European Journal of Operational Research 221 (2012) 521–532

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

A cut-and-solve based algorithm for the single-source capacitated facility location problem

Zhen Yang [a], Feng Chu [b], Haoxun Chen [a,*]

[a] Industrial Systems Optimization Laboratory, Charles Delaunay Institute and UMR CNRS STMR 6279, University of Technology of Troyes, 12 rue Marie Curie, BP 2060, Troyes 10010, France
[b] Laboratoire d'Informatique, Biologie Intégrative et Systèmes Complexes (IBISC), EA 4526, Université d'Evry Val d'Essonne, 40 Rue du Pelvoux, CE1455 Courcouronnes, 91020 Evry Cedex, France

ARTICLE INFO

Article history:
Received 9 June 2010
Accepted 28 March 2012
Available online 9 April 2012

Keywords:
Facility location
Cutting-plane method
Cut-and-solve

ABSTRACT

In this paper, we present a cut-and-solve (CS) based exact algorithm for the Single Source Capacitated Facility Location Problem (SSCFLP). At each level of CS's branching tree, it has only two nodes, corresponding to the *Sparse Problem* (SP) and the *Dense Problem* (DP), respectively. The SP, whose solution space is relatively small with the values of some variables fixed to zero, is solved to optimality by using a commercial MIP solver and its solution if it exists provides an upper bound to the SSCFLP. Meanwhile, the resolution of the LP of DP provides a lower bound for the SSCFLP. A cutting plane method which combines the lifted cover inequalities and Fenchel cutting planes to separate the 0–1 knapsack polytopes is applied to strengthen the lower bound of SSCFLP and that of DP. These lower bounds are further tightened with a partial integrality strategy. Numerical tests on benchmark instances demonstrate the effectiveness of the proposed cutting plane algorithm and the partial integrality strategy in reducing integrality gap and the effectiveness of the CS approach in searching an optimal solution in a reasonable time. Computational results on large sized instances are also presented.

1. Introduction

The single source capacitated facility location problem, referred to as SSCFLP, is one of the most widely studied combinatorial optimization problems, which can find many applications in various industrial sectors, such as telecommunication network design, production and distribution logistics. Given a set of potential locations of facilities (e.g., plants, warehouses, or other resources), the SSCFLP is to optimally locate a number of facilities with capacity constraints to serve a set of customers with given demand at the minimum total cost composed of fixed opening costs of facilities and the costs of assigning customers to such facilities. Each customer receives its all demand from exactly one open facility.

Let I be a set of potential sites (locations) for constructing the facilities, J a set of customers, d_j denotes the demand of customer j, $j \in J$; f_i the fixed cost for opening a facility at site i, $i \in I$ and the associated capacity b_i for such facility; c_{ij} is the cost of assigning customer j to facility located at site i. All parameters introduced above are assumed to be positive and integer. Define the binary decision variables y_i and x_{ij}: $y_i = 1$ if a facility is located at site $i \in I$ and 0 otherwise; $x_{ij} = 1$ if customer $j \in J$ is served by the facility located at site $i \in I$ and 0 otherwise. The SSCFLP can be formulated as the following binary integer programming:

$$z = \min \quad \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{i\in I} f_i y_i \tag{1}$$

$$\text{s.t.} \quad \sum_{i\in I} x_{ij} = 1 \quad \forall\, j \in J \tag{2}$$

$$\sum_{j\in j} d_j x_{ij} \leqslant b_i y_i \quad \forall\, i \in I \tag{3}$$

$$x_{ij} - y_i \leqslant 0 \quad \forall\, i \in I,\ \forall\, j \in J \tag{4}$$

$$x_{ij}, y_i \in \{0,1\} \quad \forall\, i \in I,\ \forall j \in J \tag{5}$$

The objective function (1) minimizes the total cost of opening facilities and assigning customers to such facilities. Together with integrality constraints (5), constraints (2) guarantee that each customer is served by exactly one facility, while capacity constraints (3) ensure that the total demand assigned to a facility cannot exceed its capacity. The constraints (4) are redundant, but they are very useful in strengthening the linear relaxation lower bound of SSCFLP.

The SSCFLP serves as a basic problem to study other more complex facility location problems, for example, the multi-commodity, multi-stage distribution problem. However, SSCFLP is strongly NP-hard. It has received a considerable attention in the literature.

* Corresponding author. Tel.: +33 325 715642; fax: +33 325 715649.
E-mail addresses: zhenyang.utt@gmail.com (Z. Yang), feng.chu@iup.univ-evry.fr (F. Chu), haoxun.chen@utt.fr (H. Chen).

0377-2217/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2012.03.047

Different methods have been proposed to solve the problem. Most of them are developed based on Lagrangian relaxation, the differences lie only in which constraints are relaxed and how a feasible solution is constructed from the solution of the relaxed problem. Klincewica and Luss (1986) present an algorithm by relaxing the capacity constraints (1). The corresponding Lagrangian subproblems are uncapacitated facility location problems. Pirkul (1987), Barcelo and Casanovas (1984) and Sridharan (1993) develop an algorithm by relaxing the assignment constraints (2). With this relaxation a number of independent sub-problems which are knapsack problems are obtained. Beasley (1993) relaxes both the assignment constraints (2) and the capacity constraints (3) and gives a comprehensive comparison between various Lagrangian heuristics to the problem. Agar and Salhi (1998) modify the Beasley's approach and make it more efficient. Other heuristics have also been proposed, like repeated matching heuristic (Rönnqvist et al., 1999), tabu search (Cortinhal and Captivo, 2003), multi-exchange heuristic (Ahuja et al., 2004) and more recently, scatter search proposed by Contreras and Díaz (2008), a hybrid algorithm that combines Lagriangian heuristic and ant colony system (Chen and Ting, 2008).

Exact methods have also been proposed for the SSCFLP. A branch-and-bound algorithm that partially explores the solution space of the problem can be found in Neebe and Rao (1983). By relaxing assignment constraints (2), Holmberg et al. (1999) obtain simultaneously a lower bound by solving the corresponding Lagrangian dual, and an upper bound by applying a repeated-matching heuristic. These lower and upper bounds are then incorporated into a branch-and-bound algorithm to find an optimal solution to the SSCFLP. Díaz and Fernández (2002) proposed an effective column generation based branch-and-price algorithm for the SSCFLP.

In this paper, we propose a cut-and-solve (CS) approach to solve the SSCFLP exactly and efficiently. The CS first proposed by Climer and Zhang (2006) for the traveling salesman problem, can be regard as a special case of local branching strategy or branch-and-cut, which branches on a set of variables rather than one variable at a time. At each level of its branching tree, the CS has only two nodes, corresponding to the *Sparse Problem* (SP) and the *Dense Problem* (DP), respectively. The SP, whose solution space is relatively small with the values of some variables fixed to zero, can be solved exactly by any MIP solver, i.e. CPLEX, in a reasonable time and its solution if it exists provides an upper bound to the SSCFLP. Meanwhile, a lower bound of the SSCFLP can be found by solving the linear relaxation of the DP. Obviously, if this lower bound is greater than or equal to the best upper bound found so far, an optimal solution of the SSCFLP is found. Otherwise, the current DP is further decomposed into a new SP and a new DP. The above procedure is repeated until a global optimum of the SSCFLP is found.

The efficiency of the CS approach strongly depends on the tightness of the linear relaxation lower bounds of the SP and the DP. As the *Lifted Cover Inequalities* (LCIs) and *Fenchel Cutting Planes* (FCPs) represent the strongest cutting planes for strengthening the linear formulation of a problem with 0–1 knapsack constraints (Kaparis and Letchford, 2010, Ramos and Sáez, 2005). In this paper, a cutting plane method which combines the lifted cover inequalities and Fenchel cutting planes to separate the 0–1 knapsack polytopes is applied to strengthen the lower bound of SSCFLP and that of DP.

As Boccia et al. (2008) present, the generation of FCPs needs to repeatedly solve 0–1 knapsack problems with continuous cost coefficients to optimality within a given accuracy. To avoid rounding error, an integer programming problem is solved to scale the coefficients of the resulting inequality to integer in Boccia et al. (2008). To generate FCP more efficiently, in this paper, we propose a simple scaling method to find valid cuts, which does not require solving the integer programming problem and only 0–1 knapsack problems with integer cost coefficient need to be solved.

The location decision plays an important role in the SSCFLP, which greatly affects the customers-facility assignment as well as the total cost including the assignment costs of customers to facilities and the fixed opening costs of facilities. With the above observation, a partial integrality strategy which preserves only the integrality of location decision variables y while relax the others integer variables to be continuous is proposed. Together with the proposed cutting plane method, the lower bound of SSCFLP and that of DP can be further tightened. Although this strategy spends more computation time than solving a totally linear relaxation of the problem, it can effectively strengthen the formulation of the problem, leading to a much tighter lower bound. As a result, more leaf nodes in the search tree of our CS algorithm can be cut off if this strategy is embedded, and the convergence speed of the algorithm can be increased. In our implementation, the partial integrality strategy is used only when the application of the cutting plane method cannot find violated valid LCIs and FCPs any more. Benefited from the strengthened model, the computational time required by our CS algorithm to solve the SSCFLP is much shorter than the computation time required by a simple application of the CS proposed by Climer and Zhang (2006) without the cutting plane method and the partial integrality strategy used.

