Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

An exact solution approach based on column generation and a partial-objective constraint to design a cellulosic biofuel supply chain

Heungjo An [a,*], Wilbert E. Wilhelm [b]

[a] Department of Systems Engineering, King Fahd University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia
[b] Department of Industrial and Systems Engineering, Texas A&M University, TAMUS 3131, College Station, TX 77843-3131, United States

ARTICLE INFO

Article history:
Received 14 September 2013
Received in revised form 21 May 2014
Accepted 15 July 2014
Available online 23 July 2014

Keywords:
Biomass/biofuel supply chain
Embedded generalized flow problem
Column generation
Partial objective constraint
Dynamic programming

ABSTRACT

This study provides an exact solution method to solve a mixed-integer linear programming model that prescribes an optimal design of a cellulosic biofuel supply chain. An embedded structure can be transformed to a generalized minimum cost flow problem, which is used as a sub-problem in a column generation approach, to solve the linear relaxation of the mixed-integer program. This study proposes a dynamic programming algorithm to solve the sub-problem in $O(m)$ time, generating improving path-flows. It proposes an inequality, called the partial objective constraint, which is based on the portion of the objective function associated with binary variables, to underlie a branch-and-cut approach. Computational tests show that the proposed solution approach solves most instances faster than a state-of-the-art commercial solver (CPLEX).

1. Introduction

This paper proposes an exact method to prescribe cellulosic biofuel supply chain design (BSCD), which involves determining facility locations, capacities, and technology types as well as a strategic plan for material flows related to production, transportation, and storage, allowing use of various types of cellulosic biomass. This work is motivated by the fact that cellulosic biomass, the feedstock for second-generation biofuels, offers promise to ameliorate concerns about food-price increases that may have resulted from use of first-generation feed stocks, which are edible crops (e.g., corn, sugar cane) and provide sustainable supply of energy, reducing greenhouse gas (GHG) emissions. However, such feedstock faces unique challenges: it has low energy density and high moisture content, is geographically dispersed, is harvested in specific seasons but must fulfill year-round demand, and loses dry-matter mass in storage. A method that can accommodate these challenges in designing the most profitable biofuel supply chain is vital to the economic viability of this emerging industry. The research objectives of this paper are a BSCD model that deals with the unique features of cellulosic biomass; an effective, exact solution method to solve large-scale instances; and a computational evaluation to benchmark our solution approach with the state-of-the-art, mixed-integer programming commercial solver CPLEX 12.1.

Fig. 1 depicts alternative locations in each of the five echelons of the biofuel supply chain, including feedstock supply, preprocessing, conversion in refineries, distribution, and consumption in customer zones. The term *upstream* refers to echelons that deal with biomass from suppliers to conversion plants; and *downstream*, to echelons that deal with biofuel from conversion plants to customers. Conversion plants themselves are included in both upstream and downstream. Fig. 1 also depicts possible upstream storage locations. An appropriate technology must be prescribed for each facility, depending upon its echelon. For more detail, we refer the reader to our recent studies (An et al., 2011b; An and Searcy, 2012).

BSCD has begun to attract considerable attention. Huang et al. (2010) proposed a multi-period model and applied it in a case study involving the use of waste biomass in California. Their model prescribes locations and capacities of new refineries and material flows from farms to end users over a year-long planning horizon. Ekşioğlu et al. (2010) formulated a multi-period mixed-integer program (MIP) for BSCD, using corn and corn stover biomass, to optimize the network design, modes of transportation, and material flows from feedstock suppliers to end users. Zhu et al. (2010) proposed a MIP to transport switch grass from farms to refineries, prescribing locations of biomass storage and conversion facilities, modes of transportation from farms to refineries, and flows of biomass over multiple time periods. For more detail, we refer the reader to our recent review (An et al., 2011a).

* Corresponding author. Tel.: +966 3 860 1086; fax: +966 3 860 2965.
E-mail addresses: hjan@kfupm.edu.sa, Heungjo.an@gmail.com (H. An).

http://dx.doi.org/10.1016/j.compchemeng.2014.07.011

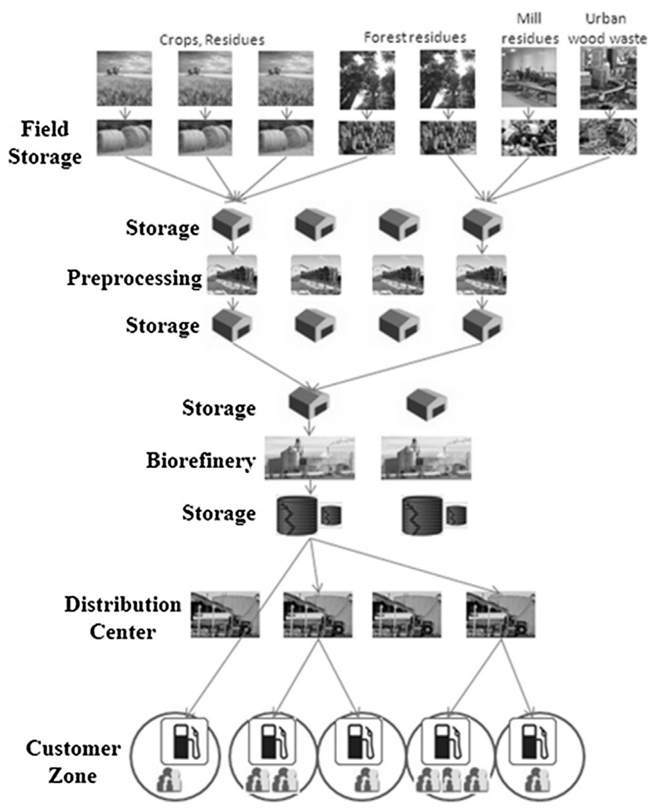


Fig. 1. The biofuel supply chain depicting alternative locations in each echelon.

In particular, this paper presents a BSCD modeling alternative to An et al. (2011b), which formulated a deterministic, time-staged, multi-commodity flow model and demonstrated managerial use in application to a region in Central Texas. Their formulation addressed several unique features of cellulosic feed stocks (e.g., high moisture content, dry matter loss in storage facilities, and single destination for feedstock supply), dealing with commodity-type changes (e.g., biomass with moisture before preprocessing to dry matter afterwards and dry biomass conversion to biofuel) in multi-commodity flow. Their model, as well as the new one we propose, can be used by managers to design profitable supply chains and by government officials to evaluate policies. In contrast to this paper, An et al. (2011b) did not propose any solution methodology; they simply applied CPLEX in their case study.

We solve our BSCD model using a column-generation (CG) decomposition approach to solve its linear relaxation at the root node by exploiting an embedded generalized network flow problem (GFP). In this CG context, we propose a backward-reaching, dynamic programming algorithm (BRA) to solve an uncapacitated, embedded GFP as a sub-problem, generating improving path-flows (i.e., columns) effectively in $O(m)$; the master problem prescribes optimal flow quantities, imposing flow bounds and other side constraints. In addition to the embedded GFP, our BSCD model involves many binary variables. To reduce runtime, we propose an inequality, a *partial objective constraint* (POC), based on the portion of the objective function associated with binary variables.

This paper is organized in four sections. Section 2 describes our BSCD model, an alternative to the multi-commodity flow model proposed by An et al. (2011b). Section 3 explains our solution methods, CG and POCs. Section 4 evaluates the performance of our solution approach through computational tests. Finally, Section 5 gives conclusions and recommendations for future research.

2. Mathematical modeling

Our earlier BSCD formulation (An et al., 2011b) deals with multi-commodity material flows, defining each commodity in the upstream as the combination of biomass type and moisture content, which depends on location and time period. In comparison, the present paper deals with a single commodity, downsizing the An et al. model and, therefore, enhancing solvability (i.e., improving the ability to be solved or, more commonly, allowing reduced run time). This section describes a two-step procedure to define each commodity and the network that represents flows, then presents our model.

We employ two devices that allow all flows to be modeled as a single commodity. The first device eliminates moisture content from biomass flows. To describe this device, let T denote the tonnage of a particular type of biomass that is harvested in a given time period and C denote the cost to transport a ton of biomass, so the total cost of transporting the harvest is CT. If the moisture content (portion by weight) of this harvest is ϕ, the dry-matter tonnage is $(1-\phi)T$. We model the flow of only dry matter, because it provides all of the biomass energy content. To compensate for transporting a lesser tonnage, we adjust transport cost per ton to $C/(1-\phi)$. These two viewpoints are equivalent because they result in the same total transportation cost: $[C/(1-\phi)][(1-\phi)T]=CT$.

The second device models the flow of energy content. Each type of (dry-matter) biomass may provide unique energy content, and tonnages can be converted appropriately into units of energy. A unit of energy that is harvested travels through the supply chain but is reduced by the amount of dry matter loss in storage and by the efficiency of the conversion technology employed. For example, if a harvested unit of energy is subject to a loss of portion $(1-h_1)$ in storage and the efficiency of the conversion process is h_2, the unit of flow that leaves the field provides a supply of less than a unit of energy at the gas pump: h_1h_2. Notice that h_1 and h_2 depend upon technologies used for storage and conversion, respectively.

We can now model the flow through the biofuel supply chain as a GFP on an acyclic graph as depicted in Fig. 2. Because the upstream flow structure for each type of biomass can be treated the same logically (An et al., 2011b), our modeling alternative forms the upstream flow network for each additional type of biomass by duplicating the nodes and arcs in the original network. Each of these flow networks is unique, however, because each type of biomass is associated with unique parameters that define moisture content, dry matter loss, and conversion efficiency. Please note that the sum of transportation capacities of arcs duplicated from an original arc is the same as that of the original arc. Fig. 2 illustrates two upstream substructures, one for each of two types of biomass. The structure of the downstream network depends upon the type of biofuel produced by the conversion technology and could represent new forms of transportation, storage and customer service for ETOH or use of existing infrastructure for drop-in fuel. Nodes i_0 and $i_{\bar{n}}$ and the arcs incident to them are needed to form the GFP structure and are discussed later. Each other node represents an (facility, technology type, location) alternative; and each downward-pointing arc, a transport. BSCD involves selecting, from alternative nodes provided, an optimal set of (facility, technology type, location) combinations, including selections we model in echelon 1 to represent (biomass type, moisture content and harvest season, and source location).

Like the pages of a book, each time period is represented by a layer in the graph. For example, the foreground of Fig. 2 depicts flows in time period t; and the background, period $t+1$. A dashed arc that is incident from a node in period t to the corresponding node in period $t+1$ allows for inventory to be carried (i.e., stored) and dry matter loss could occur on such arcs. The path at the far left of Fig. 2 represents flow (i.e., transport) through the five-echelon

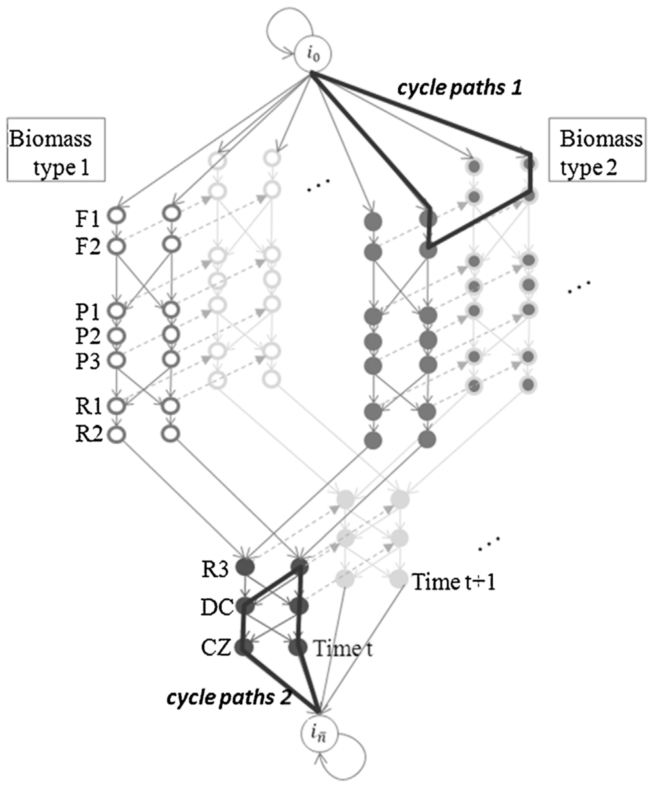


Fig. 2. Flow network.

supply chain, including (1) feedstock supply (harvest at farm F1 and field storage F2 there); (2) preprocessing, where, we assume, any moisture is removed by drying (storage at P1 beforehand, preprocessing at P2, and storage at P3 afterwards); (3) conversion, where inefficiency causes a "loss" of flow (storage of biomass at R1 beforehand, inefficient conversion at R2, storage of biofuel at R3 afterwards); and then distribution of the biofuel through (4) distribution center DC and then to (5) consumption in customer zone CZ. We define index sets of alternative facilities/locations using corresponding designations: F_{F1}, F_{F2}, F_{P1}, F_{P2}, F_{P3}, F_{R1}, F_{R2}, F_{R3}, F_{DC} and F_{CZ}.

The graph includes "dummy" start and end nodes (i.e., i_0 and $i_{\bar{n}}$, respectively, where $\bar{n} = n + 1$), along with directed arcs that connect node i_0 to each supply node and each CZ demand node to $i_{\bar{n}}$. The cost associated with each arc emanating from i_0 is the negative cost of biomass at the supply point; the lower bound for flow out of i_0 is zero; and its upper bound is the capacity of the supply node to produce biomass in each time period. Similarly, the cost of each new arc incident to $i_{\bar{n}}$ is the biofuel selling price; the lower bound for flow into $i_{\bar{n}}$ is zero; and its upper bound is the demand of the CZ demand node in each time period. Following Ahuja et al. (1993) self loops (i_0, i_0) and $(i_{\bar{n}}, \ i_{\bar{n}})$ allow a feasible flow balance at i_0 and $i_{\bar{n}}$; i.e., the flow amount on arc (i_0, i_0) $(i_{\bar{n}}, \ i_{\bar{n}})$ must equal the amount of flow-out of (flow-in to) node i_0 $(i_{\bar{n}})$. This network models generalized flow in the form of flow-circulation (Wayne, 2002). Finally, the two cycle paths shown in Fig. 2 are described in Section 3.1.5.

We now introduce our BSCD model, which involves determining facility locations, capacities, and technology types as well as a strategic plan for material flows related to production, transportation, and storage, allowing use of various types of cellulosic biomass. Table A1, which appears in the Appendix, defines all notation for reader convenience. To clarify our notation, we use a to denote an arc in the original network; and s, in the duplicate network. We use f to denote a facility as well as its location and, for convenience, treat farms and CZs as facilities (for which technology types are irrelevant). Our model incorporates two types of binary decision variables:

x_{fr}: 1 if facility f, which uses technology type r, is opened, 0 otherwise	$f \in F_{OP}$, $r \in R_f$
y_a: 1 if arc a is used, 0 otherwise	$a \in A$

and two types of continuous decision variables:

q_{fr}: Capacity of facility f, which uses technology type r,	$f \in F_{OP}$, $r \in R_f$
z_s: *Flow amount on duplicate arc s*	$s \in A^d$.

We now present model 1 and define notation in the discussion that interprets the model, which follows immediately. For reader convenience, we itemize all notation in Table A1 of the Appendix.

Model 1:

$$Z^* = \text{Max} \sum_{s \in A^d} C_s z_s - \sum_{a \in A} C_a^T y_a - \sum_{f \in F_{OP}} \sum_{r \in R_f} (C_{fr}^O x_{fr} + V_{fr} q_{fr}) \tag{1}$$

s.t.

$$\sum_{r \in R_f} x_{fr} \le 1 \quad f \in F_{OP} \tag{2}$$

$$-Q_f^F x_{fr} + q_{fr} \le 0 \quad f \in F_{OP}, \quad r \in R_f \tag{3}$$

$$\sum_{a \in A_{frt}^+} y_a \le 1 \quad f \in F_{P1} \cup F_{R1}, \quad r \in R_f, \quad t \in T \tag{4}$$

$$-Q_a^T y_a + \sum_{s \in A_a^d} z_s \le 0 \quad a \in A_T \tag{5a}$$

$$-q_{fr} + \sum_{k \in K} \sum_{s \in A_{kfrt}^{dI}} z_s \le 0 \quad f \in F_{WH}, \quad r \in R_f, \quad t \in T \tag{5b}$$

$$-q_{fr} + \sum_{k \in K} \sum_{s \in A_{kfrt}^{d+}} z_s \le 0 \quad f \in F_{PR}, \quad r \in R_f, \quad t \in T \tag{5c}$$

$$z_s \le Q_s \quad s \in A_{kfrt}^{d-}, \quad k \in K_1, \quad f \in F_{F1}, \quad r \in R_f, \quad t \in T \tag{6a}$$

$$z_s \le D_s \quad s \in A_{kfrt}^{d+}, \quad k \in K_2, \quad f \in F_{CZ}, \quad r \in R_f, \quad t \in T \tag{6b}$$

$$\sum_{s \in A_i^+} z_s - \sum_{s \in A_i^-} h_s z_s = 0 \quad i \in N^d \tag{6c}$$

$$x_{fr} \in \{0, 1\} \quad f \in F_{OP}, \quad r \in R_f \tag{7a}$$

$$y_a \in \{0, 1\} \quad a \in A_D \tag{7b}$$

$$q_{fr} \ge 0 \quad f \in F_{OP}, \quad r \in R_f \tag{7c}$$

$$z_s \ge 0 \quad s \in A^d. \tag{7d}$$

Objective (1) is to maximize the present worth of total system profit: the first term gives the revenue C_s for each unit of flow Z_s summed on all arcs $s \in A$; the second term gives the total fixed cost C_a^T associated with arc selections $C_a^T y_a$ summed on arcs $a \in A$; the third term gives the fixed cost C_{fr}^O of opening facilities $C_{fr}^O x_{fr}$ summed on facility $f \in F_{OP}$ and technology $r \in R_f$ alternatives; and the fourth gives the variable cost V_{fr} associated with the capacity of facilities opened $V_{fr} q_{fr}$ summed on $f \in F_{OP}$ and $r \in R_f$. Constraint (2) allows each facility $f \in F_{OP}$ to employ, at most, one technology type $r \in R_f$. Constraint (3) limits the capacity of facility $f \in F_{OP}$, q_{fr}, to be at most Q_f^F, if it is opened; otherwise, it allows no flow from facility f. Constraint (4) requires that each field storage (preprocessing) facility $f \in F_{F2} (F_{P3})$ use one transport link to a single preprocessing (conversion) facility $f \in F_{P1} (F_{R1})$ to facilitate management in the upstream. Constraints (5) impose flow capacity: (5a) Q_a^T for arc $a \in A$, (5b) q_{fr}

for storage facility $f \in F_{WH}$, and (5c) q_{fr} for processing facility $f \in F_{PR}$. Constraints (6) formulate the embedded GFP: (6a) imposes capacity Q_s (i.e., supply limit) to restrict flow on arc s, which is associated with farm $f \in F_{F1}$ and technology $r \in R_f$ in period $t \in T$; (6b) invokes upper bound D_s (i.e., demand) for flow on arc s, which is associated with customer zone $f \in F_{CZ}$ and technology type $r \in R_f$ in period $t \in T$; and (6c) balances flow at each node i in the network, where h_s indicates gain ($h_s > 1$), loss ($h_s < 1$) or unchanged flow ($h_s = 1$) across arc s, which is incident from node i. Due to the nature of the processes we model, BSC flow networks are acyclic $h_s \le 1$ and for all $s \in A^d$ (i.e., there are no 'gainy' arcs). Constraints (7a) and (7b) invoke binary restrictions on decision variables x_{fr} and y_a, respectively. Constraints (7c) and (7d) restrict decision variables q_{fr} and z_s, respectively, to be non-negative.

3. Solution methods

Model 1 involves two features that make solving practical, large-scale instances challenging. The first is the embedded GFP substructure, which includes a huge number of continuous variables. Section 3.1 proposes our CG approach, which incorporates our BRA to generate flow-paths in the un-capacitated version of the embedded GFP. The second is that the model incorporates a large number of binary variables to prescribe facility opening and associated technology types as well as arc selection. Section 3.2 describes our POC, which holds the goal of accelerating branch and bound (B&B).

3.1. Column generation for an embedded GFP

Many problems in areas such as energy, chemical processing, mining, water resources, and finance involve large, embedded GFPs (see Ahuja et al., 1993). GFP has been researched extensively (Vaidya, 1989; Kamath and Palmon, 1995; Wayne, 2002); but, to our knowledge, only a few studies have proposed solution approaches for an embedded GFP. Hultz and Klingman (1978) studied a GFP with a single side constraint. McBride's (1985) solver optimizes linear programs with network substructures, such as an embedded GFP. This paper devises a CG approach, treating the embedded GFP (constraints (6)) as a sub-problem, to solve the linear relaxation of model 1.

3.1.1. Forest- vs. path-flow in CG

If the flow upper bound of each arc is not considered, an optimal solution to the embedded GFP can be viewed as an augmented path-flow. Alternatively, an optimal solution can be considered an augmented forest flow, which is comprised of a set of augmented tree-flows, each of which can be represented as an aggregated set of augmented path-flows. Jones et al. (1993) analyzed the impact of the number of sub-problem extreme points on the performance of CG in a multi-commodity flow problem. They reported that using path-flow solutions in multi-commodity flow sub-problems is computationally superior to using tree flows, because a network admits fewer paths than trees, so that using path-flow solutions can result in substantially fewer master-problem iterations.

Following Jones et al. (1993), we generate columns based on path-flows, imposing the upper bound constraint on each arc flow in the master problem rather than in the sub-problem. Note that, since such an un-capacitated, embedded GFP is a linear program, the master problem of our decomposition provides the same bound at each B&B node as does the linear relaxation of model 1.

3.1.2. Path-based formulation

To implement CG, generating columns from an un-capacitated, embedded GFP, we now transform the arc-based form of model 1 to the path-based form of model 2, the linear relaxation of which is the master problem in our CG decomposition. For reader convenience, Table A2 of the Appendix summarizes the additional notation we use to formulate model 2.

Model 2:

$$Z^* = \text{Max} \sum_{p \in P} C_p \lambda_p - \sum_{a \in A} C_a^T y_a - \sum_{f \in F_{OP}} \sum_{r \in R_f} (C_{fr}^O x_{fr} + V_{fr} q_{fr}) \tag{8}$$

s.t. (2)–(4), and (7a–c)

$$-Q_a^T y_a + \sum_{p \in P} \left(\sum_{s \in A_a^d} H_s^p \right) \lambda_p \le 0 \quad a \in A_T \tag{9a}$$

$$-q_{fr} + \sum_{p \in P} \left(\sum_{k \in K} \sum_{s \in A_{kfrt}^{dI}} H_s^p \right) \lambda_p \le 0 \quad f \in F_{WH}, \quad r \in R_f, \quad t \in T \tag{9b}$$

$$-q_{fr} + \sum_{p \in P} \left(\sum_{k \in K} \sum_{s \in A_{kfrt}^{d+}} H_s^p \right) \lambda_p \le 0 \quad f \in F_{PR}, \quad r \in R_f, \quad t \in T \tag{9c}$$

$$\sum_{p \in P} (H_s^p) \lambda_p \le Q_s \quad s \in A_{i_0}^{d+} \tag{10a}$$

$$\sum_{p \in P} (H_s^p) \lambda_p \le D_s \quad s \in A_{i_{\bar{n}}}^{d-} \tag{10b}$$

$$\lambda_p \ge 0. \quad p \in P \tag{11}$$

Objective (8) is the same as (1) except that the first term is expressed relative to the revenue C_p associated with path p instead of arc s. Constraints (9a)–(9c) and (10a) and (10b), which correspond to constraints (5a)–(5c) and (6a) and (6b), respectively, replace flow variable z_s and relevant technological coefficients with variable λ_p and the appropriate coefficient based on path p using the relationship $z_s = \sum_{p \in P} H_s^p \lambda_p$; $\sum_{s \in A_a^d} z_s = \sum_{s \in A_a^d} \sum_{p \in P} H_s^p \lambda_p$; and $\sum_{s \in A_a^d} z_s = \sum_{p \in P} \left(\sum_{s \in A_a^d} H_s^p \right) \lambda_p$.

Thus, the sum of flow amount on arc s can be represented by the sum of flow amount on paths p that are associated with arc s. Constraints (11) require decision variables λ_p to be non-negative. Note that flows on path $p \in P$ in model 2 satisfy constraints (6c) of model 1. The coefficient H_s^p of λ_p implies that each unit of flow prescribed by the value of λ_p induces flow H_s^p on arc s in path p. Coefficient $H_s^p := \prod_{j \in \bar{A}_{ps}^1} h_j$ represents the product of h_j parameters for each arc j on path p from i_0 through s.

The linear relaxation of model 2 is the master problem of our CG decomposition; it uses an un-capacitated, embedded GFP sub-problem with constraints (6) to identify improving columns. Model 2 can be recast to apply Dantzig–Wolfe decomposition by adjusting the coefficient of λ_p on each arc $s \in \bar{A}_p$ from H_s^p to $H_s^p \pi^p$, where π^p is an extreme-point flow amount on path p, and by incorporating convexity constraint, $\sum_{p \in P} \lambda_p = 1$. However, our preliminary computational tests have shown that using the CG formulation we describe here gives better results for BSCP, so this paper reports only its use. Our strategy is to solve the sub-problem to determine an optimal path from i_0 to $i_{\bar{n}}$, for one unit of flow released from i_0, which is decreased along 'losy' arcs, generating a column that enters the restricted master problem (RMP), which induces a set of path-flows to determine optimal, profitable flow quantities to solve the linear relaxation of model 2.

3.1.3. Sub-problem to generate paths

Table A3 of the Appendix defines additional notation that we use to formulate the sub-problem. Path-based model 2 includes a flow variable (λ_p) for each of many paths. The simplex optimality criterion indicates that entering path p as a column in the master problem basis will improve the current solution if $(wa_p - C_p) < 0$ and that, if $(wa_p - C_p) \geq 0$ for all paths $p \in P$, the current master problem solution is optimal, where w is a vector of duals variables associated with the constraints of model 2; a_p is a column vector in the constraint matrix of model 1 that is associated with variable λ_p; and C_p is the objective function coefficient associated with variable λ_p. By using $\hat{A}_p^u$, which denotes a set of arcs in path p with non-zero entries in row u of the constraint matrix of model 1, each non-zero element of a_p can be expressed in the generalized form $\sum_{s \in \hat{A}_p^u} H_s^p$:

$$a_p = \left(0, \ldots, \sum_{s \in \hat{A}_p^u} H_s^p, \ldots, \sum_{s \in \hat{A}_p^{|U|}} H_s^p\right)^T \tag{12}$$

Let $X := \left\{X \in \{0, 1\}^{|P|} : \sum_{p \in P} X_p = 1, p \in P\right\}$. Given the vector of dual variable values at iteration k, w^k, the sub-problem in our CG procedure may be stated as follows:

SUB:

$$Z_{sub}(w^k) = \min\{(w^k a_p - C_p)X_p : X_p \in X\} = \min\left\{\left[(w_1^k, \ldots, w_{|U|}^k)\left(0, \ldots, \sum_{s \in \hat{A}_p^u} H_s^p, \ldots, \sum_{s \in \hat{A}_p^{|U|}} H_s^p\right)^T - \sum_{s \in \bar{A}_p} H_s^p C_s\right] X_p : X_p \in X\right\}$$

$$= \min\left\{\left[\sum_{s \in \bar{A}_p} H_s^p\left(\sum_{u \in \hat{R}_s} w_u^k - C_s\right)\right] X_p : X_p \in X\right\} = \min\left\{\left(\sum_{s \in \bar{A}_p} C_s^p\right) X_p : X_p \in X\right\}, \tag{13}$$

where the second equality substitutes (12) for a_p, the third re-expresses in terms of coefficients of H_s^p, and the last defines $C_s^p := H_s^p C_s'$ with $C_s' := \sum_{u \in \hat{R}_s} w_u^k - C_s$. The final form of the sub-problem is similar to the shortest path problem but with the cost of arc s, C_s^p, dependent upon path p. The conventional shortest path problem has been researched extensively (Ahuja et al., 1993), but we cannot employ existing algorithms to solve sub-problem (13) because arc cost C_s^p is a function of p, and an arc may have a different cost in association with each possible path. Therefore, we propose BRA to solve sub-problem (13), an unconventional shortest path problem.

3.1.4. *BRA to solve the sub-problem*

This section presents a definition of the shortest distance from each node to end node $i_{\tilde{n}}$, based on arc cost C_s^p, and describes BRA to solve the sub-problem. This sub-section analyzes sub-problem structure, specifies the BRA algorithm, and proposes acceleration techniques.

3.1.4.1. *Problem structure.*

Proposition 1. *The shortest distance from node i_j to end node $i_{\tilde{n}}$, $f[i_j]$, can be defined recursively using*

$$f[i_j] := \begin{cases} 0 & \text{if } i_j = i_{\tilde{n}} \\ \min\limits_{(i_j, i_{j+1}) \in A_{i_j}^{d+}} \{C'_{i_j i_{j+1}} + h_{i_j i_{j+1}} f[i_{j+1}]\} & \text{if } i_j \in N \backslash \{i_{\tilde{n}}\}, \end{cases} \tag{14}$$

where $C'_{i_j i_{j+1}} = \sum_{u \in \hat{R}_{i_j i_{j+1}}} w_u - c_{i_j i_{j+1}}$.

Proof. Given that the series $n(p)$ of nodes on path p is $i_0 - i_1 - i_2 \ldots - i_{n(p)-1} - i_{\tilde{n}}$, and denoting arc s using its start and end nodes (e.g., $s = (i_1, i_2)$), model *SUB* can be re-expressed:

$$Z_{sub}(w^k) = \min\{(C_{i_0 i_1}^p + C_{i_1 i_2}^p + \ldots + C_{i_{n(p)-1} i_{\tilde{n}}}^p) X_p : X_p \in X\} \tag{15}$$

By expanding cost parameters, C_s^p, using the definition $C_s^p = C_s' \prod_{j \in \bar{A}_{ps}^1} h_j$, we obtain

$$Z_{sub}(w^k) = \min\left\{\left[C'_{i_0 i_1} + h_{i_0 i_1} C'_{i_1 i_2} + \ldots + \left(\prod_{j \in \bar{A}^2_{p, i_{n(p)-1}}} h_j\right) C'_{i_{n(p)-1} i_{\tilde{n}}}\right] X_p : X_p \in X\right\} \tag{16}$$

Then, we can rearrange terms, collecting coefficients of arc multipliers to form a nested expression:

$$Z_{sub}(w^k) = \min\{[C'_{i_0 i_1} + h_{i_0 i_1}\{C'_{i_1 i_2} + \cdots + h_{i_{n(p)-2} i_{n(p)-1}}\{C'_{i_{n(p)-1} i_{\tilde{n}}}\}\ldots\}] X_p : X_p \in X\}. \tag{17}$$

To establish the dynamic programming recursion, let $f[i_{\tilde{n}}] : 0$; $f[i_{n(p)-1}] : C'_{i_{n(p)-1} i_{\tilde{n}}}$; $f[i_{n(p)-2}] : C'_{i_{n(p)-2} i_{n(p)-1}} + h_{i_{n(p)-2} i_{n(p)-1}} C'_{i_{n(p)-1} i_{\tilde{n}}} = C'_{i_{n(p)-2} i_{n(p)-1}} + h_{i_{n(p)-2} i_{n(p)-1}} f[i_{n(p)-1}]$; in general, $f[i_j] : C'_{i_j i_{j+1}} + h_{i_j i_{j+1}} f[i_{j+1}]$, so that $f[i_0] : C'_{i_0 i_1} + h_{i_0 i_1} f[i_1]$.

