Combining Capacity Scaling and Local Branch Approaches for the Logistics Network Design Problem

N. Katayama, S. Yurimoto

Distribution and Logistics Systems, Ryutsu Keizai University, 120 Hirahata, Ryugasaki, Ibaraki, Japan

Abstract
Logistics network design is an important issue, which appears in supply chain management and distribution network design. The solution of the problem provides the network design and routes of freight. This problem is known as an NP-hard problem. Recently, several hybrid methods for a capacitated problem have been developed, such as combining exact and heuristics approaches and a MIP-tabu search hybrid. This paper presents hybrid heuristics combining capacity scaling and local branching heuristics. A capacity scaling method is based on changing capacities depending on these residual capacities at the LP problem, and a local branching approach utilizes a general MIP solver to explore limited neighborhoods. The combining method is able to find the best or equivalent solutions for 97% of the benchmark problems, and new best solutions for 70% of the problems.

Keywords: Network Design, Logistics, Optimization

1 INTRODUCTION

The logistics network design problem represents a generic network model for applications in designing the construction and improvement of logistics, transportation, distribution and production in supply chain networks. For logistics network design problems, a wide range of application models can be found in Magnanti and Wong [1]. The logistics network design problem with limited capacities is called the capacitated network design problem (CND). The solution of CND provides the appropriate network design and routes of multicommodity flows, to minimize the total cost that is the sum of flow costs and design costs over the network with limited arc capacities. CND is formulated as a mixed integer programming problem. Binary variables are used to model the network design, selecting arcs from a candidate arc set appropriately, while continuous variables represent the volumes of path flows on the network. CND is known as an NP-hard problem.

Many approaches have been developed, such as valid inequalities, relaxation methods and heuristics. Holmberg and Yuan [2] proposed a combination algorithm of a Lagrangian relaxation method and a branch-and-bound algorithm. Chouman et al. [3] proposed cover inequalities, minimum cardinality inequalities and these lifting inequalities, as well as cut-set inequalities. Costa et al. [4] proposed Benders' decomposition, metric and cutset inequalities.

Many tabu search meta-heuristics designed to find feasible solutions within a reasonable computation time have been developed. Crainic et al. [5] proposed simplex-based tabu search methods. Ghamlouche et al. [6] [7] proposed a cycle-based tabu search method. Ghamlouche et al. [8] and Alvarez et al. [9] proposed path relinking algorithms or scatter search algorithms. Crainic and Gendron [10] and Crainic et al. [11] proposed cooperative parallel tabu searches.

The scaling heuristics is based on changing flow costs or capacities, which depend on arc flow volumes, residual capacities or dual variable information, and solving linear relaxation problems iteratively. Crainic et al. [12] proposed slope scaling heuristics for design costs, and Katayama et al. [13] proposed capacity scaling heuristics for arc capacities.

Recently, combining or hybrid methods with meta-heuristics and MIP solvers have been developed. Rodríguez-Martín and Salazar-González [14] proposed a local branching heuristics, which utilizes a general MIP solver to explore neighborhoods. Hewitt at al. [15] proposed combining exact and heuristics approaches to search very large neighborhoods by solving restricted problems. Chouman and Crainic [16] proposed a MIP-tabu search hybrid framework, combining an exact MIP method and tabu search method.

This paper presents a combining method with a capacity scaling heuristics and a local branch heuristics. The capacity scaling heuristics is an approximate iterative solution method, based on changing capacities. The proposed capacity scaling heuristics uses a path-based formulation, including tight forcing constrains and column-row generation technique for CND. Since the formulation with forcing constraints is a large mixed integer programming problem (MIP), and it takes significant amounts of time to solve MIP or its linear relaxation problems, a weak formulation without tight forcing constraints is used in many papers to the present. A column generation and row generation technique is able to reduce a problem size. Consequently, even if a path-based formulation includes tight forcing constraints, its linear relaxation problem can be solved within an appropriate computation time using a column-row generation technique. Then the capacity scaling heuristics is applied to the formulation and the feasible solutions can be obtained.

These feasible solutions may not be good solutions necessarily. Then local branch heuristics is applied to the feasible solutions, and the better solutions can be found. The local branch heuristics solve MIP with restricted neighborhoods derived from the feasible solutions. The MIP restricted neighborhoods can be solved by a MIP solver.

2 MATHEMATICAL FORMULATION

CND can be described as follows. $G=(N,A)$ denotes a directed network with the set of nodes N and the set of directed arcs A. Let K be the set of commodities on the network. For each commodity $k \in K$, let P^k be the set of paths of commodity k, and d^k the required amount of flow of commodity k from its single origin node to its single destination node.

The following measures characterize arc $(i,j) \in A$: f_{ij} the design cost of including arc (i,j) in the network design, c_{ij}^k the unit variable flow cost for commodity k flowing on arc (i,j), and C_{ij} the limited arc capacity which must be shared by all the commodities flowing on the arc.

The formulation of CND has two type variables. The first type is a binary design variable, which is defined as $y_{ij} = 1$, if arc (i, j) is included in the network design, $y_{ij} = 0$ otherwise. The second type is a continuous flow variable, which is defined by x_p^k representing the amount of the path flow of commodity k flowing on the path $p \in P^k$. Let δ_{ij}^p be the constant, $\delta_{ij}^p = 1$ if arc (i, j) is included in path p, $\delta_{ij}^p = 0$ otherwise.

The path flow based formulation CNDP for CND can be formulated as follows:

(CNDP)

$$minimize \sum_{(i,j)\in A}\sum_{k\in K} c_{ij}^k \sum_{p\in P^k} \delta_{ij}^p x_p^k + \sum_{(i,j)\in A} f_{ij} y_{ij} \quad (1)$$

$$subject\ to \sum_{p\in P^k} x_p^k = d^k \quad \forall k \in K \quad (2)$$

$$\sum_{k\in K}\sum_{p\in P^k} \delta_{ij}^p x_p^k \le C_{ij} y_{ij} \quad \forall (i,j) \in A \quad (3)$$

$$\sum_{p\in P^k} \delta_{ij}^p x_p^k \le d^k y_{ij} \quad \forall (i,j) \in A, k \in K \quad (4)$$

$$x_p^k \ge 0 \quad \forall p \in P^k, k \in K \quad (5)$$

$$y_{ij} \in \{0,1\} \quad \forall (i,j) \in A \quad (6)$$

The objective function (1) is the total cost, the sum of variable flow costs of commodities plus the sum of design costs in a given network design, and should be minimized. Constraints (2) are the flow conservation equations, representing the fact that the sum of path flows of commodity k is equal to the required amount. Constraints (3) provide the capacity constraints, which prohibit flowing if the arc is excluded, $y_{ij} = 0$, and allowing for flow up to the arc capacity if the arc is included, $y_{ij} = 1$. Constraints (4) are the forcing constraints, which prohibit flowing of commodity k if the arc is excluded, and allow for flow up to the required amount if the arc is included. Constraints (5) ensure the non-negativity of continuous variables, and constraints (6) force binary variables to assume binary values.

Let x_{ij}^k be the arc flow variable for commodity k flowing on arc (i, j). Let N_n^+ be the set of outward nodes from node n, and N_n^- be the set of inward nodes. The arc flow based formulation CNDA of CND can be formulated as follows

(CNDA)

$$minimize \sum_{(i,j)\in A}\sum_{k\in K} c_{ij}^k x_{ij}^k + \sum_{(i,j)\in A} f_{ij} y_{ij} \quad (7)$$

$$subject\ to$$

$$\sum_{i\in N_n^+} x_{in}^k - \sum_{j\in N_n^-} x_{nj}^k = \begin{cases} -d^k \ if\ n = O^k \\ d^k \ \ if\ n = D^k \\ 0 \ \ otherwise \end{cases} \quad \forall n \in N, k \in K \quad (8)$$

$$\sum_{k\in K} x_{ij}^k \le C_{ij} y_{ij} \quad \forall (i,j) \in A \quad (9)$$

$$x_{ij}^k \le d^k y_{ij} \quad \forall (i,j) \in A, k \in K \quad (10)$$

$$x_{ij}^k \ge 0 \quad \forall p \in P^k, k \in K \quad (11)$$

$$y_{ij} \in \{0,1\} \quad \forall (i,j) \in A \quad (12)$$

Constraints (8) are the flow conservation equations representing relations between the sum of arc flows into node n and the sum of arc flows out from node n for commodity k.

3 CAPACITY SCALING HEURISTICS

Capacity scaling heuristics is an approximate iterative solution method for capacitated network problems based on changing arc capacities, which depend on current flow volumes, residual capacities or design variables [13]. When solving the linear relaxation problem of CNDP, most design variables have decimal fraction at a relaxation solution. As a consequence, a feasible solution derived from the relaxation solution may not be good approximation for CNDP.

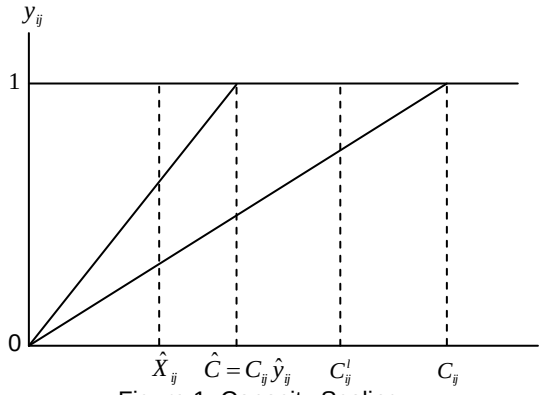

Figure 1: Capacity Scaling

Given the optimal arc flow $\hat{\boldsymbol{X}}$, let $\hat{\boldsymbol{y}}$ be the minimum continuous values which satisfy constrains (9) and (10). When the capacity $\boldsymbol{C}$ is changed to $\boldsymbol{C}\hat{\boldsymbol{y}}$ at CNDP, the optimal objective value of CNDP with capacity $\hat{\boldsymbol{C}} = \boldsymbol{C}\hat{\boldsymbol{y}}$ does not change. Then the linear relaxation problem with capacity $\boldsymbol{C}$ is solved. As a result, a 0 or 1 solution for each design variable can be obtained. The multicommodity flow problem of all fixed design variables to these binary solutions is solved, and then the optimal flow variables of CNDP can be obtained. Figure 1 shows the relations among the capacities and the flow.

As a matter of course, finding the optimal flow of CNDP is extremely difficult. If a near optimal flow can be found, $\hat{\boldsymbol{C}}$ can be estimated and a good approximate solution might be derived from it. On the other hand, by changing capacity $\boldsymbol{C}$ a little bit at a time, a near optimal flow should be sought.

Capacity scaling heuristics begins by solving the linear relaxation problem of CNDP with $\boldsymbol{C}^l$ instead of $\boldsymbol{C}$ at iteration l. Initially, $\boldsymbol{C}^1 := \boldsymbol{C}$. The linear relaxation problem LR($\boldsymbol{C}^l$) with capacity $\boldsymbol{C}^l$ at the iteration l can be formulated as follows:

LR($\boldsymbol{C}^l$)

$$minimize \sum_{(i,j)\in A}\sum_{k\in K} c_{ij}^k \sum_{p\in P^k} \delta_{ij}^p x_p^k + \sum_{(i,j)\in A} f_{ij} y_{ij} \quad (13)$$

$$subject\ to \sum_{p\in P^k} x_p^k = d^k \quad \forall k \in K \quad (14)$$

$$\sum_{k\in K}\sum_{p\in P^k} \delta_{ij}^p x_p^k \le C_{ij}^l y_{ij} \quad \forall (i,j) \in A \quad (15)$$

$$\sum_{p\in P^k} \delta_{ij}^p x_p^k \le d^k y_{ij} \quad \forall (i,j) \in A, k \in K \quad (16)$$

$$x_p^k \ge 0 \quad \forall p \in P^k, k \in K \quad (17)$$

$$0 \le y_{ij} \le C_{ij} / C_{ij}^l \quad \forall (i,j) \in A \quad (18)$$

The right hand side of constraints (15) is changed to $C_{ij}^l y_{ij}$, and the right hand side of constraints (18) is changed to C_{ij}^l / C_{ij} to enable flow up to its original capacity C_{ij}. LR($\boldsymbol{C}^l$) can be solved by a MIP solver.

Let $\tilde{\boldsymbol{y}}$ be the optimal design variable of LR($\boldsymbol{C}^l$). At the next iteration, $\boldsymbol{C}^l$ is substituted by $\lambda \boldsymbol{C}^l \tilde{\boldsymbol{y}} + (1-\lambda)\boldsymbol{C}^l$, where $\lambda (0 \le \lambda \le 1)$ is a smoothing parameter preventing rapid jumping. If all design variables converge to zero or one in the solution of LR($\boldsymbol{C}^l$) at some iteration, then the multicommodity network flow problem of all fixed design variables $\tilde{\boldsymbol{y}}$ is solved and a feasible flow of CNDP is found.

For obtaining converged solutions, it may require numerous iterations, or sometimes they may not be convergent. Consequently, when most design variables converge to zero or one, and the number of free design variables is less than a certain number B, then a branch and bound algorithm is applied for free variables, and an upper bound and a feasible solution of CNDP are found. The capacity scaling heuristics stops when the iteration number exceeds the minimum iteration number $MINN$ and the upper bound UB is found.

An outline of the capacity scaling heuristics proceeds as follows:

Capacity Scaling Heuristics

1) Set $\lambda \in (0,1)$, $\varepsilon \in (0,0.5)$, $MINN$ and B.
$C_{ij}^1 = C_{ij}, (i,j) \in A$, $UB := \infty$, $l := 1$.

2) Solve LR($\boldsymbol{C}^l$). Let $\tilde{\boldsymbol{y}}$ be the corresponding design solution.

3) For each $(i,j) \in A$,

$$\bar{y}_{ij} := \begin{cases} 0 \ \ if \ \ \tilde{y}_{ij} < \varepsilon \\ 1 \ \ if \ \ \tilde{y}_{ij} > 1-\varepsilon \\ free \ \ otherwise \end{cases} \tag{19}$$

When the number of free variables of $\bar{\boldsymbol{y}}$ is less than B,

a) Solve the restricted problem with design variable $\bar{\boldsymbol{y}}$ such that $Z < UB$, by a branch and bound algorithm, where Z is the optimal value of the restricted problem

b) If Z can be found, then $UB^l := Z$.

4) If $l > MINN$ and $UB \neq \infty$, then stop the procedure.

5) $l := l+1$ and $C_{ij}^l := \lambda C_{ij}^{l-1} y_{ij} + (1-\lambda) C_{ij}^{l-1}, (i,j) \in A$.

B is reduced. Go to step 2.

4 COLUMN AND ROW GENERATION TECHNIQUE

In capacity scaling heuristics, the linear programming problem LR($\boldsymbol{C}^l$) must be solved iteratively. Since LR($\boldsymbol{C}^l$) has an exponentially large number of path flow variables and has the forcing constraints of the number of $O(|K||A|)$, not every variable and constraint can be included in the model when solving large instances. In order to solve larger instances efficiently, a column generation technique for path flow variables is used. The row generation can also reduce the number of forcing constraints, which are only generated, as needed, via a column generation.

For each commodity k, let $\bar{P}^k \subseteq P^k$ be the initial set of paths and Δ_{ij}^p the constant, $\Delta_{ij}^p = 1$ if path flow variable $x_p^k, p \in \bar{P}^k$ exists in the formulation and (i,j) is included in path p, $\Delta_{ij}^p = 0$ otherwise. Consequently, the only forcing constraint in the formulation is $\sum_{p \in P^k} \Delta_{ij}^p > 0$.

The restricted problem RLR($\boldsymbol{C}^l, \bar{P}$) with capacity $\boldsymbol{C}^l$, which is restricted path sets $\bar{P}^k \subseteq P^k, k \in K$ and restricted forcing constraints for LR($\boldsymbol{C}^l$), is reformulated as follows:

RLR($\boldsymbol{C}^l, \bar{P}$)

$$minimize \sum_{(i,j) \in A} \sum_{k \in K} c_{ij}^k \sum_{p \in \bar{P}^k} \delta_{ij}^p x_p^k + \sum_{(i,j) \in A} f_{ij} y_{ij} \tag{20}$$

$$subject \ to \sum_{p \in \bar{P}^k} x_p^k = d^k \quad \forall k \in K \tag{21}$$

$$\sum_{k \in K} \sum_{p \in \bar{P}^k} \delta_{ij}^p x_p^k \le C_{ij}^l y_{ij} \quad \forall (i,j) \in A \tag{22}$$

$$\sum_{p \in \bar{P}^k} \delta_{ij}^p x_p^k \le d^k y_{ij} \quad \forall (i,j) \in A, k \in K, if \ \ \Delta_{ij}^p > 0 \tag{23}$$

$$x_p^k \ge 0 \quad \forall p \in \bar{P}^k, k \in K \tag{24}$$

$$0 \le y_{ij} \le C_{ij} / C_{ij}^l \quad \forall (i,j) \in A \tag{25}$$

Let $\boldsymbol{s}$ be the dual variable for constraint (21), $\boldsymbol{u}(\ge \boldsymbol{0})$ for constraint (22), $\boldsymbol{w}(\ge \boldsymbol{0})$ for constraint (24). When solving RLR($\boldsymbol{C}^l, \bar{P}$) optimally, a dual solution $(\boldsymbol{s}, \boldsymbol{u}, \boldsymbol{w})$ is obtained. The reduced cost of path flow variable x_p^k is

$$\sum_{(i,j) \in A} (c_{ij}^k + u_{ij} + w_{ij}^k) \delta_{ij}^p - s^k. \tag{26}$$

The pricing problem is used for generating new path flow variables. The pricing problem of RLR($\boldsymbol{C}^l, \bar{P}$) is disjoint for each commodity k, and then can be solved separately. The pricing problem for commodity k is written as follows:

$$z^k = minimize \sum_{(i,j) \in A} (c_{ij}^k + u_{ij} + w_{ij}^k) \delta_{ij}^p x_p^k \tag{27}$$

$$subject \ to \sum_{p \in P^k} x_p^k = d^k \tag{28}$$

$$x_p^k \ge 0 \quad \forall p \in P^k, k \in K \tag{29}$$

Given $\boldsymbol{u}$ and $\boldsymbol{w}$, this is a shortest path problem with nonnegative arc length $c_{ij}^k + u_{ij} + w_{ij}^k$, and can be solved efficiently using Dijkstra's algorithm. Let $\hat{p}$ be the optimal path of this problem. If $z^k < s^k$, then the path flow variable $x_{\hat{p}}^k$ corresponding to the optimal path $\hat{p}$ has negative reduced cost. Then path $\hat{p}$ is added to $\bar{P}^k$, the new variable $x_{\hat{p}}^k$ is generated as a new column, and $\Delta_{ij}^p := 1, (i,j) \in \hat{p}$. When adding the new path $\hat{p}$, if $\sum_{p \in P^k} \Delta_{ij}^p = 0, (i,j) \in \hat{p}$ and the forcing constraints do not exist, then they are also generated and added to RLR($\boldsymbol{C}^l, \bar{P}$) as new rows.

To summarize, the algorithm with the column-row generation technique solving LR($\boldsymbol{C}^l$) is as follows:

Column-Row Generation Technique

1) For each $k \in K$, set $\bar{P}^k$ and $\Delta_{ij}^p = 1, (i,j) \in A, p \in \bar{P}^k$.

2) Solve RLR($\boldsymbol{C}^l, \bar{P}$). Let $(\boldsymbol{s}, \boldsymbol{u}, \boldsymbol{w})$ be a corresponding dual solution.

3) For each $k \in K$,

a) Solve the shortest path problem with the arc length $c_{ij}^k + u_{ij} + w_{ij}^k, (i,j) \in A$. Let z^k be the length of shortest path $\hat{p}$.

b) If $z^k < s^k$, then path $\hat{p}$ is added to $\bar{P}^k$, generate path variable $x_{\hat{p}}^k$ and $\Delta_{ij}^p = 1, (i,j) \in \hat{p}$.

c) For each $(i,j) \in A$, if $\sum_{p \in P^k} \Delta_{ij}^p > 0$ becomes greater than 0 from 0 in step b, then corresponding forcing constraints are generated and added to RLR($\boldsymbol{C}^l, \bar{P}$).

4) If a new path is generated, then go to step 2, otherwise stop the procedure.

5 LOCAL BRANCHING HEURISTICS

The feasible solution found by the capacity scaling heuristics is not necessarily a good solution. Then local branch heuristics [13] is applied to the feasible solutions, and better solutions may be found.

Given the feasible design solution $\bar{\boldsymbol{y}}$ from the capacity scaling heuristics and a parameter $m(>0)$, the additional local branching constraint is as follows:

$$\sum_{(i,j) \in A | \bar{y}_{ij}=1} (1 - y_{ij}) + \sum_{(i,j) \in A | \bar{y}_{ij}=0} y_{ij} \le m \tag{30}$$

$$\sum_{(i,j) \in A | \bar{y}_{ij}=1} (1 - y_{ij}) + \sum_{(i,j) \in A | \bar{y}_{ij}=0} y_{ij} \ge 1 \tag{31}$$

The first branching constraint includes the domain of $m-OPT$ neighborhood of $\bar{\boldsymbol{y}}$. The second branching constraint excludes the current solution.

This method starts with a feasible design solution $\bar{\boldsymbol{y}}$ of CNDA. The reason for using CNDA instead of CNDP is that the solutions of CNDP with restricted path variables and forcing constraints derived from the column-row generation may not include the optimal solution. The branching constraint (30) associated with $\bar{\boldsymbol{y}}$ is added to

Table 1: Average Gaps for C Instances (%)

RL	LBR	IP	MIP	CAP	5-300	5-900	10-300	10-900	15-300	15-900	BEST
5.01	3.55	2.13	1.31	1.65	1.12	1.06	0.92	0.88	0.98	0.84	0.79

Table 2: Detail Results for C Instances

N/A/K/Type	OPT/LB		RL	LBR	IP	MIP	CAP	CALB		GAP(%)
100/400/010/F/L	23949	O	24022	24690	*23949*	24161	24459	*23949*	*	0.00
100/400/010/F/T	62245	L	65278	67357	65885	67233	72169	*64607*	**	3.79
100/400/010/V/L	28423	O	28485	*28423*	*28423*	*28423*	*28423*	*28423*	*	0.00
100/400/030/F/L	49018	O	51325	49872	49694	49682	51956	*49018*	**	0.00
100/400/030/F/T	130852	L	141359	141633	141365	144349	144314	1*36446*	**	4.28
100/400/030/V/T	384802	O	384926	384809	384836	384940	384880	*384802*	**	0.00
20/230/040/V/L	423848	O	424385	*423848*	424385	*423848*	*423848*	*423848*	*	0.00
20/230/040/V/T	371475	O	371811	*371475*	371779	*371475*	371906	*371475*	*	0.00
20/230/040/F/T	643036	O	645548	*643036*	643187	643538	643666	*643036*	*	0.00
20/230/200/V/L	94213	O	100404	95295	95097	94218	*94213*	*94213*	**	0.00
20/230/200/F/L	136975	L	147988	143446	141253	138491	137851	*137642*	**	0.49
20/230/200/V/T	97914	O	104689	98039	99410	98612	97968	*97914*	**	0.00
20/230/200/F/T	134811	L	147554	141128	140273	136309	136302	*136031*	**	0.90
20/300/040/V/L	429398	O	*429398*	*429398*	*429398*	*429398*	*429398*	*429398*	*	0.00
20/300/040/F/L	586077	O	590427	*586077*	*586077*	588464	587800	*586077*	*	0.00
20/300/040/V/T	464509	O	*464509*	*464509*	*464509*	*464509*	464569	*464509*	*	0.00
20/300/040/F/T	604198	O	609990	*604198*	*604198*	*604198*	*604198*	*604198*	*	0.00
20/300/200/V/L	74375	L	78184	76375	75319	75045	74913	*74830*	**	0.61
20/300/200/F/L	113144	L	123484	119143	117543	116259	115876	*115751*	**	2.30
20/300/200/V/T	74991	O	78867	76168	76198	74995	*74991*	*74991*	**	0.00
20/300/200/F/T	105846	L	113584	109808	110344	109164	*107467*	*107467*	**	1.53
30/520/100/V/L	53958	O	54904	54026	54113	54008	54012	*53958*	**	0.00
30/520/100/F/L	92653	L	102054	96255	*94388*	93967	94743	94043		1.50
30/520/100/V/T	52046	O	53017	52129	52174	52156	52270	*52046*	**	0.00
30/520/100/F/T	95852	L	106130	101102	98883	97490	98867	*97377*	**	1.59
30/520/400/V/L	112422	L	119416	114367	114042	112927	112846	*112786*	**	0.32
30/520/400/F/L	147183	L	163112	157726	154218	149920	*149446*	*149446*	**	1.54
30/520/400/V/T	114556	L	120170	115240	114922	114664	*114641*	*114641*	**	0.07
30/520/400/F/T	150215	L	163675	168561	154606	152929	*152745*	*152745*	**	1.68
30/700/100/V/L	47603	O	48723	*47603*	47612	*47603*	47614	*47603*	*	0.00
30/700/100/F/L	59321	L	63091	60272	60700	60184	60192	*59995*	**	1.14
30/700/100/V/T	45822	L	47209	45905	46046	45880	46169	*45875*	**	0.12
30/700/100/F/T	54597	L	56576	55104	55609	54926	55339	*54904*	**	0.56
30/700/400/V/L	97001	L	105116	103787	98718	97982	*97960*	*97960*	**	0.99
30/700/400/F/L	131233	L	145026	169760	152576	135109	*135100*	*135100*	**	2.95
30/700/400/V/T	94333	L	101212	96680	96168	95781	*95306*	*95306*	**	1.03
30/700/400/F/T	128027	L	141013	144926	131629	130856	*130146*	*130146*	**	1.66

O :Optimal Value, L :Lower Bound, Italic type:Best Upper Bound, ** : New Best, * : Equivalent Best

CNDA, and the added problem is solved by a MIP solver. If a better solution $\bar{y}'$ is found within a time limit T, then it becomes the new incumbent as $\bar{y} := \bar{y}'$. At this incumbent, the problem that the following constraint is replaced by (30) is solved iteratively.

$$\sum_{(i,j)\in A|\bar{y}_{ij}=1}(1-y_{ij})+\sum_{(i,j)\in A|\bar{y}_{ij}=0}y_{ij}\geq m+1 \qquad (32)$$

If the better solution cannot be found, the size of neighborhood m is reduced.

6 COMPUTATIONAL EXPERIMENTS

To evaluate the performance of the combining capacity scaling and local branching heuristics, the optimal value or a lower bound using a branch and bound algorithm is also compared to the result of the path relinking [8], the local

Table 3: Average CPU Times for C Instances (seconds)

RL	LBR	IP	MIP	CAP	5-300	5-900	10-300	10-900	15-300	15-900
9432.3	574.6	408.1	6958.6	76.3	503.5	1550.4	644.4	1495.0	514.7	1390.5

Table 4: Average Gaps for C Instances (%)

SIMP	PARA	CYCL	CAP	CALB
9.52	6.03	3.64	0.65	0.19

Table 5: Gap Distribution According to Fixed Cost and Capacity Level, R Instances (%)

	RL			CALB		
	C1	C2	C8	C1	C2	C8
F01	0.76	0.78	1.15	0.00	-0.01	-0.12
F02	2.43	2.64	3.23	-0.14	-0.17	-0.43
F10	3.09	3.04	4.11	-0.58	-0.61	-0.62

Table 6: Gap Distribution According to Problem Dimensions, R Instances (%)

N/A/K	RL	CALB	N/A/K	RL	CALB
10/25/10	0.00	0.00	20/100/40	1.37	0.00
10/25/25	0.23	0.00	20/100/100	2.05	-0.04
10/25/50	0.61	0.00	20/100/200	4.55	-0.12
10/50/10	0.08	0.00	20/200/40	3.59	-0.06
10/50/25	0.36	0.00	20/200/100	4.93	-0.66
10/50/50	1.14	0.00	20/200/100	5.41	-0.75
10/75/10	0.04	0.00	20/300/40	2.08	-0.33
10/75/25	0.41	0.00	20/300/100	4.68	-0.79
10/75/50	1.52	0.00	20/300/200	6.84	-2.24

Table 7: Average CPU Times for R instances (seconds)

SIMP	PARA	CYCL	CAP	5-300	5-900	10-300	10-900	15-300	15-900
785.4	989.7	1575.2	45.7	539.5	288.8	532.6	211.0	493.6	1390.5

branching [14], IP search [15], MIP tabu search [16], simplex-based tabu search [5], cooperative parallel tabu search [10] and cycle-based tabu search [7]. The same two data sets by Crainic et al. [6] are used. The detailed description of these problem instances is given in [6].

The first set of instances, denoted C, consists of 37 instances characterized by the number of nodes, the number of arcs and the number of commodities. Original C instances consist of 43 problems. But because 6 problems are trivial, their problems are excluded. Two letters are used to characterize the design cost level, "F" for high and "V" for low relative to the flow cost, and the capacity level "T" for tight and "L" for loose compared to the total demand.

The second set of instances, denoted R, consists of 153 problem instances characterized by three design cost levels, "F01", "F05", "F10", and three capacity levels, "C1", "C2", "C8". If the F value is small, the design costs are low, and if the F value is large, the design costs are high compared to the flow costs. If the C value is small, the arc capacities are loose, and if large, the arc capacities are tight compared to the total demand.

The experiments were performed on PCs with Pentium i7 CPU 2.93GHz with 4 cores, 16GBytes RAM. The computer code is written in C++ compiled for Ubuntu 10.0. CPLEX 12.2, a MIP solver by ILOG, is used to solve linear programming problems and mixed integer programming problems. In order to assess the solution quality relative to the optimal values or lower bounds, all instances using the branch and bound algorithm of CPLEX are solved, and a limit of 10 hours of computation time was imposed for each instance. If the problem cannot be solved optimally within the limit computation time, the lower bound found in the branch and bound algorithm is used instead of the optimal value.

A smoothing parameter λ was calibrated, and ten values from 0.025 to 0.250 were tested. An execution parameter of a branch and bound algorithm, B =75 for each instance initially. A branching parameter m set to 5, 10 and 15, and the time limit T of a branch and bound algorithm set to 300 and 900 seconds.

Table 1 displays the average gaps of results for C instances. The average gaps are relative to the optimal value/lower bound by CPLEX for the upper bound by each heuristics. RL is the result by the path relinking, LBR by local branching, IP by IP search, and MIP by MIP Tabu search. CAP is the best result found among all parameters by the capacity scaling heuristics, and BEST by the combining method with the capacity scaling and local branching heuristics. Columns from 5-300 to 15-900 are the results of the combining method. The first letter indicates the branching parameter m , and the second letter indicates the time limit T . Tables 2 display the detailed results for C instances. Column N/A/K/Type indicates the number of nodes, arcs, commodities, and the design cost level and capacity level. Column OPT/LB corresponds to the optimal value/lower bound by CPLEX. "O" indicates that the optimal value is found, and "L" indicates that the MIP solver stopped due to the time limit condition and this value is a lower bound. Column CALB is the best result by the combining method with the capacity scaling and local branching heuristics. "*" indicates that the combining method finds the same upper bound as the current best upper bound, and "**" indicates that it finds the new best upper bound.

In Table 1, when compared to MIP tabu search, which is the best result among four other heuristics, the combining method improves average gaps by 0.52%. In Tables 2, the best solutions for 36 out of 37 problems and the new best solutions for 26 problems in C instances can be obtained by the combining method. The best solution for only one problem cannot be found.

Table 3 displays the computation times in CPU seconds for the capacity scaling heuristics, the combining method and four other heuristics. These computational times are reported in their papers. Due to the fact that different CPUs are used, these computation times cannot be compared directly. But compared to other heuristics, the computational times by the capacity scaling heuristics are very short, such as 76.3 seconds. The combining method can be solved within a reasonable computation time, from 503.5 to 1550.5 seconds.

Table 4 displays the average gaps of results for R instances. SIMP is the result by simplex-based tabu search, PARA by cooperative parallel tabu search, and CYCL by cycle-based tabu search. Path relinking

heuristics and MIP tabu search heuristics solved R instances, but these average gaps are not listed because of no detailed results in their papers. In Table 4, when compared to CYCL, the combining method improves average gaps for 3.45%. Table 5 and 6 display the distribution of the average gaps relative to the optimal value/upper bound by CPLEX reported in [6]. The reason for using the gaps based on upper bounds instead of lower bounds is to compare the results of the combining method against one of the path relinking heuristics. Table 5 displays the average gaps for the upper bound by RL and CALB, according to the design cost level and the capacity level of R instances. Table 6 displays the same information, but according to problem dimensions. In table 5, the gap ranges of path relinking heuristics varies from 0.76% to 4.11%, but the range of the combining method varies from -0.62% to 0.00%. The negative value means that the upper bound is better than one by CPLEX. In Table 6, the gap ranges of path relinking varies from 0.00% to 6.84%, but the range of the combining method varies from -2.24% to 0.00%. All average gaps in left column by CALB are 0.00%, and that indicates that the optimal value is found by the combining method and CPLEX.

Table 7 displays the computation times in CPU seconds for the capacity scaling heuristics, the combining method and three other heuristics, these computational times of which are reported in their papers. Compared to other heuristics, the computational times by the capacity scaling heuristics are very short, such as 45.7 seconds. The combining method can be solved within a reasonable computation time, from 211.0 to 1390.5 seconds.

The combining method with the capacity scaling heuristics and the local branch heuristics proposed in this paper performs satisfactorily for CND. By these computational results, the combining method can offer high quality solutions with a reasonable computation time, and improve most current best solutions. See detailed results at http://www.rku.ac.jp/~katayama/ sub02english.htm.

7 CONCLUSION

In this paper, the combining method with the capacity scaling heuristics using the column-row generation technique for the strong formulation of CND and local branching heuristics are proposed. The performance of the combining method was evaluated by solving C and R instances, and computational results are satisfactory. The combining method can find the best or equivalent solutions for 97% of C instances, and new best solutions for 70% of problems.

The proposed combining method can offer high quality results. For the column-row generation technique, the computational effort can be reduced considerably. The combining method proposed in this paper offers one of the best current results among approximate solution algorithms to resolve CND.

8 ACKNOWLEDGMENTS

This research was partially supported by the Ministry of Education, Culture, Sports, Science and Technology, Grant-in-Aid for Scientific Research (C), 21510155.

9 REFERENCES

[1] Magnanti T.L., Mireault P., Wong R.T., Tailoring benders decomposition for uncapacitated network design, Mathematical Programming Study ,1986, 26, 112-155.

[2] Holmberg K., Yuan D., A Lagrangian heuristics based branch-and-bound approach for the capacitated network design problem, Operations Research, 2000, 48, 461-481.

[3] Chouman M., Crainic T.G., Gendron B., A Cutting-plane algorithm for multicommodity capacitated fixed-charge network design, CRT-2009-20, CIRRELT, Université de Montréal, 2009.

[4] Costa A.M., Cordeau J., Gendron B., Benders, metric and cutset inequalities for multicommodity capacitated network design, Computational Optimization and Applications, 2009, 42, 371-392.

[5] Crainic T.G., Gendreau M., Farvolden J., A simplex-based tabu search for capacitated network design, Journal on Computing, 2000, 12, 223-236.

[6] Ghamlouche I., Crainic T.G., Gendreau M., Cycle-based neighbourhoods for fixed-charge capacitated multicommodity network design, CRT-2001-01, CIRRELT, Université de Montréal, 2001.

[7] Ghamlouche I., Crainic T.G., Gendreau M., Cycle-based neighborhoods for fixed-charge capacitated multicommodity network design, Operations Research, 2003, 51, 655-667.

[8] Ghamlouche I., Crainic T.G., Gendreau M., Path relinking, cycle-based neighbourhoods and capacitated multicommodity network design, Annals of Operations Research, 2004, 131, 109-134.

[9] Alvarez A.M., González-Velarde J.L., De-Alba K., Scatter search for network design problem, Annals of Operations Research, 2005, 138(1), 159--178.

[10] Crainic T.G., Gendron B., Cooperative parallel tabu search for capacitated network design, Journal of Heuristics, 2002, 8, 601-627.

[11] