then, I have the following alternative master problem at iteration k, defined as

$$\text{Master}-\text{VI}^k(G,\text{K}^k)\text{: find } x_M^k\in K^k, \text{ such that } G\left(x_M^k\right)^T\left(x-x_M^k\right)\geq 0\ \forall\ x\in K^k=\{x\in conv(X^k)\}.$$

With the convergence gap, $CG^k=\ G\left(x_M^k\right)^T\left(x_S^k-x_M^k\right)$, for the stopping condition, I can define the standard SD-VI algorithm as follows:

[SD-VI]

Step 0: Set $k=0$. Choose $\varepsilon>0$, $x_M^1\in K$, and X^0 is a null matrix.

Step 1: Increment $k\leftarrow k+1$.

Solve $Sub-LP^k\left(G\left(x_M^k\right),K\right)$ and place the solution x_S^k in the matrix $X^k=\left[X^{k-1},x_S^k\right]$.

If $k = 1$ then go to Step 2; if $CG^k \geq -\varepsilon$ then STOP; else go to Step 2.

Step 2: Solve $\text{Master} - \text{VI}^k(G, K^k)$. Record $G\left(x_M^k\right)$. Go to Step 1.

In the next subsection, I describe the result of Larsson et al. [28] concerning how the nonlinear column generation (NCG) is employed in the context of SD of NLP. Based on their results, I first derive a SD of VI with NCG, then I derive a SD of VI with multiple NCG.

2.3. Simplical decomposition of NLP with nonlinear column generation (NCG)

According to Larsson et al. [28], SD of NLP with NCG (SD-NLP-NCG) is the combination of the SD principle and the nonlinear search direction finding subproblem of a primal descent algorithm for NLP, not VI.

$$\text{NLP: } Min_{x \in K} \int G(x)dx.$$

For NLP, the master problem and the subproblem of SD are shown below:

$$\boldsymbol{Master - NLP^k\left(G, K^k\right)}: \quad Min_{x \in conv\left(X^k\right)} \int G(x)dx.$$

$$\boldsymbol{Sub - LP^k\left(G\left(x_M^k\right), K\right)}: \quad Min_{x \in K}\, G\left(x_M^k\right)^T x.$$

For NCG, the above subproblem, $Sub - LP^k$, becomes the combination of the SD principle and the nonlinear search direction finding subproblem of a primal descent algorithm for NLP. Larsson et al. [28] modified the subproblem by adding a continuous regularization function, $\varphi(x, x): K \times K \mapsto R$, with the following properties:

(1) $\varphi(\cdot, y)$ is strictly convex and continuously differentiable on K for every $y \in K$;

(2) $\nabla_x \varphi(x, y) = 0$ holds if and only if $x = y$.

Then, given an iterate x_M^k, the subproblem of SD-NLP-NCG becomes an NLP.

$$\boldsymbol{Sub_{NCG} - NLP^k\left(G\left(x_M^k\right), K\right)}: Min_{x \in K}\, G\left(x_M^k\right)^T x + \varphi\left(x, x_M^k\right).$$

It is noted that the solution of $Sub_{NCG} - NLP^k$, x_S^k, is unique due to the compactness of K and the strict convexity of $\varphi(\cdot, y)$. Moreover, x_M^{k-1} is optimal in NLP if and only if x_M^{k-1} solves $Sub_{NCG} - NLP^k$, that is, $x_S^k = x_M^{k-1}$. In the numerical experiments of Larsson et al. [28], the function φ had the form:

$$\varphi(x, y) = \frac{1}{2}(x - y)^T Q(y)(x - y),$$

where $Q(y)$ is a positive definite and symmetric matrix for every y in K. Three matrixes were proposed: $Q(y) \equiv \gamma I, \gamma > 0$, $Q(y) = \nabla G\left(x_M^k\right)$, and $Q(y) = \text{diag}\, \nabla G\left(x_M^k\right)$.

3. Simplicial decomposition of VI and NCG

Based on the results of Larrsson et al. [28], I first define the method of SD-VI with a NCG (SD-VI-NCG), then SD-VI with multiple NCG (SD-VI-mNCG).

3.1. *Simplicial decomposition of VI with NCG (SD-VI-NCG)*

To derive a SD-VI with NCG, the $\boldsymbol{Sub - LP^k\left(G\left(x_M^k\right), K\right)}$ of SD-VI is replaced with $\boldsymbol{Sub_{NCG} - NLP^k\left(G\left(x_M^k\right), K\right)}$. Then, I can define the standard SD-VI-NCG algorithm as follows:

[SD-VI-NCG]

Step 0: Set $k = 0$. Choose $\varepsilon > 0$, $x_M^1 \in K$, and X^0 is a null matrix.

Step 1: Increment $k \leftarrow k + 1$.

Solve $\widetilde{Sub} - NLP^k\left(\tilde{G}\left(x; x_M^k\right), K\right)$ and place the solution x_S^k in the matrix $X^k = \left[X^{k-1}, x_S^k\right]$.

If $k = 1$ then go to Step 2; if $CG^k \geq -\varepsilon$ then STOP; else go to Step 2.

Step 2: Solve $\boldsymbol{Master - VI^k\left(G, K^k\right)}$. Record $G\left(x_M^k\right)$. Go to Step 1.

Noted that if I use $\tilde{G}\left(x; x_M^k\right) = G\left(x_M^k\right) + Q\left(x_M^k\right)\left(x - x_M^k\right)$, the subproblem is equal to the one in SD-NLP-NCG, $\boldsymbol{Sub_{NCG} - NLP^k\left(G\left(x_M^k\right), K\right)}$. That is,

$$Min_{x \in K}\, G\left(x_M^k\right)^T x + \frac{1}{2}\left(x - x_M^k\right)^T Q\left(x_M^k\right)\left(x - x_M^k\right).$$

3.1.1. *Convergence of SD-VI-NCG*

With the assumption of $\tilde{G}\left(x; x_M^k\right)$ is strictly monotone in x and $\tilde{G}\left(x_M^k; x_M^k\right) = G\left(x_M^k\right)$, I can extend the results of Chung and Fuller [20] to have the following properties.

Theorem 1a. λ^k solves $Master - VI^k(G, \Lambda^k)$ iff there exist $\lambda^k \in R_+^k$ and $\theta^k \in R$ such that the following relations hold:

$$X^{k^T} G(X^k \lambda^k) + e^k \theta^k = 0,$$

$$e^k \lambda^k - 1 = 0,$$

$$\lambda^{k^T}\left(X^{k^T} G(X^k \lambda^k) + e^k \theta^k\right) = 0.$$

Proof. This is a standard result for VI problems. See, e.g., Proposition 2.2 in Harker and Pang [25].

Theorem 2a. Given the property $\tilde{G}\left(x_M^k; \xi^k\right) = G\left(x_M^k\right)$. If x_M^k solves $\widetilde{Sub} - NLP^k\left(\tilde{G}\left(x; x_M^k\right), K\right)$, then x_M^k solves $VI(G, K)$.

Proof. Ref. new Theorem 4 in Chung and Fuller [20].

Suppose that $x_M^k = X^k \lambda^k$ solves $\widetilde{Sub} - NLP^k\left(\tilde{G}\left(x; x_M^k\right), K\right)$. It follows that $\tilde{G}\left(x_M^k; x_M^k\right)^T\left(x - x_M^k\right) \geq 0\ \forall x \in K$. As $\tilde{G}\left(x_M^k; x_M^k\right) = G\left(x_M^k\right)$, I have $G\left(x_M^k\right)^T\left(x - x_M^k\right) \geq 0\ \forall x \in K$ and I may conclude that x_M^k solves $VI(G, K)$.

Theorem 3a. Assume that $\tilde{G}(x; \xi^k)$ is strictly monotone in x. Given the property $\tilde{G}\left(x_M^k; \xi^k\right) = G\left(x_M^k\right)$. If $CG^k = G\left(x_M^k\right)^T\left(x_S^k - x_M^k\right) \geq 0$, then x_M^{k-1} solves $VI(G, K)$.

Proof. Ref. new Theorem 6(a) of Chung and Fuller [20].

We shall show that if x_M^k does not solve $VI(G, K)$, then $CG^k < 0$. By Theorem 2a, $x_M^k \neq x_S^k$, and since $\tilde{G}\left(x_M^k; \xi^k\right) = G\left(x_M^k\right)$, strict monotonicity of $\tilde{G}(x; \xi^k)$ implies that $\left(G\left(x_M^k\right) - \tilde{G}\left(x_S^k; \xi^k\right)\right)^T\left(x_M^k - x_S^k\right) > 0$. Since x_S^k solves $\widetilde{Sub} - NLP^k\left(\tilde{G}\left(x; x_M^k\right), K\right)$, it follows that

$\tilde{G}(x_S^k;\xi^k)^T(x_M^k - x_S^k) \geq 0$. Adding this last inequality to the strict inequality and multiply by -1 yields $G(x_M^k)^T(x_S^k - x_M^k) < 0$.

Theorem 4. Either $CG^k \geq 0$ at a finite iteration number k, or $CG^k < 0$ for all iterations k. In the latter case, any infinite subsequence of $\{(x_M^k, x_S^k)\}_{k=1}^{\infty}$ has at least one limit point, and if G is continuous then $lim_{k\to\infty} CG^k = 0$.

Proof. Ref. new Theorem 8 of Chung and Fuller [20].

Suppose that $CG^k = G(x_M^k)^T(x_S^k - x_M^k) < 0$ for all k and suppose, contrary to my desired conclusion, that there exists an $\varepsilon > 0$ and infinite set of iteration numbers, $\mathcal{T}$, such that $G(x_M^k)^T(x_S^k - x_M^k) < -\varepsilon$ for all $k \in \mathcal{T}$. For any k and r with $r > k$, x_S^k is one of the proposals available to $Master - VI^r(G, \Lambda^r)$; thus, I may use the complementarity conditions in Theorem 1 to derive an inequality. I do this by examining the dual feasibility constraint in $Master - VI^r(G, \Lambda^r)$ associated with the primal variable λ_k^r which is the weight associate with the proposal x_S^k: ${x_S^k}^T G(x_M^r) + \theta^r \geq 0$. I may eliminate the variable θ^r using the complementarity slackness condition $\sum_{i=1}^r \left({x_S^i}^T G(x_M^r) + \theta^r\right)^T \lambda_i^r = 0$, and using the constraint $\sum_{i=1}^r \lambda_i^r = 1$, and the fact that $x_M^r = \sum_{i=1}^r \lambda_i^r x_S^i$: $\theta^r = -{x_M^r}^T G(x_M^r)$. This allow us to rewrite the constraint associated with λ_k^r: $G(x_M^r)^T(x_S^{k+1} - x_M^r) \geq 0$, for all k and r with $r > k$. Subtracting this from the strict inequality derived earlier, yields $G(x_M^k)^T(x_S^k - x_M^k) - G(x_M^r)^T(x_S^k - x_M^r) < -\varepsilon$, for all $k, r \in \mathcal{T}$ with $r > k$. Note that continuity of G makes the left of the inequality continuous in (x_M, x_S). By the property that any infinite subsequence $\{(x_M^k, x_S^k)\}_{k=1}^{\infty}$ has at least one limit point, there exists a subset $\hat{\mathcal{T}} \subset \mathcal{T}$ such that $\lim_{k\to\infty, k\in\hat{\mathcal{T}}}(x_M^k, x_S^k) = (\hat{x}_M, \hat{x}_S)$, a limit point. Finally, I let $r \to \infty$ through values $r \in \hat{\mathcal{T}}$, in the inequality, and then let $k \to \infty$ through values $k \in \hat{\mathcal{T}}$ (this order of limits ensures that $r > k$ throughout the limiting process) to derive the contradiction $0 = G(\hat{x}_M)^T(\hat{x}_S - \hat{x}_M) - G(\hat{x}_M)^T(\hat{x}_S - \hat{x}_M) < -\varepsilon < 0$. That is, $CG^k < 0$, contradicting the assumption that $CG^k \geq 0$.

After deriving a SD-VI-NCG, I derive SD-VI with multiple NCG (SD-VI-mNCG) in the next subsection.

3.2. Simplicial decomposition of VI with multiple NCG (SD-VI-mNCG)

For having multiple NCGs, I introduce a parameter, α_j, to the additional component of SD-VI subproblem approximation. Then, I define the j^{th} subproblem approximation mapping:

$$\tilde{G}_j(x; x_M^k; \alpha_j) = G(x_M^k) + \alpha_j Q(x_M^k)(x - x_M^k).$$

The corresponding subproblem of SD-VI-mNCG:

$$\widetilde{\boldsymbol{Sub}} - \boldsymbol{NLP_j^k}\left(\tilde{\boldsymbol{G}}_{\boldsymbol{j}}(\boldsymbol{x}; \boldsymbol{x_M^k}; \boldsymbol{\alpha_j}), \boldsymbol{K}\right): Min_{x\in K}\, G(x_M^k)^T x + \frac{\alpha_j}{2}(x - x_M^k)^T Q(x_M^k)(x - x_M^k).$$

Assuming that I have J subproblems, then, for each SD iteration, I have J subproblem proposals, $(x_{S1}^k, \dots, x_{Sj}^k, \dots, x_{SJ}^k)$. For the j^{th} subproblem, the proposal, x_{Sj}^k, is stored in the matrix $X_{Sj}^k = [X_{Sj}^{k-1}, x_{Sj}^k] = [x_{Sj}^1, x_{Sj}^2, \dots, x_{Sj}^k]$. Then, let $X_{mNCG}^k = [X_{S1}^k, \dots, X_{Sj}^k, \dots, X_{SJ}^k]$.

The feasible set for the master problem at iteration k is restricted to all convex combinations of the k proposals (solutions of all subproblems). Let X_{mNCG}^k be the $n \times (k \times j)$ matrix whose columns

are solutions x_{Sj}^{i} of all subproblems solved at iterations $i = 1, \dots, k$. The weights on the proposals in the convex combination are contained in the vector $\lambda_j^k \in R^{k\times j}$. The feasible set for the master problem is defined as $\Lambda_{mNCG}^k = \left\{\lambda_j^k \in R^{k\times j} \middle| \left(e^{k\times j}\right)^T \lambda_j^k = 1, \lambda_j^k \geq 0\right\}$, where $e^{k\times j} \in R^{k\times j}$ is a vector with 1 for every entry. The master problem at iteration k is defined as

$$\boldsymbol{Master - VI_{mNCG}^{k}}\left(\boldsymbol{G}, \boldsymbol{\Lambda_{mNCG}^{k}}\right)\text{: Find } \lambda_j^k \in \Lambda_{mNCG}^k \text{, such that } G\left(X_{mNCG}^k \lambda_j^k\right)^T \left(X_{mNCG}^k \lambda - X_{mNCG}^k \lambda_j^k\right) \geq 0 \ \forall \ \lambda_j^k \in \Lambda_{mNCG}^k.$$

Alternatively, let $conv(X^k)$ represent the set of all convex combinations of the columns of X^k. Then, I have the following alternative master problem at iteration *k*, defined as

$$\boldsymbol{Master - VI_{mNCG}^{k}}\left(\boldsymbol{G}, \boldsymbol{K_{mNCG}^{k}}\right)\text{: Find } x_M^k \in K_{mNCG}^k \text{, such that } G\left(x_M^k\right)^T \left(x - x_M^k\right) \geq 0 \ \forall \ x \in K_{mNCG}^k = \left\{x \in conv\left(X_{mNCG}^k\right)\right\}.$$

Since I have J subproblems, I can have J convergence gaps for the stopping condition, according to Theorems 3 and 4, $CG_j^k = G\left(x_M^k\right)^T \left(x_{Sj}^k - x_M^k\right)$ for all j.

We first state the algorithm of SD-VI-mNCG and then the corresponding new theorems about the convergence and the stopping conditions.

[SD-VI-mNCG]

Step 0: Set $k = 0$. Choose $\varepsilon > 0$, $\alpha_j > \alpha_{j-1} > \cdots > \alpha_1 > 0, x_M^1 \in K$, and X_i^0 is a null matrix i=1,… , j.