Numerical experiments on benchmark instances indicate the efficiency of the proposed cutting plane method and partial integrality strategy in reducing integrality gap and our cut-and-solve approach in searching an optimal solution of the SSCFLP. Computational results on large sized instances with up to 80 potential facilities and 400 customers are also reported.

The remainder of the paper is organized as follows. In Section 2, the knapsack structure of SSCFLP is studied and a cutting plane method is proposed for the separation of 0–1 knapsack polytopes. Section 3 presents in detail our cut-and-solve approach for the exact resolution of the SSCFLP. In Section 4, the computational results on benchmark instances and randomly generated instances are reported and analyzed. Some concluding remarks are given in Section 5.

2. Cutting plane method

The knapsack constraints are important substructures in most facility location problems, for example, the capacitated p-median problem, the capacitated facility location problem and the SSCFLP. Each capacity constraint (3) can be transformed into a knapsack constraint by replacing variable y_i, $i \in I$ by its complement $z_i = 1 - y_i$. Moreover, the redundant constraint

$$\sum_{i\in I} b_i y_i \geqslant \sum_{j\in N} d_j \quad \text{or} \quad \sum_{i\in I} b_i z_i \leqslant \sum_{i\in I} b_i - \sum_{j\in N} d_j, \tag{6}$$

i.e., the overall capacity of all open facilities must be enough to cover the total customers' demand, is also a knapsack constraint. To simplify the presentation, we omit the facility index and introduce a general form of 0–1 knapsack polytope whose convex hull is,

$$X_{KP} = conv\left(\left\{x \in \{0,1\}^{|N|} : \sum_{j\in N} d_j x_j \leqslant b\right\}\right), \tag{7}$$

where $N = \{1,\ldots,n\}$ and $x = (x_1, x_2, \ldots, x_n)$ is a vector of variables.

As a substructure of SSCFLP, the inequalities valid for the convex hulls X_{KP} are also valid for the SSCFLP. Similar to the strategy proposed in Boccia et al. (2008) for strengthening the linear formulation of the CPMP, we explore these knapsack polytopes by using a cutting plane method, which separates the general Lifted Cover Inequalities (LCIs) and the Fenchel cutting planes (FCPs). In the

following subsections, we briefly introduce the LCIs and FCPs and propose a simple scaling method to accelerate the generation of FCPs.

2.1. LCIs and FCPs

2.1.1. Lifted Cover Inequalities (LCIs)

The exact separation of 0–1 knapsack polytope is NP-hard, which has been widely studied in the literature (Balas, 1975; Balas and Zemel, 1978; Boyd (1993a,b, 1994)). Valid inequalities for this polytope, such as *Cover Inequalities* (CIs), *Extended Cover Inequalities* (ECIs), *Lifted Cover Inequalities* (LCIs, an LCI is called simple, if it is generated without down-lifting) and *Weight Inequalities* (WIs), have been proposed as well as their separation algorithms. The application of these inequalities has gained significant success in strengthening the formulation of various problems with knapsack substructures. For detailed surveys, we refer the reader to Crowder et al. (1983), Gu et al. (1998, 1999, 2000), Ferreira et al. (1996), and Klabjan et al. (1998) for the CIs, ECIs and LCIs and Weismantel (1997) for the WIs. For recent development of these inequalities, we recommend the paper of Kaparis and Letchford (2010).

The general LCIs can be formulated as:

$$\sum_{i\in C\setminus D} x_i + \sum_{i\in N\setminus C} \alpha_i \cdot x_i + \sum_{i\in D} \beta_i \cdot x_i \leqslant |C \setminus D| + \sum_{i\in D} \beta_i - 1, \tag{8}$$

where $C \subseteq N$ defines a cover, which satisfies $\sum_{i\in C} d_i > b$, and $D \subset C$ is a subset of C. α_i and β_i are up-lifting and down-lifting coefficients, respectively. For detailed introduction of up-lifting and down-lifting techniques we refer readers to Gu et al. (1998, 1999, 2000) and Kaparis et al. (2010). Note that the calculation of α_i and β_i is NP-hard since it needs to solve a 0–1 knapsack problem to optimality, which is NP-hard.

The knapsack polytope defined by capacity constraint (3) can be expressed as:

$$X_{KP} = conv\left(\left\{(x,y) \in \{0,1\}^{|N|+1} : \sum_{j\in N} d_j x_j \leqslant b \cdot y\right\}\right), \tag{7'}$$

where $N = \{1,\ldots,n\}$ and $x = (x_1, x_2, \ldots, x_n)$. The cover inequalities corresponding to X_{KP} can be written as:

$$\sum_{j\in C} x_j \leqslant (|C| - 1)y \quad \text{or} \quad \sum_{j\in C} (y - x_j) \geqslant y \tag{9}$$

Given a fractional solution pair (x^*, y^*) to be separated, the exact separation of (9) needs to solve the following 0–1 knapsack problem with $x_j = 1$ if item j ($j \in N$) is contained in the cover C and 0 otherwise;

$$\begin{aligned} \theta = \min \quad & \sum_{j\in N} \left(y^* - x_j^*\right) x_j \\ \text{s.t.} \quad & \sum_{j\in N} d_j x_j \geqslant b + 1 \\ & x_j \in \{0,1\}, \quad \forall\, j \in N \end{aligned} \tag{10}$$

Obviously, if $\theta < y^*$, the resulted C defines a violated cover inequality. To separate CIs, we modify the dynamic program proposed by Kaparis and Letchford (2010) and propose a separation algorithm as shown in Fig. 1. The proposed dynamic program is similar to that of Kaparis and Letchford except for the definition of $f(k,r)$ and $g(k)$, and final output condition. Let

$$f(k,r) := \min\left\{\sum_{j=1}^{k}\left(y^* - x_j^*\right)x_j : \sum_{j=1}^{k} d_j x_j = r, x \in \{0,1\}^k\right\},$$
$$k = 1,\ldots,n \quad \text{and} \quad r = 0,\ldots,b,$$

$$g(k) := \min\left\{\sum_{j=1}^{k}\left(y^* - x_j^*\right)x_j : \sum_{j=1}^{k} d_j x_j \geqslant b + 1, x \in \{0,1\}^k\right\},$$
$$k = 1,\ldots,n,$$

the modified dynamic program can be summarized as:

The resulted cover inequality is then lifted to be a LCI following the procedure described in Kaparis and Letchford (2010). In our implementation, the variables in the 0–1 knapsack problem are down or up lifted according to the non-decreasing order of value c_{ij}/d_j, the cost per unit of demand of the SSCFLP.

2.1.2. Fenchel cutting planes (FCPs)

Given a fractional solution vector $x^* = (x_1^*, x_2^*, \ldots, x_n^*)$, the FCPs separation algorithm proposed by Boccia et al. (2008) is to find a hyperplane of the form:

$$\sum_{i\in N} \gamma_i \cdot x_i \leqslant \varepsilon \tag{11}$$

which is valid for X_{KP} but violated by x^* if such a hyperplane exists. (11) can also be normalized by dividing both sides of (11) by ε. We obtain

$$\sum_{i\in N} \gamma_i \cdot x_i \leqslant 1 \tag{11'}$$

The proposed separation algorithm can be summarized as follows (see Fig. 2).

We can observe that the above algorithm needs to repeatedly solve 0–1 knapsack problems with real objective function coefficients in Step 6. The Boccia et al. use a modified MINKNAP algorithm proposed by Ceselli and Righini (2005) to solve the problems to optimality within a given accuracy. Furthermore, to

Set $f(k,r) := \infty$ for $k = 1,\ldots,n$ and $r = 0,\ldots b$. Set $f(0,0) := 0$
Set $g(k) := \infty$ for $k = 1,\ldots,n$.
  For $k = 1,\ldots,n$
    For $r = 0,\ldots b$
      If $f(k-1,r) < f(k,r)$
        Set $f(k,r) := f(k-1,r)$
    For $r = 0,\ldots b - d_k$
      If $f(k-1,r) + \left(y^* - x_k^*\right) < f(k, r+d_k)$
        Set $f(k, r+d_k) := f(k-1,r) + \left(y^* - x_k^*\right)$
    For $r = b - d_k + 1,\ldots,b$
      If $f(k-1,r) + \left(y^* - x_k^*\right) < g(k)$
        Set $g(k) := f(k-1,r) + \left(y^* - x_k^*\right)$
    If $g(k) < y^*$, output the violated CI.

Fig. 1. Dynamic program for generating cover inequalities.

Step 0: Input: a knapsack constraint $\sum_{j\in N} d_j x_j \le b$ and a fractional solution $\mathrm{x}^* \in [0,1]^{|N|}$.

Step 1: Let $N^0 = \{ j \in N \mid x_j^* = 0\}, N^1 = \{ j \in N \mid x_j^* = 1\}$ and $F = N \setminus (N^0 \cup N^1)$.

Step 2: Step 0. Compute the residual capacity $b^r = b - \sum_{j\in N^1} d_j$ and set $k = 0$.