We can now simplify Eq. (17):

$$Z_{sub}(w^k) = \min\{[C'_{i_0 i_1} + h_{i_0 i_1}\{C'_{i_1 i_2} + \cdots + h_{i_{n(p)-2} i_{n(p)-1}}\{f[i_{n(p)-1}]\}\ldots\}] X_p : X_p \in X\} = \min\{f[i_0] X_p : X_p \in X\} \tag{18}$$

By using Eq. (18), we can define the shortest distance from each node j to the end node $i_{\tilde{n}}$. □

Corollary 2 follows from recursion (14) and the dynamic programming principle of optimality:

Corollary 2. *If $P[i_j] = i_j - i_{j+1} - \cdots - i_{\tilde{n}}$ is a shortest path from node i_j to $i_{\tilde{n}}$, sub-path $P[i_{j+1}] = i_{j+1} - \cdots - i_{\tilde{n}}$ is a shortest path from node i_{j+1} to $i_{\tilde{n}}$.*

3.1.4.2. *Computing algorithm.*

We now describe BRA, which is based on Proposition 1 and Corollary 2, to find the shortest path from each node to end node $i_{\tilde{n}}$. Recall that this shortest path problem is not the same as the conventional shortest path problem because of the way arc costs are defined. Let $G = (N^d, A^d)$, where N^d is the set of nodes and A^d is the set of directed arcs, denote the acyclic network on which GFP is defined. BRA orders nodes topologically (Step 1); initializes $f[i_{\tilde{n}}] = 0$ (Step 2); and sets $f[i] = M$ for each $i \in N^d$ (Step 3), where M is a big number; and updates the distance label of each predecessor of node $i_{\tilde{n}}$ (Step 4). Then, it processes nodes in decreasing topological order, updating the distance label of each predecessor i of node j (Step 5). For each node j, it scans

```
Line No.  Step  Operation
1         1.    Order nodes in G topologically
2         2.    $f[i_{\bar{n}}] \leftarrow 0$ and $succ[i_{\bar{n}}] \leftarrow \emptyset$
3         3.    $f[i] \leftarrow M$ and $succ[i] \leftarrow \emptyset$ for each $i \in N^d \backslash \{i_{\bar{n}}\}$
4         4.    For each arc incident to node $i_{\bar{n}}$: $(i, i_{\bar{n}}) \in A^d$
5                 $f[i] \leftarrow C'_{i,i_{\bar{n}}}$ and $succ[i] \leftarrow i_{\bar{n}}$
6               End for
7         5.    For each node $j \in N^d \backslash \{i_{\bar{n}}\}$ in decreasing topological order
8                 For each incoming arc $(i, j) \in A_j^{d-}$
9                     $temp \leftarrow C'_{ij} + h_{ij} f[j]$
10                    If $f[i] > temp$
11                        Then $f[i] \leftarrow temp$ and $succ[i] \leftarrow j$
12                End for
13                $j \leftarrow next\ node$
14              End for
```

Fig. 3. BRA.

incoming arcs. For each arc $(i, j) \in A_j^{d-}$, if $f[i] > C'_{ij} + h_{ij} f[j]$, it sets $f[i] = C'_{ij} + h_{ij} f[j]$. Fig. 3 gives a formal description.

Next, we establish the correctness of BRA by showing that, whenever it processes node *j* in Step 5, the optimal distance label of node *j* has already been determined so that BRA is a label-setting algorithm.

Proposition 3. *BRA is correct.*

Proof. See Appendix A.1.

Now, we analyze the worst case complexity of BRA.

Proposition 4. *The worst case complexity of BRA is O(m), where m is the number of arcs.*

Proof. See Appendix A.2.

Based on the shortest path solution found by BRA, we can construct a column representing path p, on which a unit flow emanates from start node i_0, flows as $H_s^p = \prod_{j \in \bar{A}_{ps}^1} h_j$ units on each arc *s* in path *p*, and ends as $\prod_{j \in \bar{A}_{pi_{\bar{n}}}^2} h_j$ units at node $i_{\bar{n}}$. The coefficient associated with variable λ_p in each row of model 2 represents this flow amount (i.e., H_s^p) on each arc *s* in path *p*.

3.1.5. *Acceleration techniques*

We employ two techniques to accelerate CG convergence. First, we incorporate extra dual cuts. Liang and Wilhelm (2010) generalized extra dual cuts, noting that inserting a polynomial number of extra dual cuts into RMP upon initialization restricts the dual space, potentially accelerating CG convergence. Alvelos and Valerio de Carvalho (2007) incorporated cycle paths as extra dual cuts to generate several additional, feasible flow paths by forming a linear combination of the cycle and flow paths generated by the sub-problem. Similarly, this study uses two portions of the supply chain network, an upper part of upstream and a lower part of downstream, to generate cycle paths as extra dual cuts. Fig. 2 depicts two such cycles: the first, at the northeast corner of the figure, forms a cycle with nodes (counterclockwise) (i_0 – a farm in period *t* – its field storage in period *t* – the same field storage in period *t* + 1 – the farm period *t*+1 – i_0); the second, at the southwest part of the figure, contains nodes (counterclockwise) ($i_{\bar{n}}$ – customer zone i_1 – distribution center i_2 – biofuel storage facility i_3 – distribution center i_4 – customer zone i_5 – $i_{\bar{n}}$). While generating cycles, we disregard arc direction so that some 'losy' arcs become 'gainy' arcs in some cycles generated.

The second technique is based on the conjecture that incorporating multiple columns found to be improving at each of CG iteration may lead to faster CG convergence. After solving the sub-problem once, we incorporate several improving paths (i.e., each with positive reduced cost) rather than incorporating only the best (i.e., most improving) column. However, we control the number of improving paths (columns) that are made available to RMP at each iteration to better manage runtime. Especially in early iterations, dual variable values may be far from optimal so that "good" columns may not be generated. We incorporate only a small portion (1%) of improving paths in initial iterations and, based on preliminary testing, increase that portion to 100% at a preselected iteration number.

3.2. *Partial objective constraint*

This section introduces POC, an inequality based on a portion of objective function (1) associated with binary variables. A few studies have proposed use of an inequality based on the entire objective function. For example, early papers (Woolsey, 1974; Austin and Hanna, 1983) employed such an inequality to accelerate the Gomory fractional algorithm. To our knowledge, Woolsey (1974) described the first use of such an inequality, and Austin and Hanna (1983) developed the bounded dual integer programming algorithm based on an objective cut, which they used to restrict the upper bound on each integer variable. Joseph et al. (1997) incorporated such an inequality in a heuristic to find a good integral solution. Gao (2007) presented another heuristic, which defines a bound on each integer variable based on the optimal solution value to the linear relaxation of the integer program. Recently, a few studies (Hartman et al., 2010; Lin, 2010) have proposed inequalities, which are not necessarily valid, based on a portion of the objective function. Hartman et al. (2010) used dynamic programming to identify inequalities for capacitated lot-sizing, and Lin (2010) devised inequalities based on the objective functions of decomposed sub-problems.

We emphasize that, unlike most inequalities used in integer programming, POC is not necessarily valid; the goal of a POC is to speed B&B solution by cutting off a portion of the (binary) search tree, not to define facets of the convex hull of feasible integer solutions. The following three subsections define POC; relate certain properties

(1) Solve the linear relaxation of model 1 at the root node of the B&B tree.

(2) Set the objective coefficients of binary variables in selected sets I_x and I_y to zero, creating an instance of problem P_{UB_lp}.

(3) Solve problem P_{UB_lp}, using the Dual Simplex method, starting with the optimal root-node solution as the initial feasible solution.

Fig. 4. Procedure to calculate $Z^*_{UB\_lp}$.

including that POC does not cut off all optimal integral solutions;, and give a method to obtain right-hand-side values.

3.2.1. *Definition of POC*

Let $J = \{(f, r) : f \in F_{OP}, r \in R_f\}$ and select indices of binary variables $I_x \subseteq J$ and $I_y \subseteq A$ to construct a POC. Objective function (1) can be expressed as:

$$Z* = \text{Max}\{R - B\} \tag{19}$$

where $R = -\sum_{(f,r)\in J\backslash I_x} C^O_{fr}x_{fr} - \sum_{(f,r)\in J} V_{fr}q_{fr} - \sum_{a\in A_D\backslash I_y} C^T_a y_a + \sum_{s\in A^d} C_s z_s$; and $B = \sum_{(f,r)\in I_x} C^O_{fr}x_{fr} + \sum_{a\in I_y} C^T_a y_a$.

At an optimal solution of model 1, ($\boldsymbol{q}^*$, $\boldsymbol{x}^*$, $\boldsymbol{y}^*$, $\boldsymbol{z}^*$), the optimal objective function value Z^* is

$$Z^* = R^*_{(20)} - B^*, \tag{20}$$

where $R^*_{(20)} = -\sum_{(f,r)\in J\backslash I_x} C^O_{fr}x^*_{fr} - \sum_{(f,r)\in J} V_{fr}q^*_{fr} - \sum_{a\in A_D\backslash I_y} C^T_a y^*_a + \sum_{s\in A^d} C_s z^*_s$ and $B^* = \sum_{(f,r)\in I_x} C^O_{fr}x^*_{fr} + \sum_{a\in I_y} C^T_a y^*_a$.

Letting Z_{inc} denote the objective function value of the current incumbent MIP solution,

$$R^*_{(20)} - B^* = Z^* \geq Z_{inc};$$

re-expressing and incorporating upper bound UB_POC,

$$B^* \leq R^*_{(20)} - Z_{inc} \leq UB\_POC. \tag{21}$$

By using relationship $B^* \leq UB\_POC$ from (21), we define POC based on B in (19) and UB_POC in (21):

$$B = \sum_{(f,r)\in I_x} C^O_{fr}x_{fr} + \sum_{a\in I_y} C^T_a y_a \leq UB\_POC, \tag{22}$$

Note that different POCs can be defined by selecting different subsets I_x and I_y of binary variables.

3.2.2. *POC properties*

Here, we describe several properties of POC.

Proposition 5. *POC may cut off some portion of the B&B tree but not all optimal integral solutions.*

Proof *(.).* See Appendix A.3.

In addition, POC may tighten the bound provided by the linear relaxation of model 1 because it can cut off an optimal solution of this linear relaxation. However, even though POC offers these favorable properties, an appropriate value of UB_POC must be determined to effectively tighten bounds without cutting off all optimal integral solutions. The following section presents a method to determine an appropriate value of UB_POC.

3.2.3. *A method to obtain UB_POC*

We consider upper bound $R^* \leq UB\_R^*$:

$$\sum_{(f,r)\in I_x} C^O_{fr}x_{fr} + \sum_{a\in I_y} C^T_a y_a \leq R^*_{(20)} - Z_{inc} \leq UB\_R^* - Z_{inc} \tag{23}$$

Upper bound UB_R^* can be calculated by solving problem P_{UB}, which is the same as model 1, except the objective function is changed from max $\{R - B\}$ to max $\{R\}$:

$$\mathbf{P_{UB}}: \quad Z^*_{UB} = \max\{R_{(24)}|\text{s.t.}(1) - (7)\} \tag{24}$$

Proposition 6. *$R^*_{(20)} \leq Z^*_{UB}$, where $R^*_{(20)}$, defined following Eq. (20), is a portion of the optimal objective function value of model 1 and Z^*_{UB} is the optimal objective function value of P_{UB}.*

Proof. See Appendix A.4.

Let $Z^*_{UB\_lp}$ be the optimal objective function value of P_{UB_lp}, the linear relaxation of P_{UB}, so that $Z^*_{UB\_lp} \geq Z^*_{UB}$. To determine the value of Z^*_{UB}, problem P_{UB}, a MIP, must be solved. Therefore, even though $Z^*_{UB\_lp} \geq Z^*_{UB}$ and it is just one possible value for UB_POC, it is attractive to use $Z^*_{UB\_lp}$ as an upper bound on R^*, because we can obtain it easily by solving a linear program as described in Fig. 4.

We now re-express POC:

$$\sum_{(f,r)\in I_x} C^O_{fr}x_{fr} + \sum_{a\in I_y} C^T_a y_a \leq Z^*_{UB\_lp} - Z_{inc}. \tag{25}$$

Because a model 1 solution in which all decision variables are zero is feasible, $Z_{inc} \geq 0$. In addition, whenever a new integral incumbent solution is found during the B&B search, the right-hand side of inequality (25), in principal, can be reduced, tightening POC. It may be possible to determine tighter UB_POC values, but we leave such fine improvement for future research to investigate.

3.2.4. *Selection of I_x and I_y*

This paper generates two types of POCs, each based on a particular selection of subsets I_x and I_y:

$$\text{POC1}: \quad \sum_{(f,r)\in H} C^O_{fr}x_{fr} + \sum_{a\in A} C^T_a y_a \leq UB\_POC_1,$$

which includes all binary variables (i.e., $I_x = H$ and $I_y = A$) to provide as strong a cut as possible; and

$$\text{POC2}: \quad \sum_{(f,r)\in H} C^O_{fr}x_{fr} \leq UB\_POC_2,$$

which includes only binary variables x_{fr} (i.e., $I_x = H$ and $I_y = \emptyset$) to focus on the variables that have the largest objective function coefficients (i.e., in comparison with those of the y_a variables) because CPLEX is likely to branch on them early, influencing the trajectory of the search.

In addition, we use POC1&2 to denote application of both POC1 and POC2 inequalities.

4. Results and discussion

The objectives of our computational tests are to evaluate the efficacy of our solution approach and benchmark against state-of-the-art commercial solver CPLEX 12.1. We employ C++ with CPLEX Callable Library under the Windows 7 64-bit operating system with

Table 1
Test instances.

No 1	Name 2	# Farms 3	# Periods 4	# Rows 5	Variables Bin. *X* 6	 Bin. *Y* 7	 Bin. Total 8	 Continuous 9	Embedded GFP network # Nodes 10	 # Arcs 11
1	F9T4	9	4	2813	81	648	729	5150	902	5067
2	F9T6	9	6	4205	81	972	1053	7760	1352	7677
3	F9T12	9	12	8381	81	1944	2025	15,590	2702	15,507
4	F12T4	12	4	4178	108	1152	1260	8594	1202	8484
5	F12T6	12	6	6248	108	1728	1836	12,938	1802	12,828
6	F12T12	12	12	12,458	108	3456	3564	25,970	3602	25,860
7	F15T4	15	4	5760	135	1800	1935	12,902	1502	12,765
8	F15T6	15	6	8615	135	2700	2835	19,412	2252	19,275
9	F15T12	15	12	17,183	135	5400	5535	38,942	4502	38,805
10	F25T4	25	4	12,589	225	5000	5225	33,502	2502	33,275
11	F25T6	25	6	18,845	225	7500	7725	50,352	3752	50,125
12	F25T12	25	12	37,613	225	15,000	15,225	100,902	7502	100,675
13	F34T4	34	4	20,788	306	9248	9554	60,250	3402	59,942
14	F34T6	34	6	31,130	306	13,872	14,178	90,510	5102	90,202
15	F34T12	34	12	62,156	306	27,744	28,050	181,290	10,202	180,982

an Intel(R) Core(TM)2 Quad CPU Q9650 @ 3.00 GHz and a RAM of 8 GB.

We base all parameters in our experiments on the case study reported by An et al. (2011b), which involves nine counties in the Central Texas region. We selected this test bed because it represents the size and scope of the typical BSCD problem. An et al. (2011b) discussed practical issues and conducted a sensitivity analysis with respect to relevant factors. In contrast, we focus on solution methodology.

Table 1 describes our 15 test instances, which we generate based on two factors: the numbers of farms and time periods. Our tests involve five levels (9, 12, 15, 25 and 34) of the former, and three levels (4, 6 and 12 representing quarters, bi-months and months) of the latter. The instances with 34 farms deal with the entire Central Texas region, which comprises thirty-four counties. Columns in Table 1 give (1) case number; (2) case name; numbers of (3) farms, (4) time periods, (6) binary X variables, (7) binary Y variables, (8) total binary variables, (9) continuous variables, and (10) nodes and (11) arcs in the embedded GFP.

Section 4.1 describes the design of our experiments. Section 4.2 reports the runtime to solve the linear relaxation of model 2 at the root node of the B&B search tree. Section 4.3 presents our overall evaluation based on the 15 test instances described in Table 1.

4.1. *Test procedure*

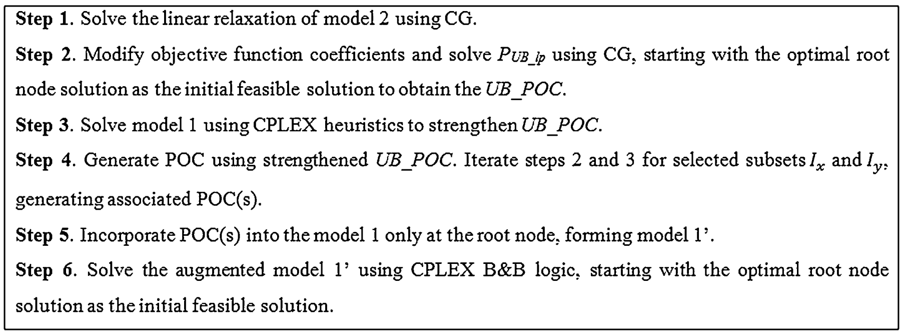
Our solution approach uses CG to solve the linear relaxation of model 2 at the root node and then augments POCs during the B&B search, which is conducted under CPLEX defaults without using CPLEX cuts. Note that CPLEX uses its strong branching rule by default during B&B search. We use CPLEX to determine bounds at nodes other than the root because preliminary tests showed that it can do so faster than our CG approach. We conjecture that this is due, at least in part, to CPLEX's dual simplex algorithm employing an early termination criterion and to the CPLEX code having been refined by professionals over a lengthy period of time, for example, by advanced programming techniques and incorporation of sophisticated algorithms, efficient data structures, and memory management techniques (Bixby, 1994, 2002; Bixby et al., 2000; Bixby and Rothberg, 2007).

Fig. 5 structures our solution procedure, elements of which are discussed in Sections 3 and 4.

4.2. *Solving the linear relaxation of model 2 using CG*

Table 2 compares runtimes required to solve the linear relaxation of model 2 at the root node using both CPLEX and our CG approach. Column 1 gives the names of test instances; Columns 2 and 3 (4 and 5) give CPLEX (CG) results (i.e., CPU runtime and number of simplex iterations). Column 6 reports time reduction = 100*(Column 4)/(Column 2), the percentage of CPLEX runtime required by our CG approach. Column 7 gives the CG iteration number at which we incorporate all improving columns at each iteration (see Section 3.1.5), and column 8 reports the total number of CG iterations required to obtain an optimal solution.

In all test instances, our CG approach solves the linear relaxation of model 2 faster than CPLEX solves model 1. As the instance size increases, the ratio of CG runtime to that of CPLEX decreases (note that the ratio is less than 1.0 on all instances); that is, the runtime advantage of CG increases with instance size. In addition,


Step 1. Solve the linear relaxation of model 2 using CG.

Step 2. Modify objective function coefficients and solve P_{UB_lp} using CG, starting with the optimal root node solution as the initial feasible solution to obtain the *UB_POC*.

Step 3. Solve model 1 using CPLEX heuristics to strengthen *UB_POC*.

Step 4. Generate POC using strengthened *UB_POC*. Iterate steps 2 and 3 for selected subsets I_x and I_y, generating associated POC(s).

Step 5. Incorporate POC(s) into the model 1 only at the root node, forming model 1'.

Step 6. Solve the augmented model 1' using CPLEX B&B logic, starting with the optimal root node solution as the initial feasible solution.

Fig. 5. Solution procedure.

Table 2
Comparison of CPLEX and CG for the linear relaxation of model 2.

Name 1	CPLEX		CG		Time Reduction	CG iteration number	
	Time (s) 2	# Simplex Iter. 3	Time (s) 4	# Simplex Iter. 5	100*Col 4/Col 2 (%) 6	100% criterion 7	Total 8
F9T4	0.22	2000	0.16	2410	71.4%	3	23
F9T6	1.03	7861	0.55	5249	53.0%	3	37
F9T12	4.29	22,256	2.48	19,333	57.8%	1	49
F12T4	0.90	7130	0.44	4513	48.4%	3	26
F12T6	2.43	13,923	1.23	8379	50.7%	9	48
F12T12	8.53	30,950	5.49	6377	64.4%	1	90
F15T4	1.12	7519	0.28	2846	25.0%	3	22
F15T6	3.09	14,875	1.00	4959	32.3%	1	44
F15T12	28.55	69,378	22.33	87,803	78.2%	21	113
F25T4	4.04	13,784	0.69	4411	17.0%	3	30
F25T6	14.24	32,384	3.32	15,249	23.3%	3	55
F25T12	124.38	60,410	32.01	86,287	25.7%	1	95
F34T4	10.73	21,208	1.21	6537	11.3%	2	23
F34T6	64.60	72,608	15.62	49,942	24.2%	3	56
F34T12	328.28	187,251	66.74	13,732	20.3%	1	111

even though CG requires more simplex iterations to solve some instances (i.e., F9T4, F15T12 and F25T12), it is faster than CPLEX. This implies that our decomposition scheme is appropriate and can deal effectively with practical, large-scale BSCD instances.

In many OR models including the proposed BSCD model, as the instance size increases, the constraint matrix becomes sparser (i.e., the portion of non-zero coefficients in the matrix becomes smaller). While CPLEX needs to deal with a large matrix regardless of zero or non-zero elements, a decomposition approach such as CG effectively uses only the non-zero part of the matrix. This principle explains the computational advantage of our CG approach compared to CPLEX as instance size increases. Note that the CG iteration number at which we began to enter all improving columns (column 7) is generally less than 20% of the total number of CG iterations.

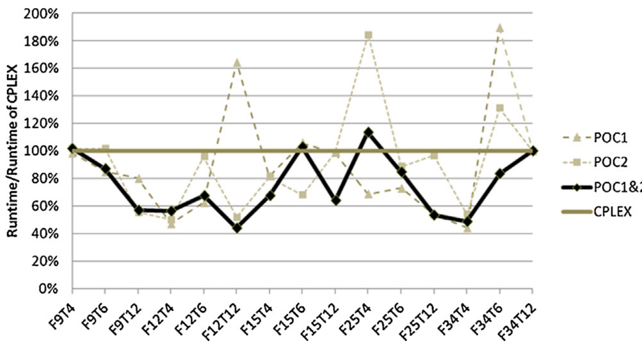


Fig. 6. Comparison of four solution methods.

4.3. Solving BSCD using CG and POC(s)

Using the trivial incumbent solution $Z_{inc} = 0$ and the incumbent solution value prescribed by CPLEX heuristics, we determine the RHS values of POC1 and POC2. Table 3 benchmarks the runtime required by our approach to solve each instance against the default setting of CPLEX B&B without CPLEX cuts. The first column gives the names of test instances. The next seven columns give CPU runtimes to prescribe an optimal, MIP solution using CPLEX, POC1, POC2, POC1&2 (i.e., both POC1 and POC2), POC1′, POC2′, and POC1′&2′, where the last three strengthen POC1, POC2 and POC1&2, respectively, by strengthening UB_POC, as described in Section 4.1 (i.e., Fig. 5). Note that our preliminary computational tests showed that CPLEX was much slower when it used its cuts to solve our test instances than when it did not, so none of the runs we report use CPLEX cuts. Our solution approach is faster than CPLEX with just a few exceptions: two instances (F12T12 and F34T6) for POC1; four instances (F9T4, F9T6, F25T4 and F34T6) for POC2; and three instances (F9T4, F15T6 and F25T4) for POC1&2.

Fig. 6 graphs CPU runtime ratios for POC1, POC2, and POC1&2 relative to CPLEX for each test instance without using CPLEX cuts. A ratio less than 1.0 indicates that the method is faster than CPLEX. POC1 is less effective than POC2 and POC1&2 in our tests. It is interesting to note that the runtime of POC1&2 is between or better than those of POC1 and POC2, with the exception of F12T4. Average runtime ratios relative to CPLEX, expressed in percentages, are 90% for POC1, 91% for POC2, 76% for POC1&2, respectively, implying that POC1&2 outperforms, on average, CPLEX, POC1, and POC2.

The strengthened POC2′ is faster than POC2 for most instances, with the exceptions of F15T12 and F34T6. In contrast, the strengthened POC1′ and POC1′&2′ do not reduce runtimes in our tests. We conjecture that the RHS values of the strengthened POC1′ and POC1′&2′ may not be tight enough to reduce runtimes.

To understand the sensitivity of runtime to the RHS value of POC1, we solve instances using successively strengthened RHS values and here describe the results for F9T6, which is one typical example of the test instances that we investigate. Table 4 gives test results; column 1 gives the test number; and column 2, the POC used. Column 3 gives the RHS value used and column 4 records associated gaps. Columns 5–7 provide performance measures (i.e., CPU runtime, simplex iterations and nodes examined).

Row 1 gives results using CPLEX defaults (without CPLEX cuts) without POC1. Row 2 applies our approach with POC1; and row 3, with POC1′, which uses CPLEX heuristics to strengthen RHS value (see Section 4.3). Rows 4–6 report results for successively strengthened RHS values of POC1. Row 7 strengthens the RHS using the integer optimal solution value. As the RHS of POC1 decreases to the optimal solution value, the runtime, simplex iterations, and nodes examined decrease as well. This result implies that the performance of POC(s) depends on the quality of the RHS value and that strengthened inequalities may serve to accelerate B&B.

4.4. Performance of POC(s)

This section analyzes and compares the performances of POC1, POC2, and POC1&2 in solving instance F12T12 which is a typical example of the test instances that we investigate. Fig. 7 compares best bound, best integer, and the number of iterations for each of four solution methods. In terms of runtime, POC1&2 is the best method; POC2 is second; CPLEX, third; and POC1, the worst. The best bound of CPLEX decreases more slowly than others and the gap provided by CPLEX decreases more slowly than others. By default,

Table 3
Comparison of runtimes.

Name 1	CPLEX 2	POC1 3	POC2 4	POC1&2 5	POC1′ 6	POC2′ 7	POC1′&2′ 8
F9T4	4.1	4.0	4.2	4.2	4.0	3.9	4.0
F9T6	10.5	8.9	10.7	9.2	9.1	10.0	8.5
F9T12	46.6	37.5	25.9	26.8	39.8	22.6	23.7
F12T4	60.8	28.9	30.6	34.4	28.9	29.9	33.9
F12T6	47.7	29.9	45.8	32.3	30.7	44.2	30.8
F12T12	205.0	337.3	106.9	91.2	337.6	105.8	91.4
F15T4	14.9	12.2	12.1	10.1	12.2	12.1	10.1
F15T6	37.4	39.7	25.6	38.6	38.9	24.1	40.3
F15T12	724.2	716.1	714.3	464.3	716.3	718.2	465.5
F25T4	142.9	98.3	263.9	162.4	96.6	228.4	239.8
F25T6	651.7	478.1	580.2	553.3	473.7	580.2	555.1
F25T12	2199.7	1189.0	2129.4	1176.7	1277.4	2138.3	1141.0
F34T4	1184.1	526.2	642.3	581.1	526.2	581.1	526.2
F34T6	2589.8	4907.5	3397.1	2164.0	4936.2	5893.0	2891.9
F34T12	>7200.0[a]	>7200.0	>7200.0	>7200.0	>7200.0	>7200.0	>7200.0

[a] >7200.0: optimal solution was not found within the time limit of 7200 s.

Table 4
Comparison of various strengthened right-hand-side values of POC1 for instance F9T6.

No 1	POC 2	RHS of POC1 3	GAP_RHS[a] 4	Runtime (s) 5	Iteration 6	Nodes 7
1	–	–	–	10.5	62,935	1781
2	POC1	19,864,300	315%	8.9	55,641	1126
3	POC1′	13,984,993	192%	8.9	55,641	1126
4	POC1_a	10,000,000	109%	7.5	49,333	1018
5	POC1_b	8,000,000	67%	6.3	40,349	884
6	POC1_c	6,000,000	25%	6.3	40,349	884
7	POC1_*[b]	4,788,056	0%	6.2	39,839	725

[a] GAP_RHS = 100*(RHS – optimal RHS)/optimal RHS.
[b] Optimal.

CPLEX has a relative MIP gap of 10^{-4} and an absolute MIP gap of 10^{-6}. The number of simplex iterations we report does not include the iterations required by CPLEX strong branching.

One interesting observation is that all methods show very small gaps between best bound and best integer solution after about the 1500th node. In the early phase of the B&B search, an x_{fr} variable, which is associated with opening a facility and has a relatively large objective coefficient, may be selected as a branching variable, because the CPLEX branching rule may tend to branch on a variable that has a large impact on the objective function value. Later, a y_a variable, which selects an arc and has a relatively small objective coefficient, may be selected as a branching variable. Even though fixing y_a to zero may have some impact on material flows, the impact of fixing it to one may be very small. In addition, there are many more y_a variables (e.g., 3564 for F12T12) than x_{fr} variables (e.g., 108 for F12T12).

The box in the upper part of Fig. 7 enlarges the chart for the range between 1st and 1000th node in the B&B search. POC1&2 results

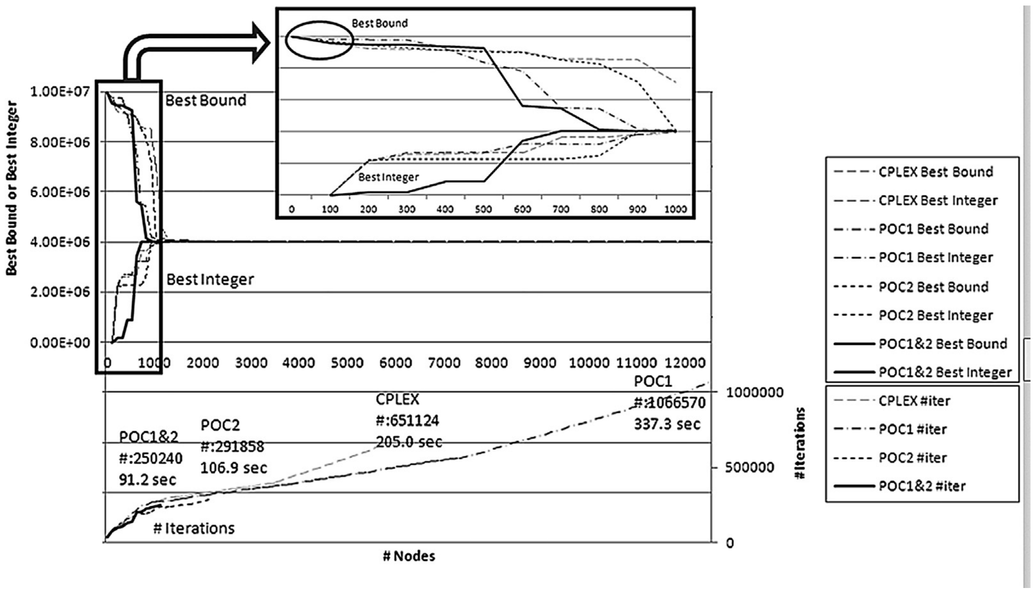


Fig. 7. Comparison of best bound, best integer, and # iterations based on four solution methods (instance: F12T12, scope: the entire B&B search).

in the smallest best bound in the early phase of the B&B search. Even though the sequence of integer incumbents found by POC1&2 increases more slowly than other methods, the gap between best bound and best integer decreases faster than others. This earlier convergence of POC1&2 leads to a faster run time. However, even though the gap between best bound and best integer solution value decreases faster for POC1 than for CPLEX, POC1 searched more nodes and made more simplex iterations than CPLEX, resulting in longer runtime for this particular instance. This might be a result of the particular sequence of branching decisions in combination with the large search space associated with y_a variables.

Although POC1&2 performs better than POC1 or POC2 on average, it is not possible to accurately predict how well it will perform on any particular instance. For example, its performance is between the performances of POC1 and POC2 on F12T6 and somewhat worse on F12T4. This would be partly because it is difficult to predetermine which variable CPLEX strong branching will select to branch. If two methods branch differently in the early phase of their searches, performances may differ dramatically, perhaps for the better, perhaps not. Linderoth and Savelsbergh (1999) noted that the performance of strong branching is quite sensitive to the subset of variables for which dual simplex iterations are performed.

5. Conclusions and future work

This paper presents a new approach to prescribe an optimal solution for BSCD. We show how to model material flows as single-commodity, generalized flows, an alternative to the multi-commodity flow model of An et al. (2011b). Our CG approach solves the linear relaxation our model at the root node of the B&B search tree faster than CPLEX. For example, its runtime is just 11.3% of that of CPLEX on one instance (F34T4). In this CG context, our BRA solves the sub-problem, an uncapacitated, embedded generalized minimum cost circulation problem, generating improving flow-paths (i.e., columns) effectively in $O(m)$. Our approach can be applied to many other important problems that involve an embedded GFP.

We devise POCs, inequalities based on a portion of the objective function, and augment them to the linear relaxation of BSCD to cut off some portion of the B&B search tree with the goal of facilitating solution. Average ratios of runtime of each of our methods to that of CPLEX are 90% for POC1, 91% for POC2, 76% for POC1&2, respectively, implying that POC1&2 outperforms, on average, CPLEX, POC1, and POC2. POC1&2 tends to find an integral solution earlier than POC1 and POC2, so that it may contribute to faster convergence. We expect that POCs could be used effectively to solve other MIPs that involve an objective function structure similar to ours.

This research identifies several fertile topics for future research. The embedded GFP sub-problem, a linear program, can be solved efficiently, but it may be helpful to define a second type of sub-problem that is an integer problem to facilitate solution by allowing bounds at B&B nodes to be tightened. Second, our CG approach can enter all improving columns but it does not identify all alternative optimal shortest paths. Thus, it would be interesting to study the impact of making alternative optimal shortest paths available to the master problem. Third, future research could investigate avenues to improve our column-management techniques with the goal of enhancing solution capability. Fourth, POC may be an attractive candidate for further study that would assess the range of problem types for which it is most effective. Finally, our solution approach offers promise in application to other problems that involve an embedded generalized flow problem. In particular, our CG approach holds potential for solving non-linear problems, especially in the petrochemical industry. Several commercial solvers (e.g., GRTMPS, PIMS, and RPMS) solve non-linear problems by solving a sequence of linear programs (Zhang et al., 1985; Mouret et al., 2011) and our CG approach may be able to solve non-linear problems that embed GFP faster than other successive linear programming algorithms.

Acknowledgement

We acknowledge the supporting environment provided by King Fahd University of Petroleum & Minerals for the publication of this work.

Appendix.

See Tables A1–A3.

A.1. *Proof of proposition 3*

Suppose that BRA has processed nodes $i_{\bar{n}}, i_n, \ldots, i_k$ and their distance labels are optimal. Next, BRA processes node i_{k-1}. Let a shortest path from node i_{k-1} to the node $i_{\bar{n}}$ be $i_{k-1} - i_h - \cdots - i_{\bar{n}}$ where $i_h > i_{k-1}$. By Property 2, path $i_h - \cdots - i_{\bar{n}}$ must be a shortest path from node i_h to $i_{\bar{n}}$. Since BRA processes nodes in decreasing topological order and $(i_{k-1}, i_h) \in A^{d-}_{i_h}$, node i_h is included in $\{i_k, \ldots, i_{\bar{n}}\}$ and the distance label of node i_h, $f[i_h]$, is equal to the shortest distance of the path from node i_h to $i_{\bar{n}}$ by hypothesis. While processing node i_h, arc (i_{k-1}, i_h) must be scanned and the distance label of node i_{k-1}, $f[i_{k-1}]$, set equal to $C'_{i_{k-1},i_h} + h_{i_{k-1}i_h} f[i_h]$, identifying the shortest distance from i_{k-1} to $i_{\bar{n}}$ (i.e., path $i_{k-1} - i_h - \cdots - i_{\bar{n}}$). Therefore, when BRA processes node i_{k-1}, its distance label is already optimal. Even if alternative optima exist, the optimal distance label of node i_{k-1}, $f[i_{k-1}]$, is not affected because the shortest distance associated with each alternative shortest path is the same as $f[i_{k-1}]$. □

A.2. *Proof of proposition 4*

Step 1, ordering nodes topologically can be done in $O(m)$ (Ahuja et al., 1993). Step 2 runs in $O(1)$. Steps 3 and 4 run in $O(n)$ and $O(m)$, respectively, where n is the number of nodes. Step 5 examines each arc just once (lines 7 and 8) and each line (i.e., lines 9–11 and 13) within step 5 runs in $O(1)$ so that total runtime of step 5 is $O(m)$. Since $m \geq n$ according to the network structure of the embedded GFP, the worst-case complexity of BRA is $O(m)$. □

A.3. *Proof of proposition 5*

If the value of *UB_POC* were greater than B^* and it were decreased, POC would tighten restriction (22) on binary variables in sets I_x and I_y so that some feasible integral solutions of model 1 can be rendered infeasible to POC and, thus, cut off by POC.

To prove the second part of Property 5, we show that, after optimizing model 1 then incorporating a POC and re-optimizing, the original optimal solution to model 1 remains optimal. Let ($\boldsymbol{q}^*$, $\boldsymbol{x}^*$, $\boldsymbol{y}^*$, $\boldsymbol{z}^*$) be an optimal solution of model 1 with objective function value $Z(\boldsymbol{q}^*, \boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*) = R^* - B^* = R^* - \left(\sum_{(f,r)\in I_x} C^O_{fr} x^*_{fr} + \sum_{a\in I_y} C^T_a y^*_a\right)$. POC restricts $\sum_{(f,r)\in I_x} C^O_{fr} x_{fr} + \sum_{a\in I_y} C^T_a y_a$ by *UB_POC*, which is an upper bound of $\sum_{(f,r)\in I_x} C^O_{fr} x^*_{fr} + \sum_{a\in I_y} C^T_a y^*_a$. Therefore, optimal solution values of binary variables of model 1 in sets I_x and I_y, (i.e., x^*_{fr}, $(f, r) \in I_x$; and y^*_a, $a \in I_y$) are still feasible with respect to POC. In addition, optimal solution values of binary variables of model 1 not in sets I_x and I_y as well as continuous variables of model 1 (i.e., $\boldsymbol{q}^*$ and $\boldsymbol{z}^*$), are feasible with respect to POC, because POC does not restrict any those variables. This implies that ($\boldsymbol{q}^*$, $\boldsymbol{x}^*$, $\boldsymbol{y}^*$, $\boldsymbol{z}^*$) is feasible with respect to POC.

Table A1
Notation for model 1.

Indices	
a: Arc	$a \in A$
b: Biomass type	$b \in B$
e: Biofuel type	$e \in E$
f: Facility	$f \in F$
i: Duplicate node	$i \in N^d$
k: Commodity	$k \in K$
l: Layer (echelon)	$l \in L$
r: Technology type	$r \in R_f$
s: Duplicate arc	$s \in A^d$
t: Time	$t \in T$
Sets	
A: Directed arcs	$:=A^+_{frt} \cup A^I_{frt}$
A^d: Duplicate directed arcs	$:=A^{d+}_{kfrt} \cup A^{dI}_{kfrt}$
A^d_a: Arcs that are duplicate from original arc a	
$A^+_{frt}(A^-_{frt})$: Directed arcs in period t that start or end at node frt	
A^I_{frt}: Arc for which flow represents inventory held at facility f of type r from period t to period t+1	
$A^{d+}_{kfrt}(A^{d-}_{kfrt})$: Duplicate arcs in period t that start (end) at duplicate node $kfrt$	
A^{dI}_{kfrt}: Duplicate arc that represents inventory held at facility f of type r from period t to period $t+1$	
A_T: Directed arcs associated with transportation	$:=A^+_{frt}, f \in F_{F2} \cup F_{P3} \cup F_{R3} \cup F_{DC}, r \in R_f, t \in T$
B: Feedstock (biomass) types supplied by facility (e.g., farm)	$f \in F_{F1}$
F_l: Candidate locations for facilities in echelon l, feedstock supply site (F_{F1}, F_{F2}), or CZ (F_{CZ}), $l \in L$	
F_{WH0}: Warehouses where biomass is held before preprocessing	$:=F_{F2} \cup F_{P1}$
F_{WH1}: Warehouses where biomass is held	$:=F_{WH0} \cup F_{P3} \cup F_{R1}$
F_{WH2}: Warehouses (i.e., storage tanks) where biofuel is held	$:=F_{R3} \cup F_{DC}$
F_{WH}: Warehouses	$:=F_{WH1} \cup F_{WH2}$
F_{PR}: Process facilities (preprocessing, conversion refinery)	$:=F_{P2} \cup F_{R2}$
F_{OP}: Facilities	$:=F \setminus (F_{F1} \cup F_{CZ})$
F_{UP}: Upstream facilities	$:=F_{F1} \cup F_{F2} \cup F_{P1} \cup F_{P2} \cup F_{P3} \cup F_{R1}$
F_{DOWN}: Downstream facilities	$:=F_{R2} \cup F_{R3} \cup F_{DC} \cup F_{CZ}$
F: All facilities	$:=F_{UP} \cup F_{DOWN}$
K_1: Feedstock types (i.e., commodities)	$:=\{(f,t,b)\}, f \in F, t \in T, b \in B$
K_2: Biofuel commodity	$:=\{e\}, e \in E$
K: Commodities	$:=K_1 \cup K_2$
L: Echelons, $\{F1, F2, P1, P2, P3, R1, R2, R3, DC, CZ\}$	
N^d: Duplicate nodes	
R_f: Types of technologies at facility f, $f \in F_l$	
T: Time periods	
Parameters	
C^T_a: Fixed cost of selecting arc a	
C^O_{fr}: Fixed cost of opening facility f of technology type r	
C_s: Revenue (>0) or cost (<0) for a unit flow on arc s	
D_s: Upper bound on flow on arc s, which is associated with demand of the starting node $kfrt$ of arc s	
h_s: Multiplier associated with arc s	
Q^T_a: Upper bound on flow on arc a	
Q^F_f: Capacity limit of facility f (biomass storage, preprocessing, refinery, or biofuel storage)	
Q_s: Upper bound on flow on arc s, which is associated with supply capacity of the end node $kfrt$ of arc s	
V_{fr}: Variable cost per unit of capacity of opening facility f of technology type r	
V^T_s: Variable cost for a unit of flow on arc s (variable transportation cost on transportation arc, variable holding cost on inventory arc)	
Decision variables	
q_{fr}: Capacity of facility f of technology type r	$f \in F_{OP}, r \in R_f$
x_{fr}: 1 if facility f of type r is opened, 0 otherwise	$f \in F_{OP}, r \in R_f$
y_a: 1 if arc a is used, 0 otherwise	$a \in A$
z_s: Flow amount on duplicate arc s	$s \in A^d$

By way of contradiction, suppose that there exists an optimal solution ($\boldsymbol{q}'$, $\boldsymbol{x}'$, $\boldsymbol{y}'$, $\boldsymbol{z}'$) of model 1 with POC such that $Z_{(POC)}(\boldsymbol{q}', \boldsymbol{x}', \boldsymbol{y}', \boldsymbol{z}') > Z_{(POC)}(\boldsymbol{q}^*, \boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$, where $Z_{(POC)}(\boldsymbol{q}', \boldsymbol{x}', \boldsymbol{y}', \boldsymbol{z}')$ is the objective function value of model 1 with POC evaluated at optimal solution ($\boldsymbol{q}'$, $\boldsymbol{x}'$, $\boldsymbol{y}'$, $\boldsymbol{z}'$) and $Z_{(POC)}(\boldsymbol{q}^*, \boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$, at ($\boldsymbol{q}^*$, $\boldsymbol{x}^*$, $\boldsymbol{y}^*$, $\boldsymbol{z}^*$). Since ($\boldsymbol{q}'$, $\boldsymbol{x}'$, $\boldsymbol{y}'$, $\boldsymbol{z}'$) is also a feasible solution to model 1, $Z(\boldsymbol{q}', \boldsymbol{x}', \boldsymbol{y}', \boldsymbol{z}') \le Z(\boldsymbol{q}^*, \boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$. The feasibility of both ($\boldsymbol{q}^*$, $\boldsymbol{x}^*$, $\boldsymbol{y}^*$, $\boldsymbol{z}^*$) and ($\boldsymbol{q}'$, $\boldsymbol{x}'$, $\boldsymbol{y}'$, $\boldsymbol{z}'$) to model 1 and POC implies that $Z_{(POC)}(\boldsymbol{q}^*, \boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*) = Z(\boldsymbol{q}^*, \boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ and $Z_{(POC)}(\boldsymbol{q}', \boldsymbol{x}', \boldsymbol{y}', \boldsymbol{z}') = Z(\boldsymbol{q}', \boldsymbol{x}', \boldsymbol{y}', \boldsymbol{z}')$. Therefore, $Z_{(POC)}(\boldsymbol{q}', \boldsymbol{x}', \boldsymbol{y}', \boldsymbol{z}') \le Z_{(POC)}(\boldsymbol{q}^*, \boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$. This contradicts the assumption that $Z_{(POC)}(\boldsymbol{q}', \boldsymbol{x}', \boldsymbol{y}', \boldsymbol{z}') > Z_{(POC)}(\boldsymbol{q}^*, \boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$, establishing the proposition. □

Table A2
Notation for model 2.

Indices	
p: path	$p \in P$
Sets	
P: Paths from i_0 to $i_{\bar{n}}$ that satisfy flow balances (6c)	
$\bar{A}^1_{ps}$: Duplicate arcs from i_0 to the one immediately preceding arc s on path p	
$\bar{A}^2_{pi}$: Duplicate arcs from i_0 to node i on path p	
$\bar{A}_p$: Duplicate arcs on path p	
$\bar{N}_p$: Duplicate nodes on path p	
Parameters	
C_p: Variable cost of a unit flow on path p	$:=\sum_{s \in \bar{A}_p} H^p_s C_s \quad p \in P$
H^p_s	$:=\prod_{j \in \bar{A}^1_{ps}} h_j \quad p \in P \quad \text{for} s \in \bar{A}_p, \quad 0 \quad \text{otherwise}$
Decision variables	
λ_p: Flow amount on path p	$p \in P$

Table A3
Notation for sub-problem.

Indices	
u: row	$u \in U$
Sets	
$\hat{A}^u_p$: Arcs in path p with non-zero entries in row u of the constraint matrix of model 1	
$\hat{R}_s$: Rows with non-zero entries in the column associated with arc s of the constraint matrix of model 1	
U: Rows of the constraint matrix of models 1 and 2	
Parameters	
a_p: Column vector of coefficients associated with variable λ_p in the constraint matrix of model 2	$p \in P$
w_u: Dual variable associated with row u in model 2	$u \in U$
Decision variables	
X_p: 1 if path p is used, 0 otherwise	$p \in P$

A.4. *Proof of proposition 6*

By way of contradiction, suppose that there exist optimal solutions $\boldsymbol{u}^*$ of model P_{UB} and $\boldsymbol{v}^*$ of model 1, respectively, such that $R_{(20)}(\boldsymbol{v}^*) = R^*_{(20)} > Z^*_{UB} = R_{(24)}(\boldsymbol{u}^*)$, where $R_{(20)}(\boldsymbol{v}^*)$ is the portion (defined as $R^*_{(20)}$) of the objective solution value of model 1 associated with optimal solution $\boldsymbol{v}^*$, and $Z^*_{UB} = R_{(24)}(\boldsymbol{u}^*)$ is the objective function value of model P_{UB} evaluated at optimal solution $\boldsymbol{u}^*$. Because the constraints of model 1 constitute model P_{UB}, $\boldsymbol{v}^*$ is also a feasible solution with respect to model P_{UB}. This implies that $R^*_{(20)} = R_{(20)}(\boldsymbol{v}^*) = R_{(24)}(\boldsymbol{v}^*) \le R_{(24)}(\boldsymbol{u}^*) = Z^*_{UB}$, where $R_{(24)}(\boldsymbol{v}^*)$ is the objective function value of P_{UB} associated with feasible solution $\boldsymbol{v}^*$. The inequality follows from the fact that $\boldsymbol{v}^*$ is a feasible solution to P_{UB}, a maximizing problem, and $\boldsymbol{u}^*$ is an optimal solution. This contradicts the assumption that $R^*_{(20)} > Z^*_{UB}$. □

References

Ahuja RK, Magnanti TL, Orlin JB. Network flows theory, algorithms, and applications. Prentice-Hall, Inc; 1993.

Alvelos F, Valerio de Carvalho JM. An extended model and a column generation algorithm for the planar multicommodity flow problem. Networks 2007;50(1):3–16.

An H, Wilhelm WE, Searcy SW. Biofuel and petroleum-based fuel supply chain research: a literature review. Biomass Bioenergy 2011a;35(9):3763–74.

An H, Wilhelm WE, Searcy SW. A mathematical model to design a lignocellulosic biofuel supply chain system with a case study based on a region in Central Texas. Bioresour Technol 2011b;102(17):7860–70.

An H, Searcy SW. Economic and energy evaluation of a logistics system based on biomass modules. Biomass Bioenergy 2012;46:190–202.

Austin LM, Hanna ME. A bounded dual (all-integer) integer programming algorithm with an objective cut. Nav Res Logist Q 1983;30:271–81.

Bixby RE. Commentary progress in linear programming. ORSA J Comput 1994;6(1):15–22.

Bixby RE. Solving real-world linear programs: a decade and more of progress. INFORMS J Comput 2002;50(1):3–15.

Bixby RE, Fenelon M, Gu Z, Rothberg EE, Wunderling R. MIP: theory and practice – closing the gap. Kluwer Academic Publishers; 2000. p. 19–49.

Bixby RE, Rothberg E. Progress in computational mixed integer programming – a look back from the other side of the tipping point. Ann Oper Res 2007;149:37–41.

Ekşioğlu SD, Li S, Zhang S, Sokhansanj S, Petrolia D. Analyzing impact of intermodal facilities on design and management of biofuel supply chain. Transp Res Rec: J Transp Res Board 2010;2191:144–51.

Gao P. An efficient bound-and-stopped algorithm for integer linear programs on the objective function hyperplane. Appl Math Comput 2007;185:301–11.

Hartman J, Büyüktahtakin İE, Smith JC. Dynamic-programming-based inequalities for the capacitated lot-sizing problem. IIE Trans 2010;42:915–30.

Huang Y, Chen CW, Fan Y. Multistage optimization of the supply chains of biofuels. Transp Res E 2010;46:820–30.

Hultz J, Klingman D. Solving singularly constrained generalized network problems. Appl Math Optim 1978;4:103–19.

Jones KL, Lustig IJ, Farvolden JM, Powell WB. Multicommodity network flows: the impact of formulation on decomposition. Math Program 1993;62:95–117.

Joseph A, Gass SI, Bryson NA. A computational study of an objective hyperplane search heuristic for the general integer linear programming problem. Math Comput Model 1997;25(10):63–76.

Liang D, Wilhelm WE. A generalization of column generation to accelerate convergence. Math Program Ser A 2010;122(2):349–78.

Lin H. [Dissertation] Models and solution approaches for efficient design and operation of wireless sensor networks [Dissertation]. Texas A&M University; 2010.

Linderoth JT, Savelsbergh MWP. A computational study of search strategies for mixed integer programming. INFORMS J Comput 1999;11(2):173–87.

McBride RD. Solving embedded generalized network problems. Eur J Oper Res 1985;21:82–92.

Mouret S, Grossmann IE, Pestiaux P. A new Lagrangian decomposition approach applied to the integration of refinery planning and crude-oil scheduling. Comput Chem Eng 2011;35:2750–66.

Vaidya PM. Speeding up linear programming using fast matrix multiplication. In: Proc. 30th IEEE Annual symposium on Foundations of Computer Science; 1989. p. 332–7.

Wayne KD. A polynomial combinatorial algorithm for generalized minimum cost flow. Math Oper Res 2002;27(3):445–59.

Woolsey RED. Some practical aspects of solving integer programming problems. In: Zoints S, editor. Linear and integer programming. Prentice Hall; 1974.

Zhang J, Kim N, Lasdon L. An improved successive linear programming algorithm. Manage Sci 1985;31:1312–31.

Zhu X, Li X, Yao Q, Chen Y. Challenges and models in supporting logistics system design for dedicated-biomass-based bioenery industry. Bioresour Technol 2010;102(2):1344–51.