Step 1: Increment $k \leftarrow k + 1$.

Solve $\widetilde{Sub} - NLP_j^k\left(\tilde{G}_j\left(x; x_M^k; \alpha_j\right), K\right)$ for all j, and place the solution x_{Si}^k in the matrix $\left[X_{Si}^{k-1}, x_{Si}^k\right], i = 1, \dots, j$; and update $X_{mNCG}^k = \left[X_{S1}^k, \dots, X_{Sj}^k\right]$. If $k = 1$ then go to Step 2; else if $CG_j^k = G\left(x_M^k\right)^T \left(x_{Sj}^k - x_M^k\right) \geq -\varepsilon$ then STOP; else go to Step 2.

Step 2: Solve Master $- \text{VI}_{mNCG}^k\left(G, K_{mNCG}^k\right)$. Record $G\left(x_M^k\right)$. Go to Step 1.

3.2.1. The computational sequence of SD-VI-mNCG

The SD-VI-mNCG algorithm first solves $\widetilde{Sub} - NLP_1^1$ with $\alpha_1 = 0$, to generate a column proposal $\left(x_{Sj}^1\right)$ for all j to be placed in the matrix X_{mNCG}^1 of $Master - VI_{mNCG}^1$ to enlarge the set Λ_{mNCG}^1. $Master - VI_{mNCG}^1$ is solved to obtain a new x_M^1 for new $G(x_M^1)$ and $Q(x_M^1)$. The algorithm proceeds with $k = k + 1$, and the new column proposals from $\widetilde{Sub} - NLP_j^k$ is added to the matrix X_{Sj}^k for all j and update X_{mNCG}^k which $Master - VI_{mNCG}^k$ uses to define its convex combinations, and the next solution of the $Master - VI^k$ produces a new $G\left(x_M^k\right)$ and $Q\left(x_M^k\right)$ for $\widetilde{Sub} - NLP_j^k$. The algorithm proceeds in this manner until the stopping criterion is satisfied (see Theorem 5 in the following subsection).

3.2.2. Convergence of SD-VI-mNCG

Theorem 1b. λ_j^k solves Master $- \text{VI}_{mNCG}^k\left(G, \Lambda_{mNCG}^k\right)$ iff there exist $\lambda_j^k \in R_+^{k\times j}$ and $\theta_j^k \in R$ such that all of the following relations hold:

$$X_{mNCG}^{k\ T} G\left(X_{mNCG}^k \lambda_j^k\right) + e^{k\times j}\theta_j^k = 0,$$

$$e^{k\times j}\lambda_j^k - 1 = 0,$$

$$\lambda_j^{kT}\left({X_{mNCG}^k}^T G\left(X_{mNCG}^k\lambda_j^k\right) + e^{k\times j}\theta_j^k\right) = 0.$$

Theorem 2b. Given the property $\tilde{G}_j\left(x_M^k; x_M^k; \alpha_j\right) = G\left(x_M^k\right)$. If x_M^k solves any one of J subproblem: $\widetilde{Sub} - NLP_j^k\left(\tilde{G}_j\left(x; x_M^k; \alpha_j\right), K\right)$, then x_M^k solves $VI(G, K)$.

Proof. Ref. new Theorem 4 in Chung and Fuller [20].

Suppose that x_M^k solves the j^{th} subproblem, $\widetilde{Sub} - NLP_j^k$. It follows that $\tilde{G}\left(x_M^k; x_M^k; \alpha_j\right)^T\left(x - x_M^k\right) \geq 0\ \forall x \in K$. As $\tilde{G}_j\left(x_M^k, x_M^k; \alpha_j\right) = G\left(x_M^k\right)$, I have $G\left(x_M^k\right)^T\left(x - x_M^k\right) \geq 0\ \forall x \in K$ and I may conclude that x_M^k solves $VI(G, K)$.

Theorem 3b. Assume that $\tilde{G}_j\left(x; x_M^k; \alpha_j\right)$ is strictly monotone in x. Given the property $\tilde{G}_j\left(x_M^k; x_M^k; \alpha_j\right) = G\left(x_M^k\right)$. If any $CG_j^k = G\left(x_M^k\right)^T\left(x_{Sj}^k - x_M^k\right) \geq 0$, then x_M^k solves $VI(G, K)$.

Proof. Ref. new Theorem 6(a) of Chung and Fuller [20].

We shall show that if x_M^k does not solve $VI(G, K)$, then $CG^k < 0$. By Theorem 2b, $x_M^k \neq x_{Sj}^k$, and since $\tilde{G}_j\left(x_M^k; x_M^k; \alpha_j\right) = G\left(x_M^k\right)$, strict monotonicity of $\tilde{G}_j\left(x_M^k; x_M^k; \alpha_j\right)$ implies that $\left(G\left(x_M^k\right) - \tilde{G}_j\left(x_M^k; x_M^k; \alpha_j\right)\right)^T\left(x_M^k - x_{Sj}^{k+1}\right) > 0$. Since x_{Sj}^k solves $\widetilde{Sub} - NLP_j^k\left(\tilde{G}_j\left(x; x_M^k; \alpha_j\right), K\right)$, it follows that $\tilde{G}_j\left(x_{Sj}^k; x_M^k; \alpha_j\right)^T\left(x_M^k - x_{Sj}^k\right) \geq 0$. Adding this last inequality to the strict inequality and multiplying by -1 yields $G\left(x_M^k\right)^T\left(x_{Sj}^k - x_M^k\right) = CG_j^k < 0$.

Theorem 5. Let $\alpha_j > \alpha_{j-1} > \cdots > \alpha_1 > 0$ and let z_j^k be the optimal value of objective function of the subproblem $\widetilde{Sub} - NLP_j^k$. Then $z_j^k \geq z_{j-1}^k \geq \cdots \geq z_1^k$, and $CG_j^k \geq CG_{j-1}^k \geq \cdots \geq CG_1^k$, where $CG_j^k = G\left(x_M^k\right)^T\left(x_{Sj}^k - x_M^k\right)$.

Proof. For any subproblem i, let $x_{S\,i}^k$ solve $\widetilde{Sub} - NLP_i^k\left(\tilde{G}_j\left(x; x_M^k; \alpha_i\right), K\right)$, and let $z_i^k\left(x_{Si}^k\right)$ be the corresponding objective value. That is, $z_i^k = G\left(x_M^k\right)^T x_{S\,i}^k + \frac{\alpha_i}{2}\left(x_{S\,i}^k - x_M^k\right)^T Q\left(x_M^k\right)\left(x_{S\,i}^k - x_M^k\right)$. As $x_{S\,i+1}^k$ is another feasible solution, it follows that $G\left(x_M^k\right)^T x_{S\,i+1}^k + \frac{\alpha_i}{2}\left(x_{S\,i+1}^k - x_M^k\right)^T Q\left(x_M^k\right)\left(x_{S\,i+1}^k - x_M^k\right) \geq G\left(x_M^k\right)^T x_{S\,i}^k + \frac{\alpha_i}{2}\left(x_{S\,i}^k - x_M^k\right)^T Q\left(x_M^k\right)\left(x_{S\,i}^k - x_M^k\right)$. Since $\alpha_{i+1} > \alpha_i$ for all j, I have $G\left(x_M^k\right)^T x_{S\,i+1}^k + \frac{\alpha_{i+1}}{2}\left(x_{S\,i+1}^k - x_M^k\right)^T Q\left(x_M^k\right)\left(x_{S\,i+1}^k - x_M^k\right) \geq G\left(x_M^k\right)^T x_{S\,i}^k + \frac{\alpha_i}{2}\left(x_{S\,i}^k - x_M^k\right)^T Q\left(x_M^k\right)\left(x_{S\,i}^k - x_M^k\right)$. That is $z_{i+1}^k \geq z_i^k$. Adding $\left(-G\left(x_M^k\right)^T\left(x_M^k\right)\right)$ in both sides, I have $G\left(x_M^k\right)^T\left(x_{S\,i+1}^k - x_M^k\right) + \frac{\alpha_{i+1}}{2}\left(x_{S\,i+1}^k - x_M^k\right)^T Q\left(x_M^k\right)\left(x_{S\,i+1}^k - x_M^k\right) \geq G\left(x_M^k\right)^T\left(x_{S\,i}^k - x_M^k\right) + \frac{\alpha_i}{2}\left(x_{S\,i}^k - x_M^k\right)^T Q\left(x_M^k\right)\left(x_{S\,i}^k - x_M^k\right)$. It implies $CG_{i+1}^k + \frac{\alpha_{i+1}}{2}\left(x_{S\,i+1}^k - x_M^k\right)^T Q\left(x_M^k\right)\left(x_{S\,i+1}^k - x_M^k\right) \geq CG_i^k + \frac{\alpha_i}{2}\left(x_{S\,i}^k - x_M^k\right)^T Q\left(x_M^k\right)\left(x_{S\,i}^k - x_M^k\right)$. When x_M^k solve $\widetilde{Sub} - NLP_{i+1}^k$, $x_M^k = x_{S\,i+1}^k$ and $\frac{\alpha_{i+1}}{2}\left(x_{S\,i+1}^k - x_M^k\right)^T Q\left(x_M^k\right)\left(x_{S\,i+1}^k - x_M^k\right) = 0$. Then, $CG_{i+1}^k \geq CG_i^k + \frac{\alpha_i}{2}\left(x_{S\,i}^k - x_M^k\right)^T Q\left(x_M^k\right)\left(x_{S\,i}^k - x_M^k\right) \geq CG_i^k$.