Step 3: Set up the separation linear programming problem LP^k:

$$\begin{aligned} \max \quad & \sum_{j\in F} \gamma_j \cdot x_j^* - \varepsilon \\ s.t. \quad & \sum_{j\in F} \gamma_j \cdot x_j^i \le \varepsilon \quad \forall i = 1,\cdots,k \\ & \gamma_j \in [0,1] \quad \forall j \in F, \end{aligned}$$

where $\mathrm{x}^i = (x_j^i)_{j\in F}, \forall i = 1,\cdots,k$ is a set of solutions that satisfy knapsack constraint $\sum_{j\in F} d_j x_j \le b^r$ generated in Step 6. Obviously, if k=0, then no constraint is added.

Step 4: Solve problem LP^k and denote its optimal solution as γ^k.

Step 5: If $(\gamma^k)^T \cdot \mathrm{x}^* \le \varepsilon$, then no violated cutting plane exists. Stop.

Step 6: Solve the knapsack problem KP^k:

$$\begin{aligned} \max \quad & \sum_{j\in F} \gamma_j^k \cdot x_j \\ s.t. \quad & \sum_{j\in F} d_j \cdot x_j \le b^r \\ & x_j \in \{0,1\} \quad \forall j \in F \end{aligned}$$

and denote the optimal solution of KP^k as $\bar{\mathrm{x}}$.

Step 7: If $(\gamma^k)^T \cdot \bar{\mathrm{x}} > \varepsilon$, then set $k := k+1$ and $\mathrm{x}^k = \bar{\mathrm{x}}$. Return to Step 3;

Otherwise, $(\gamma^k)^T \cdot \mathrm{x} \le \varepsilon$ is violated by the fractional solution x^*, which will be lifted to be a facet of knapsack polytope by up-lifting and down-lifting of variables as described in Step 9 and in Step 10.

Step 8: To avoid any problem induced by rounding errors, the following mixed integer program

$$\begin{aligned} \min \quad & t \\ s.t. \quad & \overline{\gamma_j^k} = t\gamma_j^k, \quad \forall j \in F \\ & \bar{\varepsilon} = t\varepsilon \\ & \overline{\gamma_j^k} \in \mathbf{Z}, \quad \forall j \in F \\ & \bar{\varepsilon} \in \mathbf{Z} \\ & t \ge 1 \end{aligned}$$

is solved by calling a MIP solver to scale the coefficients of the obtained inequality to integer.

Step 9: Down-lift variables in N^1 by solving a series of 0-1 knapsack problems.

Step 10: Up-lift variables in N^0 by solving a series of 0-1 knapsack problems.

Step 11: Output the final violated knapsack facet.

Fig. 2. FCPs algorithm of Boccia et al.

avoid rounding errors incurred due to the continuous coefficients of the resulting violated inequality, a mixed integer programming problem is solved in Step 8 to scale the coefficients to integer. To avoid repeatedly solving knapsack problems with fractional objective function coefficients, we firstly scale the coefficients of objective function in Step 6 to integer in an approximate way. The MINKNAP proposed by Pisinger (1995) is then used to optimally solve the resulted 0–1 knapsack problem. As a result, we need not to solve the integer programming problem to scale the resulting inequality in Step 8. Unlike the scaling method proposed

Step 0: Input: a knapsack constraint $\sum_{j\in N} d_j x_j \le b$, a fractional solution $x^* \in [0,1]^{|N|}$ and a big positive integer M.

Step 1: Let $N^0 = \{ j \in N \mid x_j^* = 0 \}$, $F = N \setminus N^0$ and set $k = 0$.

Step 2: Set up the separation linear programming problem LP^k:

$$\begin{aligned} \max \quad & \sum_{j\in F} \gamma_j \cdot x_j^* \\ s.t. \quad & \sum_{j\in F} \gamma_j \cdot \bar{x}_j^i \le 1 \quad \forall i = 1,\cdots,k \\ & \gamma_j \in [0,1] \quad \forall j \in F, \end{aligned}$$

where $\bar{x}^i = \left(\bar{x}_j^i\right)_{j\in F}, \forall i = 1,\cdots,k$ is a set of solutions that satisfy knapsack constraint $\sum_{j\in F} d_j \bar{x}_j \le b$ obtained in Step 5. Obviously, if k=0, then no constraint is added.

Step 3: Solve problem LP^k and denote its optimal solution by γ^k.

Step 4: If $\sum_{j\in F} \lfloor M\gamma_j^k \rfloor \cdot x_j^* \le M$, then no violated cutting plane exists, stop.

Step 5: Solve the knapsack problem KP^k:

$$\begin{aligned} \max \quad & \sum_{j\in F} \lfloor M\gamma_j^k \rfloor \cdot x_j \\ s.t. \quad & \sum_{j\in F} d_j \cdot x_j \le b \\ & x_j \in \{0,1\} \quad \forall j \in F \end{aligned}$$

and denote the optimal solution of KP^k by $\bar{x}$.

Step 6: If $\sum_{j\in F} \lfloor M\gamma_j^k \rfloor \cdot \bar{x}_j > M$, then set $k := k+1$ and $\bar{x}^k = \bar{x}$. Return to Step 2; Otherwise, $\sum_{j\in F} \lfloor M\gamma_j^k \rfloor \cdot x_j \le M$ is violated by the fractional solution x^*.

Step 7: Up-lift variables in N^0 by solving a series of 0-1 knapsack problems.

Step 8: Output the final violated knapsack facet.

Fig. 3. New FCPs separation algorithm.

in Kaparis and Letchford (2010), we introduce a big scalar M rather than a percentage in our approach.

2.2. FCP generation algorithm

Our scaling method is based on the following observation.

Obvervation: If $\sum_{j\in N}\gamma_j \cdot x_j \le 1$ is valid for X_{KP}, then $\sum_{j\in N}\lfloor M\gamma_j \rfloor \cdot x_j \le M$ is also valid, where M is a positive integer.

Proof. Since $\sum_{j\in N}\lfloor M\gamma_j \rfloor \cdot x_j / M = \sum_{j\in N}(\lfloor M\gamma_j \rfloor / M) \cdot x_j \le \sum_{j\in N}\gamma_j \cdot x_j \le 1$, then $\sum_{j\in N} \lfloor M\gamma_j \rfloor \cdot x_j \le M$. □

Obviously, $\sum_{j\in N}\lfloor M\gamma_j \rfloor \cdot x_j \le M$ can approach $\sum_{j\in N}\gamma_j \cdot x_j \le 1$ for any given accuracy, when $M \to +\infty$.

We can also observe that the resulting inequality of Boccia's algorithm depends on the down-lift order of the variables in set N^1. To further detect the violated inequalities, we set $N^1 = \varnothing$ in our algorithm. In other words, the variables in set $\{j \in N | x_j^* = 1\}$ are also considered in the implicit enumeration generation process of FCPs. Based on the above observations, our new FCP generation algorithm can be described as follows (see Fig. 3).

Note that together with the integrality conditions (5), capacity constraints (3) indicate that if $y_i = 0$, then $x_{ij} = 0$ for any $j \in J$. The Fenchel cuts valid for the knapsack polytope defined by constraints (3) take the form:

$$\sum_{j\in J} \lfloor M\gamma_{ij} \rfloor \cdot x_{ij} + M \cdot z_i \le M, \quad \forall\, i \in I, \tag{12}$$

where $z_i = 1 - y_i$ is a complementary variable of y_i.

2.3. Framework of the cutting plane method

As reported in Kaparis and Letchford (2010), the cutting plane method which combines the LCIs and FCPs is one of the most effective methods in separating 0–1 knapsack polytope. To reduce computational time, the FCPs generation algorithm is called only when the LCIs separation algorithm fails to detect new violated inequalities.

The cutting plane method used in this paper can be described as follows: At each iteration of the cutting plane method, the linear relaxation model of SSCFLP or the DP of SSCFLP is solved. For each knapsack polytope, we first generate the LCIs by using the dynamic programming based heuristic proposed by Kaparis and Letchford (2010). If no violated LCIs can be detected, the above modified FCPs generation algorithm is called to find valid FCPs. Otherwise, we solve the strengthened linear relaxation model. The algorithm is repeated until no new violated FCPs can be found.

3. Cut-and-solve approach

As described above, the generation of LCIs and FCPs is NP-hard. To our best knowledge, no branch-and-cut method has incorporated them at each node of the search tree to strengthen the underlying model due to the requirement of high computational efforts. Boccia et al. (2008) use a Fenchel cutting plane method to tighten the linear formulation of CPMP only at the root node and then call a MIP solver to the problem exactly. In this paper, we incorporate the LCIs and FCPs generated by using a cutting plane method into a cut-and-solve (CS) approach to solve the SSCFLP exactly. The cut-and-solve approach can be regarded as a special local branching strategy or branch-and-cut, which branches on a set of variables rather than a single variable at a time. Since different branching strategies lead to different performances to a specific problem, an appropriate choice of the branching strategy can significantly improve the convergence of a tree searching based exact algorithm. A brief introduction of cut-and-solve approach is given in the following subsection.