With Theorem 4, the stopping condition can be defined by $CG_j^k = G\left(x_M^k\right)^T\left(x_{Sj}^k - x_M^k\right)$. In addition, it should be noted that the SD-VI-mNCG can be easily reduced to SD-NLP-mNCG for nonlinear programming by replacing the $\text{Master} - \text{VI}_{mNCG}^k\left(G, K_{mNCG}^k\right)$ with $\text{Master} - \text{NLP}^k\left(G, K_{mNCG}^k\right)$ in the Subsection 2.3.

4. Empirical results of SD with multiple NCG

In my implementation, the test models, the column generation algorithms, and the reference algorithm without column generation are coded into GAMS programs, executed on a PC with Intel Processor (12th Gen Core i7-12700, 2100 Mhz, 12 Core(s), 20 Logical Processors and 16 GB RAM. The reference algorithm is a relaxation algorithm [29] by which the test models are solved so that I can have reference results for evaluating the accuracy and speed of the column generation algorithms. The master problem is also solved by the same relaxation algorithm. Each iteration of the relaxation algorithm is a nonlinear programming (NLP) calculation. For all models solved by SD-VI method, in the first subproblem at step k = 1, the initial values of $G(x_M^1)$ and α_j are arbitrarily set to 10 and 0, respectively, which produces first proposals that ignore the nonlinear column generation component. In the relaxation method for the reference calculations, the first NLP also has the initial values of all x_M are set to 10.

For all column generation calculations, I set the convergence tolerance $\varepsilon = 0.000001$, except where noted. The relaxation iterations of the restricted master problems and the reference method, and all subproblems, except subproblems with $\alpha_j = 0$, are solved by CONOPT 3 called from GAMS. The subproblems with $\alpha_j = 0$ are linear programs that are solved by CPLEX. It should be noted that the SD-VI-mNCG with $\alpha_j = 0$ reduces to the SD-VI method of [26]. The SD-VI-mNCG with one NCG and $\alpha_j = 0.5$ reduces to the SD-VI-NCG method of [28].

In the calculations, many similar optimization problems are solved repeatedly: The subproblem, a sequence of optimization problems related to the restricted master problem, and another related to the reference algorithm; except for the first time, each time an optimization calculation is done, it is started from the last solution in order to reduce computation time. This process is not used in the multiple column generation method since I assume that each nonlinear column generation subproblem is implemented in a server individually and all these subproblems are solved in parallel.

To summarize, even if a column generation algorithm takes more time than no column generation algorithm (the relaxation algorithm), there can be an advantage to a column generation approach, in model development. However, if column generation takes a very large amount of time, then such an advantage might not be worthwhile. Therefore, I am interested in tests that measure the time of column generation, compared with no column generation, and in variants of multiple column generation algorithms that may run in shorter times.

4.1. A simple example (NEW)

An example is Example 2 of Nagurney and Dhanda [30], a VI problem of a multiproduct multipollutant oligopolistic market model (**MMOM-VI**) with ambient-based pollution permits and transaction costs. In this model, m firms or sources of pollution, r pollutants emitted by the firms, and n receptor points exist. I also let e_i^t be the amount of pollutant t emitted by firm i, l_{ij}^t be the number of licenses for pollutant t at receptor point j held by firm i, l_{ij}^{t0} be the initial allocation of licenses made by a regulatory agency and p_j^t be the price of the licenses for pollutant t that affects receptor point j.

The cost of purchasing licenses for specific pollutant t that affects receptor point j for source i is given by $\sum_{j=1}^{n} p_j^{t*}\left(l_{ij}^t - l_{ij}^{t0}\right)$, where p_j^{t*} is the market clearing price determined by the VI.

Each firm i in the oligopoly is faced with cost f_i for producing the vector of quantities q_i, where

$$f_i = f_i(q_i) = \sum_{d=1}^{s}\left[c_{id}q_{id} + \frac{\beta_{id}}{\beta_{id}+1}K_{id}^{-1/\beta_{id}}q_{id}^{(\beta_{id}+1)/\beta_{id}}\right],$$

and q_{id}= the quantity of product d produced by firm i.

Each firm i in the region is also faced with joint-cost g_i, where $g_i = g_i(e_i, q_i) = \sum_{t=1}^{r}[g1_{it}(e_i^t)^2 + g2_{it}e_i^t + g4_{it}] + \sum_{d=1}^{s} g3_{it}\, q_{id}$.

The transaction cost function employed by firm i for pollutant t at receptor point j was of the form $c_{ij}^t = c_{ij}^t(l_{ij}^t) = \emptyset 1_{ijt}(l_{ij}^t)^2 + \emptyset 2_{ijt} l_{ij}^t + \alpha_{ijt}$.

The firms are oligopolistic in their product markets, and they affect the prices of the outputs. The price of product d is denoted by $\rho_d = \rho_d(\sum_{i=1}^{m} q_{id})$.

Then, I can have the **MMOM-VI** model shown below.

MMOM-VI: Find $(q_{id}^*, e_i^{t*}, l_{ij}^{t*}) \in K_{q,e,l}$ such that

$$\sum_{i=1}^{m}\sum_{d=1}^{s}\left[\frac{\partial f_i(q_i^*)}{\partial q_{id}} + \frac{\partial g_i(e_i^*, q_i^*)}{\partial q_{id}} - \frac{\partial \rho_d(\sum_{i=1}^{m} q_{id}^*)}{\partial q_{id}} q_{id}^* - \rho_d\left(\sum_{i=1}^{m} q_{id}^*\right)\right] \times [q_{id} - q_{id}^*]$$

$$+ \sum_{i=1}^{m}\sum_{t=1}^{r} \frac{\partial g_i(e_i^*, q_i^*)}{\partial e_i^t} \times [e_i^t - e_i^{t*}] + \sum_{i=1}^{m}\sum_{t=1}^{r}\sum_{j=1}^{n} \frac{c_{ij}^t(l_{ij}^{t*})}{\partial l_{ij}^t} \times \left[l_{ij}^t - l_{ij}^{t*}\right] \geq 0,$$

$$\forall (q_{id}, e_i^t, l_{ij}^t) \in K_{q,e,l} = \{(q_{id}, e_i^t, l_{ij}^t)|,$$

$$l_{ij}^t - h_{ij}^t e_i^t \geq 0 \qquad \forall j, t,$$

$$\sum_{i=1}^{m} l0_{ij}^t - \sum_{i=1}^{m} l_{ij}^t \geq 0 \qquad (p_j^t) \qquad \forall j, t,$$

$$(q_{id}, e_i^t, l_{ij}^t) \geq 0 \quad \forall d, j, t\},$$

where $\sum_{i=1}^{m} l0_{ij}^t$ is the environmental quality standard, and h_{ij}^t denotes the contribution of one unit of emission by source i to the average pollutant concentration of type t at receptor point j. All the parameters and the reference results can be found in the online appendix of [30]. For convenience, they are given in the Appendix of the current paper.

Similarly, I adopt $Q(x_M) = \text{diag}\, \nabla G(x_M)$. That is, let $x_M = (q_{idM}, e_{iM}^t, l_{ijM}^t)$, and I have

$$Q(q_{idM}, e_{iM}^t, l_{ijM}^t) =$$

$$\begin{pmatrix} \sum_{i=1}^{m}\sum_{d=1}^{s}\left[\frac{1}{\beta_{id}} K_{id}^{-\frac{1}{\beta_{id}}} q_{idM}^{\left(\frac{1}{\beta_{id}}-1\right)} - \rho_d{}^{\frac{1}{1.1}} q_{idM}\left(\frac{2.1}{(1.1)^2}\right)\left(\sum_{i=1}^{m} q_{idM}\right)^{\frac{-3.2}{1.1}} + \frac{2}{1.1}\rho_d{}^{\frac{1}{1.1}}\left(\sum_{i=1}^{m} q_{idM}\right)^{\frac{-2.1}{1.1}}\right], \\ \sum_{i=1}^{m}\sum_{t=1}^{r} 2 * g1_{it}, \\ \sum_{i=1}^{m}\sum_{t=1}^{r}\sum_{j=1}^{n} 2 * \emptyset 1_{ijt}, \end{pmatrix}$$

which is added to the subproblem and the $\boldsymbol{K} = K_{q,e,l}$.

Table 1 summarises the performance of SD-VI-NCG(x) and SD-VI-mNCG of **MMOM-VI**, in which the first column provides the names of the models. Ref_method is the reference method, which is the relaxation algorithm. SD-VI-NCG($\alpha/2$) is the method of simplicial decomposition with one NCG of $\alpha/2$. For instance, SD-VI-NCG(0.5) is the SD-VI-NCG with $\alpha/2 = 0.5$. The last two rows of Table 1 present the performance of SD-VI-mNCG. I use m=3, 5 to represent how many NCG

subproblems are used. For example, SD-VI-3NCG(0.1, 0.3, 0.5) consists of three subproblems with $\alpha/2$ = 0.1, 0.3, and 0.5. It is interesting that all two SD-VI-mNCG methods use the smallest number of decomposition steps, 7. However, there is no conclusive result in terms of computational performance. The last column of Table 1 shows the stopping condition mentioned in Theorem 5 that the SD-VI was terminated by the subproblem with the largest value of $\alpha/2$.

Table 1. Performance of SD-VI-NCG(x) and SD-VI-mNCG of **MMOM-VI**.

Methods	Dstep	Time(M)	Time(S)*	Time(T)	Stopped by NCG
Ref_method				0.222	
SD-VI [26]	16	1.484	0.032	1.516	n.a.
SD-VI-NCG(0.1)	10	0.808	0.125	0.933	n.a.
SD-VI-NCG(0.3)	8	0.482	0.063	0.545	n.a.
SD-VI-NCG(0.5) [28]	10	0.424	0.110	0.534	n.a.
SD-VI-3NCG(0.1,0.3,0.5)	7	0.499	0.095	0.594	NCG(0.5)
SD-VI-5NCG(0.1,…,0.5)	7	0.531	0.096	0.627	NCG(0.5)

* It is considered that all subproblems are solved in parallel, and the greatest iterative solution time of a subproblem is used to calculate the total solution time of subproblems. Dstep = decomposition step; Time(M) = solution time of Master-VI in second; Time(S) = solution time of subproblem in second; Time(T) = solution time of the method in second; Stopped by NCG = the SD-VI terminated by the subproblem with the corresponding value of $\alpha/2$.

4.2. The test model (was 4.1)

Since the SD method is usually used for solving transportation network equilibrium problems, I use this kind of problems to develop test model. A large-scale real transportation network equilibrium problem is used[1]. The network of this problem represents the extra-urban area of the city of Arezzo (Italy). It consists of 213 nodes, 598 arcs and 2423 O/D pairs. The form of the link (i,j) cost functions is asymmetric:

$$G(x_{ij}) = a * f_{ij} + b * f_{ij} * \left(\frac{x_{ij}+d*x_{ji}}{c_{ij}}\right)^p,$$

where

$G(x_{ij})$ = Link travel time (i,j), from node i to j,

f_{ij} = free flow time of the link (i,j),

c_{ij} = capacity of the link (i,j),

$d = asymmetric\ factor = 0.5$,

a, b, p = constant.

The objective function of the subproblem, $\widetilde{Sub} - NLP_j^k\left(\tilde{G}_j\left(x; x_M^{k-1}; \alpha_j\right), K\right)$, is

$$G\left(x_M^k\right)^T x + \frac{\alpha_j}{2}\left(x - x_M^k\right)^T Q\left(x_M^k\right)\left(x - x_M^k\right),$$

[1] https://pages.di.unipi.it/passacantando/test_networks.html.

where I adopt $Q(x_M^k) = \text{diag}\,\nabla G(x_M^k)$. That is,

$$Q(x_M^k) = \frac{b*f_{ij}*p}{c} * \left(\frac{x_{M,ij}^k + d*x_{M,ji}^k}{c}\right)^{p-1},$$

with different α_j, I can have different nonlinear column generation subproblems.

We first provide the computational performance results of SD-NLP with multiple NCG, and then the results of SD-VI with multiple NCG. For having SD-NLP test model, I set the asymmetric factor d=0.

4.3. Performance of SD with multiple NCG for NLP (was 4.2)

In this subsection, I report the solution time of simplicial decomposition with one NCG subproblem of NLP. Table 2 summarises the computational performance of SD-NLP-NCG, in which the first column provides the names of the models. SD-NLP-NCG($\alpha/2$) is the method of simplicial decomposition with one NCG of $\alpha/2$. For instance, SD-NLP-NCG(0.5) is the SD-NLP-NCG with $\alpha/2 = 0.5$. Hence, SD-NLP-NCG(0) is the simple SD-NLP with LP subproblem, and SD-NLP-NSD(0.5) is the one in Larsson et al. [28]. From Table 2, the empirical result shows that the decomposition steps taken by SD-NLP is 87, which is expected and much higher than any SD-NLP-NCG methods with $\alpha/2 = 0.1, 0.2, 0.3, 0.4, 0.5,$ and 1.0. The range of the decomposition steps is 8 to 22. The SD-NLP-NCG with $\alpha/2$=0.4 provides the best performance in terms of the number of decomposition steps, 8. However, there is no clear relationship between the number of decomposition steps and the total solution time while changing the value of α. On the other hand, the fastest method is SD-NLP-NCG(0) because its subproblem is an LP solved by CPLEX, which is one of the fastest LP solvers. It is noted that SD-NLP-NCG(0) is a simple SD-NLP, and it takes more time to solve its master NLP problem comparing with other methods with different values of α. It implies that SD-NLP-NCG(0) may not be the fastest if a larger model incurs a larger number of generated columns in the master problem.

Table 2. Computational performance of SD-NLP-NCG and SD-NLP-mNCG.

Method	Dstep	Time(M)	Time(S)	Time(T)
Ref_method				84.797
SD-NLP [26]	87	2.129	17.359	19.488
SD-NLP-NCG(0.1)	18	0.454	107.488	107.942
SD-NLP-NCG(0.2)	11	0.154	203.720	203.874
SD-NLP-NCG(0.3)	10	0.190	267.908	268.098
SD-NLP-NCG(0.4)	8	0.140	198.733	198.873
SD-NLP-NCG(0.5) [28]	9	0.172	222.281	222.453
SD-NLP-NCG(1.0)	22	0.356	276.747	277.103
SD-NLP-3NCG(0.1,0.3,0.5)	6	0.125	182.187	182.312
SD-NLP-5NCG(0.1,…,0.5)	6	0.467	229.782	230.249
SD-NLP-10NCG(0.1,…,1.0)	6	0.156	891.548	891.704

* Dstep = decomposition step; Time(M) = solution time of Master-VI in second; Time(S) = solution time of subproblem in second; Time(T) = solution time of the method in second.

The last three rows of Table 2 present the performance of SD-NLP-mNCG. I use m=3, 5, and 10 to represent how many NCG subproblems are used. For example, SD-NLP-3NCG(0.1, 0.3, 0.5)

consists of three subproblems with $\alpha/2$ = 0.1, 0.3, and 0.5. All three SD-NLP-mNCG methods use fewer decomposition steps, as expected, than SD-NLP-NCG. It is interesting that all three SD-NLP-mNCG methods use 6 decomposition steps, which may imply that computational performance of SD-NLP-mNCG does not rely on the number of subproblems.

4.4. Performance of SD with multiple NCG for VI (was 4.3)

In this subsection, I report the solution time of simplicial decomposition of VI with one NCG subproblem, which is similar to the one in Larsson et al. [28]. Table 3 summarizes the computational performance of SD-VI-NCG(x). This empirical result shows that the SD-VI-NSD with $\alpha/2$=0.3 provides the best performance in terms of the number of decomposition steps, 13. It is expected that the number of decomposition steps of SD-VI-NSD($\alpha/2$) is much smaller than that of SD-VI, 152, as discussed in Larsson et al. [28]. Like the SD-NLP-NCG results, there is no clear relationship between the number of decomposition steps and the total solution time while changing the value of α. On the other hand, the fastest method is SD-VI because its subproblem is an LP solved by CPLEX. It is noted that SD-VI takes more time to solve its master VI problem. It implies that SD-VI may not be the fastest if a larger model incurs a larger number of generated columns in the master problem.

Table 3. Computational performance of SD-VI-NCG(x).

Method	Dstep	Time(M)	Time(S)	Time(T)
Ref_method				117.517
SD-VI [26]	152	46.419	30.422	76.841
SD-VI-NCG(0.1)	23	5.307	259.453	264.760
SD-VI-NCG(0.2)	15	3.656	236.845	240.501
SD-VI-NCG(0.3)	13	3.266	358.343	361.609
SD-VI-NCG(0.4)	16	2.094	261.173	263.267
SD-VI-NCG(0.5) [28]	17	0.705	298.846	299.551
SD-VI-NCG(1.0)	37	1.051	316.221	317.272

* Dstep = decomposition step; Time(M) = solution time of Master-VI in second; Time(S) = solution time of subproblem in second; Time(T) = solution time of the method in second.

Table 4 reports the solution time of simplicial decomposition with multiple NCG subproblems (SD-VI-mNCG). That is, in each iteration of the SD-VI, there are a number of NCG subproblems with different values of $\alpha/2$. I first use three NCG subproblems for my test model. Table 4 summarises the computational performance of SD-VI-3NCG with nine sets of $\alpha/2$. For example, SD-VI-3NCG (0.1,0.2,0.3) consists of three subproblems with $\alpha/2$ = 0.1, 0.2, and 0.3. As expected, on average, SD-VI-3NCG uses fewer decomposition steps than SD-VI-NSD. The range of decomposition steps of SD-VI-3NCG is 11 to 13, while SD-VI-NCG is 13 to 23 shown in Table 3. Moreover, it is observed that the number of decomposition steps exploited in SD-VI-3NCG is more stable. I further use five NCG subproblems, with $\alpha/2$ = 0.1, 0.2, 0.3, 0.4, and 0.5, (SD-VI-5NCG(0.1,…,0.5)), the resulting decomposition steps is 13, in the range of 11–13. 11 decomposition steps are also used in ten NCG subproblems with $\alpha/2$ = 0.1,0.2,…, and 1.0, (SD-VI-5NCG(0.1,…,0.5)).

Table 4. Computational performance of SD-VI-mNCG, m=3, 5, and 10.

Methods	Dstep	Time(M)	Time(S)*	Time(T)	Stopped by NCG
SD-VI-3NCG(0.1,0.2,0.3)	12	2.555	378.484	381.039	NCG(0.3)
SD-VI-3NCG(0.1,0.2,0.4)	12	2.686	383.442	386.108	NCG(0.4)
SD-VI-3NCG(0.1,0.2,0.5)	12	2.461	556.939	559.400	NCG(0.5)
SD-VI-3NCG(0.1,0.3,0.4)	13	2.964	380.704	383.668	NCG(0.4)
SD-VI-3NCG(0.1,0.3,0.5)	12	2.824	506.611	509.435	NCG(0.5)
SD-VI-3NCG(0.1,0.4,0.5)	12	2.487	308.468	310.955	NCG(0.5)
SD-VI-3NCG(0.2,0.3,0.4)	12	2.842	409.687	412.529	NCG(0.4)
SD-VI-3NCG(0.2,0.3,0.5)	11	2.533	448.810	451.343	NCG(0.5)
SD-VI-3NCG(0.2,0.4,0.5)	11	2.584	503.531	506.115	NCG(0.5)
SD-VI-3NCG(0.3,0.4,0.5)	13	2.273	585.607	587.880	NCG(0.5)
SD-VI-5NCG(0.1,…,0.5)	11	2.560	441.654	444.214	NCG(0.5)
SD-VI-10NCG(0.1,…,1.0)	11	2.901	2004.108	2007.009	NCG(1.0)

* It is considered that all subproblems are solved in parallel, and the greatest iterative solution time of a subproblem is used to calculate the total solution time of subproblems. Dstep = decomposition step; Time(M) = solution time of Master-VI in second; Time(S) = solution time of subproblem in second; Time(T) = solution time of the method in second; Stopped by NCG = the SD-VI terminated by the subproblem with the corresponding value of $\alpha/2$.

Remarks on using the CONOPT solver: Solvers, the MINOS and the PATHNLP cannot find the solution after 8 hours. The CONOPT solver can obtain solutions within an hour in all cases. When the CONOPT solver generates a "feasible" solution to the subproblem, not an optimal solution, in particular, in the first few iterations, the corresponding CG cannot be used for the stopping criterion.

In implementing the multiple subproblem cases, there are two phases. The first phase is to allow a few subproblems to generate infeasible, feasible, or optimal solutions due to the capacity of CONOPT. When any subproblem generates an infeasible solution in an iteration, I use its previous solution obtained from the previous iteration to substitute this infeasible solution. By doing so, a set of feasible solutions can be obtained in each iteration until all subproblems obtain their optimal solutions and thereafter, which is the second phase and follows my theoretical results.

4.5. Re-optimization performance (was 4.4)

For studying the re-optimization performance of the SD-VI-mNCG, the asymmetric parameter ***d*** of the test model (see Subsection 4.1) is changed to 0.6 from 0.5. There are three approaches, SD-VI_(0.6), Final_soln(0.5)_(0.6), and All_soln(0.5)_(o.6), to re-optimization analysis of different multi-subproblem methods.

(1) SD-VI_(0.6): Use the exiting coding of SD-VI-NCG(x), where x=0, 0.1, 0.3, 0.5, and 1.0, with new d=0.6, as reference result.

(2) Final_soln(0.5)_0.6: Use the previous final solution obtained from d=0.5 as the feasible starting point to run the re-optimization process for the model with new d=0.6.

(3) All_soln(0.5)_(0.6): Use all previous proposals with d=5 to start the re-optimization process with new d=0.6.

Table 5 summarizes the computational results. As expected, Approach 3, All_soln(0.5)_(0.6), used the smallest decomposition steps in all methods. With Approach 3, the decomposition steps of SD-VI-5NSD is 8, which is smaller than the decomposition steps of SD-VI-NSD(x), ranging from 10

to 23.

Table 5. Re-optimization performance of SD-VI-NCG(x) and SD-VI-5NCG.

Method	Approach	Decomp Steps	Time (Master)	Time (Subproblem)	Time (Total)
SD-VI [26]	SD-VI_(0.6)	149	52.693	30.255	82.948
	Final_soln(0.5)_0.6	157	64.549	31.217	95.766
	All_soln(0.5)_(0.6)	128	67.091	26.051	93.142
SD-VI-NCG(0.1)	SD-VI_(0.6)	33	9.024	241.580	250.604
	Final_soln(0.5)_0.6	20	4.066	141.748	145.814
	All_soln(0.5)_(0.6)	15	5.008	31.955	36.963
SD-VI-NCG(0.3)	SD-VI_(0.6)	18	5.703	309.780	315.483
	Final_soln(0.5)_0.6	14	3.455	182.830	186.285
	All_soln(0.5)_(0.6)	10	2.635	17.156	19.791
SD-VI-NCG(0.5) [28]	SD-VI_(0.6)	24	1.421	272.547	273.968
	Final_soln(0.5)_0.6	18	1.845	309.125	311.970
	All_soln(0.5)_(0.6)	14	1.350	17.640	18.990
SD-VI-NCG(1.0)	SD-VI_(0.6)	50	1.623	391.047	392.670
	Final_soln(0.5)_0.6	34	1.190	478.035	479.225
	All_soln(0.5)_(0.6)	23	0.713	18.547	19.260
SD-VI-5NCG(0.1,…,0.5)	SD-VI_(0.6)	10	1.961	322.283	324.244
	Final_soln(0.5)_0.6	13	4.138	439.282	443.420
	All_soln(0.5)_(0.6)	8	2.410	18.786	20.786

4.6. A comparison summary of the method SD-VI-mNCG, SD-VI, and SD-VI-NCG

Table 6 summarized the performance of the method SD-VI-mNCG, the SD-VI of Lawphongpanich and Hearn [26], and the SD-VI-NCG(0.5) of Larsson et al. [28]. In terms of column generation steps, SD-VI-mNCG is always smaller than SD-VI and SD-NCG. In terms of computational performance, for the model MMOM, the methods with NCG are faster than the SD-VI. However, SD-VI is the fastest method for the model Arezzo. As mentioned above, SD-VI takes more time to solve its master VI problem. It implies that SD-VI may not be the fastest if a larger model incurs a larger number of generated columns in the master problem. In short, solving multiple subproblems in each SD step can reduce the number of iterations. There is no conclusive result for the computational performance. Similar performance results are found in the re-optimization processes.

Table 6. Comparison of the performance of SD-VI-mNCG, SD-VI, and SD-VI-NCG.

Methods	Dstep	Time(M)	Time(S)*	Time(T)	Model
SD-VI [26]	16	1.484	0.032	1.516	MMOM
SD-VI-NCG(0.5)	10	0.424	0.110	0.534	MMOM
SD-VI-3NCG(0.1,0.3,0.5)	7	0.499	0.095	0.594	MMOM
SD-NLP [26]	87	2.129	17.359	19.488	Arezzo
SD-NLP-NCG(0.5) [28]	9	0.172	222.281	222.453	Arezzo
SD-NLP-3NCG(0.1,0.3,0.5)	6	0.125	182.187	182.312	Arezzo
SD-VI [26]	152	46.419	30.422	76.841	Arezzo
SD-VI-NCG(0.5) [28]	17	0.705	298.846	299.551	Arezzo
SD-VI-3NCG(0.1,0.3,0.5)	12	2.824	506.611	509.435	Arezzo

4.7. Other practical examples

The simple example of a multiproduct multipollutant oligopolistic market model in Subsection 4.1 and the application of transportation network equilibrium problems in Subsection 4.2 are practical examples of directly applying the SD-VI-mNCG to the problem. Indeed, the SD-VI-mNCG can be used in some combined methods that solve part of the problems. For example, Sharma et al. [31] developed a method combining the simplicial decomposition, gauss-seidel, and the augmented Lagrangian to solve the coordination problem of the optimal power and traffic flows with EVs. Wang et al. [32] employed SD in their combined method to have a parallel decentralized solution for multi-regional unit commitment with convex AC power flow constraints. It should be noted that the constraint sets of SD-VI-mNCG can be convex constraint sets. In addition, the SD-VI-mNCG can also be employed to solve stochastic transportation problems [28].

5. Conclusions and further research

In this paper, I derive a simplicial decomposition with multiple nonlinear column generation subproblems, which can be applied to nonlinear programming and variational inequalities. I provide convergence properties and derive new stopping condition.

From the computational performance, I generally conclude that for solution methods, simplicial decomposition of variational inequalities with multiple nonlinear column generation subproblems (SD-VI-mNCG), can perform better than the simplicial decomposition of variational inequalities (SD-VI) and the SD-VI with a nonlinear column generation subproblem (SD-VI-NCG) in terms of decomposition steps. The number of column generation steps is greatly reduced, and the long-tail convergence property of SD-VI is alleviated. The same conclusion can be applied to nonlinear programming.

Concerning the selection of the number of subproblems, m, the empirical results show that 3 to 5 is fine.

Concerning the selection of the values α_j, I use $(\alpha_j/2)= 1/2$ as the reference value by which the subproblem becomes diagonalized Newton algorithm. This approach may be beneficial in other column generation schemes, like Dantzig-Wolfe decomposition of VI, which is one of further research directions. Moreover, employing SD-VI-mNCG for different kind of VI problems, like large-scale energy equilibrium problems, can be another research direction.

On the other hand, since there is no conclusive result for the computational performance, it is worthwhile to consider the computational performance as one of the further research topics. For instance, I may not need to wait for all subproblems to be solved to move on to the next SD step. The calculation framework can be one master problem to m subproblems until one of the subproblems solves the VI problem.

Use of AI tools declaration

The author declares that they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

Financial support for William Chung's work came from the Research Grants Council of Hong Kong S.A.R., China (CityU 11500022).

Conflict of interest

The author declares no conflicts of interest.

References

1. E. Çelebi, J. David Fuller, Master problem approximations in Dantzig-Wolfe decomposition of variational inequality problems with applications to two energy market models, *Comput. Oper. Res.*, **40** (2013), 2724–2739. https://doi.org/10.1016/j.cor.2013.05.012
2. F. Murphy, A. Pierru, Y. Smeers, A tutorial on building policy models as mixed-complementarity problems, *Interfaces*, **46** (2016), 465–481. https://doi.org/10.1287/inte.2016.0842
3. W. Chung, Approximate Dantzig-Wolfe decomposition to solve a class of variational inequality problems with an illustrative application to electricity market models, *Expert Syst. Appl.*, **118** (2019), 140–151. https://doi.org/10.1016/j.eswa.2018.09.043
4. E. Bettiol, L. Létocart, F. Rinaldi, E. Traversi, A conjugate direction based simplicial decomposition framework for solving a specific class of dense convex quadratic programs, *Comput. Optim. Appl.*, **75** (2020), 321–360. https://doi.org/10.1007/s10589-019-00151-4
5. D. Uciński, Construction of constrained experimental designs on finite spaces for a modified E-optimality criterion, *Int. J. Appl. Math. Comp.*, **30** (2020), 659–677. https://doi.org/10.34768/amcs-2020-0049
6. M. Guignard, A. Ahlatcioglu, The convex hull heuristic for nonlinear integer programming problems with linear constraints and application to quadratic 0–1 problems, *J. Heuristics*, **27** (2021), 251–265. https://doi.org/10.1007/s10732-019-09433-w
7. P. Delle Site, Pricing of connected and autonomous vehicles in mixed-traffic networks, *Transport. Res. Rec.*, **2675** (2021), 178–192. https://doi.org/10.1177/0361198120985850
8. M. Morabit, G. Desaulniers, A. Lodi, Machine-learning-based column selection for column generation, *Transport. Sci.*, **55** (2021), 815–831. https://doi.org/10.1287/trsc.2021.1045
9. L. Nazareth, Numerical behavior of LP algorithms based upon the decomposition principle, *Linear Algebra Appl.*, **57** (1984), 181–189. https://doi.org/10.1016/0024-3795(84)90186-1
10. J. Nazareth, *Computer solution of linear programs*, New York: Oxford University Press, 1987.
11. J. Ho, Convergence behavior of decomposition algorithms for linear programs, *Oper. Res. Lett.*, **3** (1984), 91–94. https://doi.org/10.1016/0167-6377(84)90048-8
12. M. Lübbecke, J. Desrosiers, Selected topics in column generation, *Oper. Res.*, **53** (2005), 1007–1023. https://doi.org/10.1287/opre.1050.0234
13. G. Dantzig, *Linear programming and extensions*, Santa Monica: RAND Corporation, 1963. https://doi.org/10.7249/R366
14. E. Beale, P. Hughes, R. Small, Experiences in using a decomposition program, *Comput. J.*, **8** (1965), 13–18. https://doi.org/10.1093/comjnl/8.1.13

15. F. Murphy, Column dropping procedures for the generalized programming algorithm, *Manage. Sci.*, **19** (1973), 1310–1321. https://doi.org/10.1287/mnsc.19.11.1310
16. R. O'Neill, Technical note-column dropping in the Dantzig-Wolfe convex programming algorithm: computational experience, *Oper. Res.*, **25** (1977), 148–155. https://doi.org/10.1287/opre.25.1.148
17. R. Marsten, W. Hogan, J. Blankenship, The boxstep method for Large-scale optimization, *Oper. Res.*, **23** (1975), 389–405. https://doi.org/10.1287/opre.23.3.389
18. T. Larsson, M. Patriksson, Simplicial decomposition with disaggregated representation for the traffic assignment problem, *Transport. Sci.*, **26** (1992), 4–17. https://doi.org/10.1287/trsc.26.1.4
19. W. Chung, J. Fuller, Y. Wu, A new decomposition method for multiregional economic equilibrium models, *Oper. Res.*, **54** (2006), 643–655. https://doi.org/10.1287/opre.1060.0274
20. W. Chung, J. David Fuller, Subproblem approximation in Dantzig-Wolfe decomposition of variational inequality models with an application to a multicommodity economic equilibrium model, *Oper. Res.*, **58** (2010), 1318–1327. https://doi.org/10.1287/opre.1090.0803
21. W. Chung, Truncated Dantzig-Wolfe decomposition for a class of constrained variational inequality problems, *Comput. Econ.*, in press. https://doi.org/10.1007/s10614-023-10422-2
22. M. Morabit, G. Desaulniers, A. Lodi, Machine-learning-based arc selection for constrained shortest path problems in column generation, *INFORMS Journal on Optimization*, **5** (2022), 191–210. https://doi.org/10.1287/ijoo.2022.0082
23. S. Kraul, M. Seizinger, J. Brunner, Machine learning–supported prediction of dual variables for the cutting stock problem with an application in stabilized column generation, *INFORMS J. Comput.*, **35** (2023), 692–709. https://doi.org/10.1287/ijoc.2023.1277
24. Y. Dirickx, L. Jennergren, *Systems analysis by multilevel methods*, New York: Wiley, 1979.
25. P. Harker, J. Pang, Finite-dimensional variational inequality and nonlinear complementarity problems: a survey of theory, algorithms and applications, *Math. Program.*, **48** (1990), 161–220. (1990). https://doi.org/10.1007/BF01582255
26. S. Lawphongpanich, D. Hearn, Simplical decomposition of the asymmetric traffic assignment problem, *Transport. Res. B-Meth.*, **18** (1984), 123–133. https://doi.org/10.1016/0191-2615(84)90026-2
27. J. David Fuller, W. Chung, Dantzig-Wolfe decomposition of variational inequalities, *Comput. Econ.*, **25** (2005), 303–326. https://doi.org/10.1007/s10614-005-2519-x
28. T. Larsson, M. Patriksson, C. Rydergren, Applications of simplicial decomposition with nonlinear column generation to nonlinear network flows, In: *Network optimization*, Berlin: Springer, 1997, 346–373. https://doi.org/10.1007/978-3-642-59179-2_17
29. S. Dafermos, Relaxation algorithms for the general asymmetric traffic equilibrium problem, *Transport. Sci.*, **16** (1982), 231–240. https://doi.org/10.1287/trsc.16.2.231
30. A. Nagurney, K. Dhanda, Marketable pollution permits in oligopolistic markets with transaction costs, *Oper. Res.*, **48** (2000), 424–435. https://doi.org/10.1287/opre.48.3.424.12429
31. S. Sharma, Q. Li, W. Wei, An enhanced SD-GS-AL algorithm for coordinating the optimal power and traffic flows with EVs, *IEEE Trans. Smart Grid*, in press. https://doi.org/10.1109/TSG.2024.3358805

32. Z. Wang, G. Li, Y. Xiao, S. Tang, M. Teng, A parallel decentralized solution for multi-regional unit commitment with convex AC power flow constraints, *Proceedings of 7th Asia Conference on Power and Electrical Engineering (ACPEE)*, 2022, 1364–1373. https://doi.org/10.1109/ACPEE53904.2022.9783909

Appendix

Parameters and reference results of MMOM-VI.

For cost function, $f_i = f_i(q_i) = \sum_{d=1}^{s}\left[c_{id}q_{id} + \frac{\beta_{id}}{\beta_{id}+1}K_{id}^{-1/\beta_{id}}q_{id}^{(\beta_{id}+1)/\beta_{id}}\right]$:

	c_{id}		K_{id}		β_{id}	
	d_1	d_2	d_1	d_2	d_1	d_2
i_1	2	5	5	4	1.2	1.9
i_2	6	7	3	6	1.9	1.8
i_3	4.9	6.4	2	4	2.5	2.1

For joint-cost function, $g_i = g_i(e_i, q_i) = \sum_{t=1}^{r}[g1_{it}(e_i^t)^2 + g2_{it}e_i^t + g4_{it}] + \sum_{d=1}^{s} g3_{it}\, q_{id}$:

	$g1_{it}$		$g2_{it}$		$g4_{it}$		$g3_{id}$	
	t_1	t_2	t_1	t_2	t_1	t_2	d_1	d_2
i_1	1.4	1.8	-10	-20	10	14	1.5	1.5
i_2	1.4	2.7	-15	-5	5	15	3.5	2.5
i_3	1.7	2.3	-5	-10	6	2	4.1	3.1

For transaction cost function, $c_{ij}^t = c_{ij}^t(l_{ij}^t) = \emptyset 1_{ijt}(l_{ij}^t)^2 + \emptyset 2_{ijt}l_{ij}^t + \alpha_{ijt}$:

		$j_1.t_1$	$j_1.t_2$	$j_2.t_1$	$j_2.t_2$
$\emptyset 1_{ijt}$	i_1	0.09	0.05	0.05	0.01
	i_2	0.03	0.04	0.09	0.05
	i_3	0.07	0.04	0.03	0.06
$\emptyset 2_{ijt}$	i_1	-7	-8	-1	-5
	i_2	-8	-7	-5	-8
	i_3	-5	-1	-8	-3
α_{ijt}	i_1	0.004	0.009	0.005	0.005
	i_2	0.002	0.006	0.003	0.006
	i_3	0.004	0.007	0.006	0.009

For the price functions of product d: $\rho_d = \rho_d(\sum_{i=1}^{m} q_{id})$, $\rho_d = 5000$.

For constraints $l_{ij}^t - h_{ij}^t e_i^t \geq 0$:

		$j_1.t_1$	$j_1.t_2$	$j_2.t_1$	$j_2.t_2$
h_{ij}^t	i_1	0.09	0.05	0.05	0.01
	i_2	0.03	0.04	0.09	0.05
	i_3	0.07	0.04	0.03	0.06

For $\sum_{i=1}^{m} l0_{ij}^t - \sum_{i=1}^{m} l_{ij}^t \geq 0$, $l0_{ij}^t = 3$.

Reference results are generated by PIES without a decomposition algorithm:

	e_i^t		q_{id}		l_{ij}^t			
	t_1	t_2	d_1	d_2	$j_1.t_1$	$j_1.t_2$	$j_2.t_1$	$j_2.t_2$
i_1	0	1.439	71.768	83.500	0	7.291	0	7.197
i_2	4.815	0.171	55.595	61.952	8.490	1.709	2.407	1.803
i_3	0.304	0	67.514	61.687	0.510	0	6.593	0