3.1. Framework of the cut-and-solve approach

The cut-and-solve (CS) approach that branches on a set of variables was firstly proposed by Climer and Zhang (2006). At each level of its searching tree, it has only two nodes. One node associated with the constraint that the sum of the values of the integer variables in set Ω is smaller than or equal to an integer value σ is called the *Sparse Problem* (SP). The other node associated with the constraint that the sum is greater than or equal to $\sigma + 1$ is called the *Dense Problem* (DP). The constraint associated with the DP is called a *piercing cut*. For a binary integer programming, σ is usually set to 0, thus all variables in set Ω are fixed to 0 in the SP. The solution space of SP can be greatly reduced by selecting an appropriate Ω. As a consequence, the SP can be solved exactly in a reasonable time by a MIP solver and its solution if it exists provides an upper bound to the original problem (minimization problem). The best upper bound $UB_{\min}$ found so far is then updated in case of improvement and the SP is cut off from the searching tree. Meanwhile, a lower bound *LB* for the DP can be obtained by solving its linear relaxation model. Obviously, if this *LB* is greater than or equal to $UB_{\min}$, then the solution corresponding to $UB_{\min}$ is an optimal solution of the original problem. Otherwise, a new variable set Ω is defined, which decomposes the current DP into a new SP and a new DP. The above procedure is repeated until an optimal solution of the original problem is found. The cut-and-solve approach for a minimization binary integer program can be summarized as in Fig. 4.

At each level of the CS, given a fractional solution of P_k, the Ω is defined as the set of variables whose reduced cost is smaller than or equal to a given positive value α.

$$\Omega_k = \{\text{variable}|\text{reduced cost}(\text{variable}) \leqslant \alpha\} \tag{13}$$

For a detailed introduction of CS, we refer readers to Climer and Zhang (2006).

3.2. Lower bound technique

Beside the cutting plane method proposed in Section 2.2, some characteristics of SSCFLP can also be used to tighten its lower bound. For the SSCFLP, the number of opened facilities is an important decision that affects the assignment of customers to facilities as well as the value of the objective function. Since the number must be an integer, in our implementation, we first branch the SSCFLP at the root node by introducing the following two inequalities:

$$\sum_{i\in I} y_i \leqslant \mu \tag{14}$$

$$\sum_{i\in I} y_i \geqslant \mu + 1 \tag{15}$$

Given the fractional solution (x^*, y^*) at the root node, μ is simply defined as

$$\mu = \left\lfloor \sum_{i\in I} y_i^* \right\rfloor \tag{16}$$

The SSCFLP is thus divided into two subproblems, one with constraint (14) is denoted by *P*1 and the other with constraint (15) is denoted by *P*2. Obviously, the minimum of them

Cut-and-solve algorithm

Step 1: Denote the original binary integer programming problem by P_0 and initialize $UB_{\min} = +\infty$, $k := 0$;

Step 2: Define the variable set Ω_k to decompose P_k into two subproblems:

$SP_k : P_k$ with additional constraint $sum(\text{var}iables \in \Omega_k) = 0$

$DP_k : P_k$ with additional constraint $sum(\text{var}iables \in \Omega_k) \geq 1$

Step 3: Solve SP_k to optimality and denote its optimal objective value by UB if it exists. If $UB < UB_{\min}$, then update $UB_{\min} := UB$;

Step 4: Find a lower bound LB for DP_k.

Step 5: If $LB \geq UB_{\min}$, then an optimal solution of P_0 is found, **stop.**

Otherwise, set $k := k+1$ and $P_k := DP_{k-1}$. Return to Step 2.

Fig. 4. Cut-and-solve framework.

$$LB = \min\{LB_1, LB_2\} \tag{17}$$

is a valid lower bound for the SSCFLP, where LB_1 and LB_2 are the lower bound of $P1$ and $P2$, respectively. Note that the cutting plane method proposed in Section 2 can be used to strengthen this lower bound.

3.3. Partial integrality

The partial integrality in this paper means that the integrality of a part of integer decision variables is preserved while relaxing the other integer decision variables to be continuous. Obviously, the lower bound obtained by partial integrality is at least as tight as that of linear relaxation.

In this paper, we preserve the integrality of location decision variables y. First, the SSCFLP is transformed into the classical *Capacitated Facility Location Problem* (CFLP), which is also NP-hard. To solve the CFLP exactly in a reasonable time, the partial integrality is used after the application of the cutting plane method proposed in Subsection 2.2, which is similar to the strategy proposed by Boccia et al. (2008) for the *Capacitated p-median Problem* (CPMP). The formulation of the original problem is strengthened first by using the cutting plane method. Then a MIP solver is called to solve the resulting model exactly. Numerical tests in Section 4 show that the resulted CFLP obtained by partial integrality can be solved efficiently due to a very tight formulation of the original problem generated by the proposed cutting plane method.

Preserving the integrality of location decision variables is based on the following two considerations:

(1) The location decision affects the assignment of customers to facilities as well as the objective value incurred by opening facilities, which plays an important role in the SSCFLP. Keeping the integrality of these variables in the relaxed model of the SSCFLP may greatly improve the lower bound of the problem.
(2) As described in the above subsection, the cut-and-solve approach proposed by Climer and Zhang (2006) uses reduced cost information to define the variables set Ω which is then applied to divide the current dense problem into a new sparse problem and a new dense problem. For the SSCFLP, the set Ω can be defined in a more straightforward manner. Together with the integrality conditions (5), constraints (4) imply that if $y_i = 0$, then $x_{ij} = 0$ for all $j \in J$. The number of decision variables can be greatly reduced if we know that no facility will be opened at a site. The partial integrality strategy helps to determine those "non-promising" location variables. Given a fractional solution (x^k, y^k) of the dense problem at level k derived by applying partial integrality strategy, the site with $y_i^k = 0$ means to some extend a low probability that a facility will be opened at the site. For this reason, the variable set Ω_k at level k of the cut-and-solve searching tree is defined as:

$$\Omega_k = \{i \in I | y_i^k = 0\} \tag{18}$$

Obviously, the cutting plane method in which partial integrality strategy is used can further strengthen the formulation of the problem.

3.4. Upper bound

Based on the fractional solution (x^*, y^*) of DP at any level of the cut-and-solve, a short term memory tabu search is proposed to find a feasible solution and its corresponding upper bound of the SSCFLP. The initial solution of the problem is obtained by an iterative fixation procedure. We first fix those variables with value 1. Then, we allocate one unassigned customer, which has minimum cost $c_{ij}(1 - \bar{x}_{ij}) + f_i(1 - \bar{y}_i)$, to a facility at a time, where $(\bar{x}, \bar{y})$ is the current partial solution. During the fixation procedure, we consider feasible assignments only. The procedure is repeated until all customers have been allocated.

A local search algorithm is then used to improve the initial solution. We explore the classical *shift* and *swap* neighborhoods. The shift neighborhood contains all solutions that differ from the current one in the assignment of one customer whereas the swap neighborhood considers all solutions that can be reached from the current one by changing the assignment of two customers which are assigned to two different facilities in the current solution. At each iteration of the local search procedure the best admissible move is selected and performed until a local optimum is obtained. To further improve the upper bound, a tabu search is applied to the obtained local optimum. The tabu search is similar to the local search algorithm except that we use a tabu list to prevent any solution visited recently to be revisited. At each iteration of the tabu search, the best admissible non-tabu move is selected and performed and its reversion is declared as a tabu move during the next t iterations, where t is the tabu-tenure parameter selected randomly from a range $[T_{\min}, T_{\max}]$. The tabu status of a move will be ignored if it leads to a better solution.

3.5. Exact algorithm for the SSCFLP

In this subsection, we summarize the framework of the proposed exact algorithm for the SSCFLP as follows (see Fig. 5).

4. Numerical results

The performance of our cut-and-solve algorithm has been evaluated on benchmark instances and randomly generated instances. The algorithm was coded in C++ with Visual Studio 2005. The commercial software CPLEX version 12.1 teaching edition (CPLEX, 2009) was used as a MIP solver in the evaluation. We used the default setting of CPLEX except that the relative gap tolerance EpGap is set to be 0. The default values for clique, cover and GUB cover cuts generation are used. The CPLEX will automatically determine how often it should try to generate each class of cuts. All experiments were carried out on an IMB R52 with Intel Pentium 1.73 Giga Hertz processor and 2G RAM.

For the SSCFLP, an important feature is the relationship between the fixed opening costs of facilities and the assignment costs of allocating customers to facilities. Intuitively, when fixed opening costs dominate assignment costs, one may expect a small number of open facilities and small slacks for the facility capacity constraints. On the contrary, when assignment costs are dominating, a larger number of open facilities as well as larger facility capacity slacks may be expected. Thus, to evaluate the performance of our cut-and-solve approach in both cases, we consider three sets of benchmark instances. The first set of 71 benchmark instances (H1 ~ H71) is used in Holmberg et al. (1999) and in Ahuja et al. (2004), in which the assignment costs dominate fixed opening costs for most instances. The second set represents the opposite situation, it contains 57 Delmaire et al. (1999) benchmark instances (D1 ~ D57). The third set contains 20 randomly generated instances that are relatively larger in size than those in set 1 and set 2.

To simplify the presentation of computational results, all column headings in Tables 1–4 are explained in the following:

Step 1: Solve the linear relaxation of SSCFLP. If an integer solution is obtained, then it is an optimal solution of SSCFLP, Stop.

Step 2: Apply the cutting plane method that separates LCIs and FCPs to strengthen the lower bound of SSCFLP and denote the corresponding solution by (x^*, y^*). If an integer solution is obtained, then it is an optimal solution of SSCFLP, Stop; otherwise, based on (x^*, y^*), apply the tabu search algorithm to obtain an upper bound UB of SSCFLP.

Step 3: Let $\mu = \sum_{i \in I} y_i^*$. If μ is fractional, then divide the SSCFLP into two subproblems $P1$ and $P2$ (see Subsection 3.2,) and calculate the lower bound $LB = \min\{LB_1, LB_2\}$, where LB_1 and LB_2 are the lower bounds corresponding to $P1$ and $P2$, respectively. If the solution corresponding to LB is integer, then it is an optimal solution of SSCFLP, Stop; If $LB = UB$, then the solution corresponding to UB is an optimal solution of SSCFLP, Stop.

Step 4: Denote the strengthened SSCFLP as DP_0 and set $k := 0$.

Step 5: Together with the partial integrality strategy, apply the cutting plane method to DP_k and denote the corresponding lower bound as LB^k. If $LB^k \geq UB$, the solution corresponding to UB is already optimal, stop.

Step 6: Based on the lower bound solution of DP_k, apply the tabu search algorithm to find feasible solution to the SSCFLP and update the UB.

Step 7: Divide the current DP_k into two new subproblems SP_{k+1} and DP_{k+1}. Solve SP_{k+1} to optimality by calling an MIP solver. If an optimal solution exists, update the UB.

Set $k := k + 1$ and go to Step 5.

Step 8: If one of the stopping criterions (*i.e.* maximum computational time) is reached, output the best upper and lower bound found so far.

Otherwise, output the optimal solution of SSCFLP.

Fig. 5. Framework of the exact algorithm.

Table 1
Problem sizes of SSCFLP benchmark instances.

Set	Problems	$\|I\|$	$\|J\|$	C_{TOT}/D_{TOT}
1	H1–H12	10	50	1.37–2.06
2	H13–H24	20	50	2.77–3.50
3	H25–H40	30	150	3.03–6.06
4	H41–H55	10–30	70–100	1.52–8.28
5	H56–H71	30	200	1.97–3.95
6	D1–D6	10	20	–
7	D7–D17	15	30	–
8	D18–D25	20	40	–
9	D26–D33	20	50	–
10	D34–D41	30	60	–
11	D42–D49	30	70	–
12	D50–D57	30	90	–

C_{TOT}/D_{TOT}	The ratio of total capacity of the facilities over total demand of the customers
Gap^H	The relative optimality gap in percentage of the lower bound proposed by Holmgerg et al. (1999) at the root node, which is defined as (optimal value − lower bound)/optimal value × 100
Gap^D	The relative optimality gap in percentage of the lower bound proposed by Delmaire et al. (1999) at the root node, which is defined as (optimal value − lower bound)/optimal value × 100
Opt	The optimal cost value
Gap^{CS}	The relative optimality gap in percentage of the lower bound obtained by the cutting plane method with partial integrality strategy at the root node
Dev^{Cplex}	The relative gap achieved by Cplex after a computation of 50,000 s or less, which is defined as (best known upper bound − best lower bound)/best known upper bound × 100
Dev^{CS}	The relative gap achieved by cut-and-solve after a computation of 50,000 s or less, which is defined as (best known upper bound − best lower bound)/best known upper bound × 100
BUB	The best upper bound found so far
$t\,(s)^{LB}$	Computational time in seconds to obtain the lower bound corresponding to gap Gap^{CS}
t (s)	Computational time in seconds to reach an optimal solution
m	Number of branching nodes in the cut-and-solve approach

Table 2
Computational results on Holmberg's benchmarks.

H	Opt	Gap^H	Cplex time	CS1				CS2				CS3			
				Gap^{CS}	t (s)^{LB}	t (s)	m	Gap^{CS}	t (s)^{LB}	t (s)	m	Gap^{CS}	t (s)^{LB}	t (s)	m
1	8848	0.00	0.13	0.14	0.06	0.11	1	**0**	0.12	0.14	0	**0**	0.17	0.19	0
2	7913	0.10	0.3	0.19	0.08	0.45	4	0.02	0.23	0.41	1	**0**	0.3	0.33	0
3	9314	0.01	0.17	**0**	0.03	0.03	0	**0**	0.04	0.04	0	**0**	0.06	0.06	0
4	10714	0.23	0.16	0.2	0.13	0.3	2	**0**	0.69	0.7	0	**0**	0.69	0.7	0
5	8838	0.09	0.2	0.11	0.06	0.11	1	0.006	0.12	0.17	1	0.005	0.14	0.18	1
6	7777	0.01	0.2	0.09	0.06	0.14	1	**0**	0.11	0.11	0	**0**	0.08	0.09	0
7	9488	0.09	0.19	0.12	0.08	0.14	1	0.007	0.19	0.23	1	0.005	0.23	0.28	1
8	11088	0.53	0.19	0.21	0.08	0.13	1	0.006	0.23	0.28	1	0.005	0.33	0.38	1
9	8462	0.10	0.17	0.09	0.05	0.09	1	**0**	0.09	0.12	0	**0**	0.14	0.17	0
10	7617	0.09	0.11	**0**	0.06	0.09	0	**0**	0.12	0.15	0	**0**	0.14	0.17	0
11	8932	0.10	0.16	**0**	0.05	0.08	0	**0**	0.07	0.07	0	**0**	0.07	0.07	0
12	10132	0.17	0.09	0.17	0.06	0.13	1	**0**	0.16	0.17	0	**0**	0.15	0.15	0
13	8252	0.02	0.3	**0**	0.06	0.06	0	**0**	0.04	0.04	0	**0**	0.04	0.04	0
14	7137	0.00	0.2	**0**	0.02	0.02	0	**0**	0.03	0.03	0	**0**	0.03	0.03	0
15	8808	0.01	0.2	**0**	0.03	0.03	0	**0**	0.05	0.05	0	**0**	0.05	0.05	0
16	10408	0.27	0.2	**0**	0.34	0.38	0	**0**	0.26	0.47	0	**0**	0.27	0.47	0
17	8227	0.01	0.33	**0**	0.13	0.16	0	**0**	0.23	0.26	0	**0**	0.23	0.26	0
18	7125	0.01	0.2	**0**	0.02	0.02	0	**0**	0.03	0.03	0	**0**	0.03	0.03	0
19	8886	0.62	0.41	0.02	0.2	0.52	1	0.02	0.23	1.08	1	**0**	0.34	0.37	0
20	10486	0.22	0.34	**0**	0.34	0.38	0	**0**	0.69	0.72	0	**0**	0.67	0.72	0
21	8068	0.01	0.2	**0**	0.03	0.03	0	**0**	0.03	0.03	0	**0**	0.03	0.03	0
22	7092	0.00	0.11	**0**	0.03	0.03	0	**0**	0.03	0.03	0	**0**	0.03	0.03	0
23	8746	0.09	0.16	**0**	0.13	0.17	0	**0**	0.12	0.17	0	**0**	0.12	0.17	0
24	10273	0.67	0.17	0.16	0.19	0.44	1	**0**	0.22	0.26	0	**0**	0.22	0.26	0
25	10630	0.53	1.34	0.18	0.88	1.31	1	0.16	14.76	15.26	1	0.05	18.14	18.64	1
26	10771	0.47	1.16	0.19	0.59	1.97	2	0.19	5.03	6.73	1	0.07	6.72	9.5	1
27	12322	1.01	2.14	0.2	1.02	1.52	1	0.19	25.8	27.09	1	0.15	28.01	28.58	1
28	13722	0.95	1.92	0.18	0.95	1.53	1	0.17	20.22	21.19	1	0.14	18.54	19.01	1
29	12371	0.42	5.23	0.16	1.7	4.61	1	0.06	31.8	35.28	1	0.05	36.12	39.42	1
30	11331	1.99	22.94	0.45	3.66	15.08	4	0.35	21.98	71.9	3	0.2	34.72	95.19	2
31	13331	1.74	36.09	0.38	4.13	14.16	4	0.31	19.58	67.76	3	0.18	35.14	90.51	2
32	15331	1.73	50.55	0.33	3.47	14.48	4	0.27	20.37	71.54	3	0.16	31.04	81.03	2
33	11629	0.46	0.81	0.07	0.78	1.19	1	0.003	7.65	8.11	1	**0**	7.61	7.86	0
34	10632	0.02	0.75	**0**	0.22	0.22	0	**0**	0.39	0.39	0	**0**	0.39	0.39	0
35	12232	0.53	0.94	**0**	0.36	0.36	0	**0**	2.51	2.51	0	**0**	2.51	2.51	0
36	13832	1.24	0.98	0.06	0.73	1.11	1	**0**	7.11	7.11	0	**0**	7.11	7.11	0
37	11258	0.01	0.61	**0**	0.39	0.64	0	**0**	2.45	2.72	0	**0**	2.45	2.72	0
38	10551	0.01	0.64	0.003	0.39	0.81	1	**0**	5.25	5.34	0	**0**	5.22	5.34	0
39	11824	0.01	0.59	**0**	0.16	0.16	0	**0**	5.25	0.17	0	**0**	0.17	0.17	0
40	13024	0.01	0.5	**0**	0.17	0.17	0	**0**	5.25	0.17	0	**0**	0.17	0.17	0
41	6589	0.17	0.23	0.13	0.25	0.59	1	0.04	5.34	6.17	1	**0**	6.31	6.41	0
42	5663	0.64	0.42	0.14	0.5	1.59	2	0.08	3.51	5.42	2	0.008	4.11	6.04	1
43	5214	0.00	0.23	**0**	0.09	0.09	0	**0**	0.09	0.09	0	**0**	0.11	0.11	0
44	7028	0.03	0.28	0.15	0.16	0.3	1	**0**	0.48	0.58	0	**0**	0.48	0.58	0
45	6251	0.02	0.36	**0**	0.08	0.08	0	**0**	0.08	0.08	0	**0**	0.08	0.08	0
46	5651	0.23	0.5	0.02	0.56	1.13	1	**0**	1.53	1.62	0	**0**	1.56	1.64	0
47	6228	0.01	0.28	0.55	0.09	0.3	1	**0**	0.19	0.22	0	**0**	0.19	0.22	0
48	5596	0.20	0.31	0.17	0.31	1.03	2	**0**	0.34	0.42	0	**0**	0.34	0.42	0
49	5302	0.02	0.28	0.08	0.3	0.97	2	**0**	0.11	0.11	0	**0**	0.11	0.11	0
50	8741	0.89	0.7	0.39	0.19	0.89	1	0.31	9.91	15.67	1	0.21	10.64	17.61	1
51	7414	2.60	2.84	0.7	0.7	2.28	2	0.61	7.08	13.47	2	0.22	8.5	13.23	1
52	9178	0.12	0.27	0.2	0.14	0.3	1	0.02	4.5	4.69	1	0.02	5.58	5.76	1
53	8531	0.00	0.41	0.07	0.2	0.39	1	**0**	0.84	0.84	0	**0**	0.8	0.84	0
54	8777	0.01	0.09	0.2	0.22	0.34	1	**0**	4.03	4.03	0	**0**	3.98	4.03	0
55	7654	0.69	0.61	0.25	0.47	1.05	1	0.1	6.23	9.28	1	0.01	6.75	10.95	1
56	21103	0.02	4.27	0.34	3.86	39.78	13	0.12	4.72	28.19	7	0.05	11.04	38.23	3
57	26093	1.34	26.86	0.47	12.02	191.53	23	0.25	17.73	109.37	9	0.06	49.36	172.05	4
58	37239	3.27	302	0.33	18.03	192.13	23	0.18	19.4	105.64	9	0.06	48.12	236.33	6
59	27282	1.22	355.66	0.3	3.83	44.63	3	0.11	13.08	22.64	1	0.01	32.76	52.25	1
60	20534	0.00	0.63	**0**	0.2	0.2	0	**0**	0.2	0.2	0	**0**	0.2	0.2	0
61	24454	0.00	0.73	**0**	0.31	0.31	0	**0**	0.31	0.31	0	**0**	0.31	0.31	0
62	32643	1.11	11.61	0.03	5.58	9.73	1	0.02	21.83	35.51	1	0.004	32.87	54.22	1
63	25105	0.08	0.95	0.05	1.06	1.88	1	0.006	4.89	5.7	1	**0**	5.45	5.9	0
64	20530	0.00	0.58	**0**	0.14	0.14	0	**0**	0.15	0.15	0	**0**	0.15	0.15	0
65	24445	0.00	0.59	**0**	0.19	0.19	0	**0**	0.19	0.19	0	**0**	0.19	0.19	0
66	31415	0.82	16.88	0.12	8.75	19.48	2	0.04	99.97	203.62	1	0.01	125	249.56	1
67	24848	0.30	2.23	0.08	1.55	3.63	1	0.07	4.41	20.28	1	0.01	7.33	26.92	1
68	20538	0.00	0.71	**0**	0.2	0.2	0	**0**	0.15	0.15	0	**0**	0.15	0.15	0
69	24532	0.00	0.89	**0**	0.76	1.2	0	**0**	0.28	0.72	0	**0**	0.28	0.72	0
70	32321	0.53	5.14	0.12	4.72	8.09	1	0.08	45.73	71.94	1	0.03	55.84	91.69	1
71	25540	0.11	1.8	0.03	2.13	2.78	1	**0**	4.84	5.3	0	**0**	4.86	5.3	0
Ave		0.41	12.27	0.12	8.31			0.053		14.39		0.024		19.94	

(In the table, Gap^{CS} in boldface indicates that for the corresponding instance its lower bound at the root node reaches the optimal value.)

Table 3
Computational results on Delmaire's benchmarks.

D	Opt.	Gap^D	Cplex time	CS1				CS2				CS3			
				Gap^{CS}	$t(s)^{LB}$	t (s)	m	Gap^{CS}	$t(s)^{LB}$	T (s)	m	Gap^{CS}	$t(s)^{LB}$	t (s)	m
1	2014	1.74	0.8	4.76	0.08	0.89	3	0.98	0.34	0.83	2	0.26	0.51	0.76	1
2	4251	0.41	2.22	2.2	0.06	0.52	1	0.35	0.51	0.65	1	0.03	0.75	0.8	1
3	6051	0.01	1.88	0.53	0.06	0.17	1	***0***	0.44	0.44	0	***0***	0.47	0.47	0
4	7168	0.6	1.33	1.48	0.03	0.45	1	0.44	1.17	1.36	1	0.35	1.69	1.87	1
5	4551	0.02	1.95	1.96	0.06	1.34	2	0.12	0.36	0.76	2	0.04	0.41	1.36	2
6	2269	**0**	0.25	1.11	0.06	0.22	2	**0**	0.3	0.3	0	**0**	0.33	0.33	0
7	4366	0.62	4.19	3.37	0.36	2.03	5	0.28	1.87	4.54	2	0.0014	2.92	6.08	1
8	7926	0.57	26.3	0.95	0.23	5.95	1	0.5	1.94	2.14	1	0.04	2.28	2.52	1
9	2480	0.85	4.58	2.05	0.11	0.94	1	0.76	0.44	0.64	1	0.76	0.45	0.66	1
10	23112	0.07	4.86	0.24	0.15	1.91	1	0.05	1.42	1.58	1	0.047	1.42	1.59	1
11	3447	0.11	15.39	2.13	0.36	20.11	11	***0***	1.53	1.54	0	***0***	1.55	1.55	0
12	3711	0.24	41.33	1.68	0.48	6.48	9	0.28	1.95	3.67	3	0.1	2.63	3.95	1
13	3760	0.04	33.64	1.04	0.26	2.33	8	0.42	1	1.73	2	0.2	1.62	3.66	2
14	5965	0.71	5.42	2.09	0.11	2.76	1	0.55	2.64	3.84	1	0.41	3.59	5.2	1
15	7816	0.26	4.34	0.85	0.16	4.33	2	0.26	0.62	0.86	1	0.26	0.66	0.89	1
16	11543	0.28	6.52	0.58	0.09	2.84	1	0.18	0.67	1.04	1	0.18	0.67	1.06	1
17	9884	0.17	6.14	0.66	0.25	2	1	0.17	2.58	2.94	1	0.15	3.44	3.84	1
18	15607	**0**	21.83	0.44	0.36	1.91	1	**0**	2.7	2.84	1	**0**	2.73	2.86	1
19	18683	0.05	3.58	0.17	0.45	0.87	1	0.05	0.95	1.11	1	0.05	0.97	1.13	1
20	26561	0.07	239.6	0.41	0.39	86.33	1	0.04	2.58	7.12	1	0.04	2.7	7.26	1
21	7295	0.105	162.8	2.07	0.79	11818	23	0.11	4.87	6.65	1	0.1	5.87	8.83	1
22	3271	0.03	6.72	1.5	0.55	24.36	2	0.1	3.36	4.31	1	0.05	4.98	7.08	1
23	6036	0.01	5.97	0.53	0.5	1.36	2	0.03	3.47	5.05	1	0.03	4.76	6.86	1
24	6327	0.008	5.81	0.27	0.5	1.14	2	***0***	3.17	3.2	0	***0***	3.83	3.84	0
25	8947	**0**	4.12	0.033	0.42	0.66	1	**0**	2.09	2.12	0	**0**	2.12	2.16	0
26	4448	0.27	19.94	1.14	0.53	17.67	1	0.23	3.2	4.19	1	0.23	3.53	4.48	1
27	10921	0.01	18.69	0.15	0.88	1.72	1	0.06	10.14	14.98	1	0.028	11.78	18.41	1
28	11117	0.05	12.38	0.19	0.58	1.53	1	0.05	5.28	7.3	1	0.05	7.3	10.34	1
29	9832	0.11	12.83	0.33	0.66	2.97	1	0.11	3.94	4.42	1	0.106	4.09	4.56	1
30	10816	0.21	3600	1.72	2.63	13150	14	0.2	9.3	299	3	0.19	13.44	214.52	3
31	4466	0.18	148.8	0.77	0.87	11.47	2	0.19	5.48	8.59	1	0.18	7.67	11.76	1
32	9881	0.04	10.14	0.25	0.59	1.51	1	0.01	6.45	6.64	1	0.01	6.58	6.78	1
33	39463	0.11	3600	0.4	0.171	3600	–	0.09	6.23	80.81	1	0.09	7.31	75.51	1
34	4701	0.03	7.73	0.62	1.22	43.95	4	0.23	10.55	24.62	4	0.144	17.39	118.44	3
35	5456	0.02	2.56	0.07	1.26	2.13	1	**0**	2.31	2.36	0	**0**	2.58	2.62	0
36	16781	0.15	300.3	0.43	1.8	207.6	1	0.04	15.62	17.8	1	0.04	15.73	17.91	1
37	14668	0.05	203.5	0.38	1.51	69.33	3	0.13	16.87	64.94	3	0.097	19.92	44.97	3
38	47249	0.01	6.5	0.045	1.08	3.41	1	0.007	3.36	3.91	1	0.007	2.68	3.11	1
39	41007	0.006	60.8	0.087	0.703	11.41	2	0.01	4.42	8.64	2	0.009	4.53	10.31	2
40	61633	0.02	504.7	0.11	1.94	280.83	1	0.02	7.97	18.56	1	0.02	8.06	18.67	1
41	17246	**0**	1.44	0.01	0.22	0.34	1	**0**	0.64	0.67	0	**0**	0.59	0.65	0
42	7887	0.01	18.84	0.12	1.39	2.52	1	0.01	6.08	6.44	1	0.01	6.56	6.86	1
43	5114	0.03	21.67	0.47	0.7	29.31	1	0.06	26.72	27.33	1	0.039	29.2	29.98	1
44	36022	0.08	3600	0.25	0.75	3600	–	0.02	14.63	47.72	1	0.016	14.78	40.72	1
45	17676	0.01	7.91	0.018	0.81	1.47	1	**0**	3.98	4.08	0	**0**	4.09	4.17	0
46	48701	0.004	327	0.12	0.59	241.7	1	0.004	19.26	20.34	1	0.004	19.5	20.59	1
47	66230	0.015	434.8	0.08	1.26	3600	–	0.02	14.15	52.8	2	0.017	16.51	46.53	2
48	58964	0.01	25.4	0.056	0.91	3.56	1	0.01	21.65	23.04	1	0.01	24.87	26.11	1
49	79614	0.016	870.1	0.081	1.45	1401	1	0.015	15.36	29.26	1	0.015	16.76	41.09	1
50	5937	0.03	10.59	0.357	1.11	2.7	2	0.23	22.06	36.04	2	0.13	27.23	50.94	2
51	9060	0.088	3600	0.54	1.156	3600	–	0.16	23.04	141.7	1	0.128	25.53	52.87	1
52	34652	0.022	1042	0.11	1.81	202.5	1	0.022	8.95	39.14	1	0.022	8.48	38.83	1
53	30038	0.01	27.13	0.088	1.59	13.47	1	0.01	6.29	7.59	1	0.01	6.44	7.72	1
54	43583	**0**	8.78	0.028	0.8	1.73	1	**0**	9.03	9.12	0	**0**	9.42	9.53	0
55	69610	0.03	3600	0.077	1.81	3600	–	0.003	15.59	18.23	1	0.003	15.78	18.42	1
56	64474	**0**	1.8	0.004	0.89	1.16	1	**0**	7.72	7.81	0	**0**	8.03	8.12	0
57	49791	**0**	0.95	0.003	0.83	1.03	1	**0**	8.75	8.84	0	**0**	9.17	9.28	0
ave		0.15		0.74				0.13				0.08			

(In the table, Gap^{CS} in boldface indicates that for the corresponding instance its lower bound at the root node reaches the optimal value and Gap^{CS} in italics means that the optimality gap is smaller than that derived by column generation of Díaz et al. (2002).)

The problem size information of all test instances is given in Table 1.

In the implementation of Fenchel cutting planes generation algorithm, the big scalar M is set to be 10^6. The cutting plane loop without using partial integrality strategy will be terminated if the improvement of objective function value in four consecutive iterations is smaller than 0.1 or the maximum number of iterations 60 is reached. With the partial integrality strategy, the maximum number of iteration is set to 3 and the cutting plane loop will be terminated if the improvement of objective function value in two consecutive iterations is smaller than 1. In calculating the upper bound, the tabu-tenure values are set to $T_{min} = 8$ and $T_{max} = 16$; the stopping criterion is 750 iterations without improvement.

To demonstrate the effectiveness of the proposed cutting plane method in reducing integrality gap and the cut-and-solve approach in searching an optimal solution, we consider three versions of the cut-and-solve algorithm: *CS1*: the cutting plane method using LCIs only, together with the partial integrality strategy, where the iteration number of cutting plane loop is set to 1; *CS2*: an extended version of CS1 which also uses FCPs; *CS3*: CS2, whose maximum

Table 4
Numerical results on randomly generated instances.

P	$\|I\| \times \|J\|$	C_{TOT}/D_{TOT}	LB	Gap^{CS}	$t(s)^{LB}$	BUB_{CS3}	Dev^{CS}	T (s)	m	BUB_{Cplex}	Dev^{Cplex}	Cplex time (seconds)
1	30 × 200	1.37–2.06	30137.3	0.14	73.8	30181	0	396.5	2	30182	0.03	50000
2			28889.1	0.12	123.2	28923	0	571.9	2	28923	0.03	50000
3			28115.4	0.05	33.2	28131	0	48.33	1	28131	0	183.25
4			28119.4	0.12	43.1	28152	0	66.1	1	28152	0	2786.0
5			27634.6	0.04	17.45	27646	0	20.2	1	27646	0	14.72
6	60 × 200	2.77–3.50	27935.1	0.15	125.1	27977	0	798.2	1	27979	0.03	50000
7			29646.3	0.19	389.0	29704	0	44636.8	4	29709	0.12	50000
8			27977.0	0.06	458.6	27993	0	1524.0	3	27993	0	41475
9			27657.6	0.12	263.1	27691	0	1047.1	3	27691	0.04	50000
10			29150.2	0.15	528.7	29195	0	32978.2	4	29216	0.17	50000
11	60 × 300	3.03–6.06	35616.4	0.09	752.7	35648	0	2051.4	1	35802	0.74	50000
12			35449.3	0.07	93.5	35474	0	225.9	1	35474	0	3157
13			33859.0	0.04	1606.7	33872	0	1910.9	1	33912	0.90	50000
14			33084.1	0.04	749.2	33096	0	873.2	1	33101	0.17	50000
15			30912.5	0.02	1370.6	30918	0	1611.8	1	30919	0.71	50000
16	80 × 400	1.52–8.28	39299.1	0.05	1748.1	39318	0	3998.3	2	39324	0.19	50000
17			37069.1	0.02	1300.0	37076	0	1583.9	1	37076	0.26	50000
18			43843.2	0.04	3910.1	43859	0	15358.4	1	43865	0.09	50000
19			37334.2	0.03	2034.3	37344	0	5208.0	2	37344	0	18334
20			43483.8	0.06	3730.2	43508	0	8899.0	2	43745	0.78	50000

number of iterations of cutting plane loop is set to 4 and the loop is terminated if the improvement of objective function value in two consecutive iterations is smaller than 1. Note that all versions of CS use the partial integrality strategy.

Since it is difficult to compare the speeds of two computers (the number of processors, CPU speed or the RAM memory of one computer is not known), we compare the lower bounds found in the literature, the computational time of Cplex running on the same computer with that of our algorithm.

4.1. Computational results on Holmberg's benchmarks

Table 2 shows the computational results on Holmberg's benchmark instances. By comparing the column Gap^H and Gap^{CS}, we can find that the proposed partial integrality based cutting plane method outperforms the Lagrangian relaxation (Holmgerg et al., 1999) by deriving a tighter lower bound for the SSCFLP. The average relative optimality gap has been reduced from 0.41% to 0.12%, 0.053% and 0.024% by the three versions of the cut-and-solve algorithms, respectively. By comparing Gap^{CS} of CS1 with those of CS2 and CS3, we can find that the Fenchel cutting plane is much stronger than the lifted cover inequality in strengthening the lower bound.

In terms of computational time, the average computational time for Cplex to obtain an optimal solution for 71 instances is 12.27 s, whereas the CS1 uses only 8.31 s. The cut-and-solve is at least comparable with Cplex, which uses branch-and-cut with clique, cover and GUB cover cuts generation activated. By comparing $t(s)^{LB}$ of the three CS algorithms, we can find that the generation of Fenchel cutting planes requires more computational efforts than that of LCIs, which leads to a longer computational time for CS2 and CS3. However, for 42 of 71 instances, the lower bound generated by CS2 at the root node reaches the optimal value, and 47 instances for CS3 compared with 26 instances for CS1.

From Table 2, we can also observe that the tighter the lower bound, the smaller the number of branching nodes (column m) generated by the proposed CS algorithm. Especially for CS3, only 5 of 71 instances whose number of branching nodes is greater than 1. In Tables 2 and 3, $m = 0$ means that the lower bound solution is optimal or the lower bound is equal to the upper bound found by the tabu search algorithm. As a result, no branching is required in the CS approach. On the other hand, $m = 1$ can be explained as that the most "promising" facilities that are opened in an optimal solution have been detected at the root node by the partial integrality based cutting plane method. Only one branch is enough to prove the optimality of the solution of the sparse problem at the first level.

4.2. Computational results on Delmaire's benchmarks

In this subsection, we compare our computational results with those obtained by branch-and-price (Díaz and Fernández, 2002) and by Cplex. The numerical results are listed in Table 3. From the table, we can find that the lower bound obtained by our cutting plane method with partial integrality strategy (column Gap^{CS}) is tighter than that of column generation of Díaz and Fernández (2002) (column Gap^D). The average relative optimality gap between the lower bound obtained by column generation and the optimal value is 0.15%, whereas the gap of the proposed cutting plane method is 0.13% for CS2 and 0.08% for CS3. For 21 of 57 instances, the lower bound generated by column generation has been improved by CS2, and 25 instances by CS3.

The CS2 and CS3 outperform Cplex in computational time, where the computational time of Cplex is given by the 4th column of Table 3. There are four instances (D30, D33, D44, D51) that cannot be solved to optimality by Cplex within a time limit of 3600 s, whereas CS2 and CS3 both solve all instances exactly with a computational time less than 300 s.

4.3. Large sized instances

In this subsection, we extend our numerical test to randomly generalized large instances. They are classified into four subsets which differ in the problem size and the ratio C_{TOT}/D_{TOT} as given by the second and the third column of Table 4, respectively. Note that the interval from which the ratio $C_{TOT}/D_{TOT}$is uniformly generated is the same as the first 4 subsets of Holmberg's benchmarks (see Table 1).

Each subset contains five instances which are randomly generated in a way similar to that of Cornuéjols et al. (1991) for generating test instances of the CFLP and to that of Klose (2000) for the TSCFLP. The geographical locations of potential facilities and cus-

tomers are first generated uniformly on a unit square. The Euclidean distance between facility and customer is multiplied by 10 to define unit assignment cost. The customer demands are generated from $U[5,35]$, where $U[p,q]$ denotes the uniform distribution defined on the interval $[p,q]$. The capacities b_i are generated from $[10,160]$. The ratio C_{TOT}/D_{TOT} is then used to scale the capacities. Finally, the fixed opening cost of each facility i is defined by the formula $f_i = U[0,90] + U[100,110]\sqrt{b_i}$.

We compare our CS3 with Cplex in solving the instances. For the CS3, the maximum number of iterations of the cutting plane loop with partial integrality strategy is set to 20. Since solving the large-sized instances is quite time consuming, we set a time limit of 50,000 seconds for the resolution of each instance. Each algorithm of CS3 and Cplex will be terminated after the running of this amount of time. We compare the two algorithms in terms of computation time, the number of instances optimally solved, and the relative gap between the best upper bound and the best lower bound found within a time limit of 50,000 CPU seconds. The computational results are presented in Table 4.

From Table 4, we can find that the computational time is instance-sensitive for both CS3 and Cplex, but the computation time of CS3 is much shorter than that of Cplex. Compared with Cplex which only solves six instances to optimality, CS3 finds an optimal solution for all 20 instances within the time limit. For 14 of 20 instances, CS3 can find an optimal solution within 3600 seconds (one hour), whereas for the same number of instances, Cplex can not find an optimal solution after a running time of 50,000 seconds. For the unsolved instances, the relative gap between the best upper bound and the best lower bound found by Cplex is between 0.03% and 0.9%. From the above numerical analysis, we can say CS3 significantly outperforms Cplex in optimally solving the large sized instances of SSCFLP with much less computational time.

5. Conclusion and remarks

In this paper, a cutting plane method which generates lifted cover inequalities and the Fenchel cutting planes is proposed to separate the 0–1 knapsack polytope. Combined with the cutting plane method and the partial integrality strategy, a cut-and-solve (CS) based algorithm is developed to exactly solve the SSCFLP. Numerical tests on benchmark instances and randomly generated instances demonstrate that the effectiveness of the proposed cutting plane method in reducing the optimality gap and the CS in searching an optimal solution of the SSCFLP.

For future research, one possible direction is to consider more generalized valid cutting planes, which take several knapsack constraints into account at the same time. Such cutting planes may further strengthen the original model of the SSCFLP so that its large instances can be more quickly solved.

Acknowledgements

The authors are grateful to the referees for their valuable and insightful comments and to Prof. Elena Fernández for providing some of the tested instances.

References

Agar, M., Salhi, S., 1998. Lagrangean heuristics applied to a variety of large capacitated plant location problems. Journal of the Operational Research Society 49, 1072–1084.

Ahuja, R.K., Orlin, J.B., Pallottino, S., Scaparra, M.P., Scutellà., 2004. A multi-exchange heuristic for the single-source capacitated facility location problem. Management Science 50 (6), 749–760.

Balas, E., 1975. Facets of the knapsack polytope. Mathematical Programming 8, 146–164.

Balas, E., Zemel, E., 1978. Facets of the knapsack polytope from minimal covers. SIAM Journal on Applied Mathematics 34, 119–148.

Barcelo, J., Casanovas, J., 1984. A heuristic Lagrangian algorithm for the capacitated plant location problem. European Journal of Operational Research 15, 212–226.

Beasley, J.E., 1993. Lagrangian heuristics for location problems. European Journal of Operational Research 65, 383–399.

Boccia, M., Sforza, A., Sterle, C., Vasilyev, I., 2008. A cut and branch approach for the capacitated p-median problem based on Fenchel cutting planes. Journal of Mathematical Modeling and Algorithms 7, 43–58.

Boyd, E., 1993a. Generating Fenchel cutting planes for knapsack polyhedra. SIAM Journal on Optimization 3, 734–750.

Boyd, E. 1993b. Solving Integer Programs with Cutting Planes and Preprocessing. In: IPCO, pp. 209–220.

Boyd, E., 1994. Fenchel cutting planes for integer programs. Operation Research 42, 53–64.

Ceselli, A., Righini, G., 2005. A branch and price algorithm for the capacitated p-median problem. Networks 45 (3), 125–142.

Chen, C.H., Ting, C.J., 2008. Combining lagrangian heuristic and ant colony system to solve the single source capacitated facility location problem. Transportation Research Part E 44, 1099–1122.

Climer, S., Zhang, WX., 2006. Cut-and-solve: an iterative search strategy for combinatorial optimization problems. Artificial Intelligence 170, 714–738.

Contreras, I.A., Díaz, J.A., 2008. Scatter search for the single source capacitated facility location problem. Annals of Operations Research 157, 73–89.

Cornuéjols, G., Sridharan, R., Thizy, J.-M., 1991. A comparison of heuristics and relaxations for the capacitated plant location problem. European Journal of Operational Research 50, 280–297.

Cortinhal, M.J., Captivo, M.E., 2003. Upper and lower bounds for the single source capacitated location problem. European Journal of Operational Research 151, 333–351.

CPLEX, 2009. ILOG CPLEX 12.1. Reference Manual.

Crowder, H., Johnson, E., Padberg, M., 1983. Solving large-scale 0–1 linear programming programs. Operation Research 31, 803–834.

Delmaire, H., Díaz, J., Fernández, E., Ortega, M., 1999. Reactive grasp and tabu search based heuristics for the single source capacitated plant location problem. Information Systems and Operational Research 37 (3), 194–225.

Díaz, J.A., Fernández, E., 2002. A branch-and-price algorithm for the single source capacitated plant location problem. Journal of the Operational Research Society 53, 728–740.

Ferreira, C.E., Martin, A., Weismantel, R., 1996. Solving multiple knapsack problems by cutting planes. SIAM Journal on Optimization 6, 858–877.

Gu, Z., Nemhauser, G.L., Savelsbergh, M.W.P., 1998. Cover inequalities for 0–1 linear programs: computation. INFORMS Journal on Computing 10, 427–437.

Gu, Z., Nemhauser, G.L., Savelsbergh, M.W.P., 1999. inequalities for 0–1 linear programs: complexity. INFORMS Journal on Computing 11, 117–123.

Gu, Z., Nemhauser, G.L., Savelsbergh, M.W.P., 2000. Sequence independent lifting in mixed integer programming. Journal of Combinatorial Optimization 4, 109–129.

Holmberg, K., Ronnqvist, M., Yuan, D., 1999. An exact algorithm for the capacitated facility location problems with single sourcing. European Journal of Operational Research 113, 544–559.

Kaparis, K., Letchford, A.N., 2010. Separation algorithms for 0–1 knapsack polytopes. Optimization Online, Mathematical Programming 124 (1–2), 69–91.

Klabjan, D., Nemhauser, G.L., Tovey, C., 1998. The complexity of cover inequality separation. Operations Research Letters 23, 35–40.

Klincewica, J., Luss, H., 1986. A lagrangian relaxation heuristic for capacitated facility location with single-source constraints. Journal of the Operational Research Society 37, 495–500.

Klose, A., 2000. A Lagrangean relax-and-cut approach for the two-stage capacitated facility location problem. European Journal of Operational Research 126, 408–421.

Neebe, A., Rao, M., 1983. An algorithm for the fixed-charge assigning users to sources problem. Journal of the Operational Research Society 34, 1107–1113.

Pirkul, H., 1987. Efficient algorithm for the capacitated concentrator location problem. Computers and Operations Research 14, 197–208.

Pisinger, D., 1995. An expanding-core algorithm for the exact 0–1 knapsack problem. European Journal of Operational Research 87, 175–187.

Ramos, M.T., Sáez, J., 2005. Solving capacitated facility location problems by Fenchel cutting planes. Journal of the Operational Research Society 56, 297–306.

Rönnqvist, M., Tragantalerngsak, S., Holt, J., 1999. A repeated matching heuristic for the single-source capacitated facility location problem. European Journal of Operational Research 116, 51–68.

Sridharan, R., 1993. A lagrangian heuristic for the capacitated plant location problem with single source constraints. European Journal of Operational Research 66, 305–312.

Weismantel, R., 1997. On the 0–1 knapsack polytope. Mathematical Programming 77, 49–68.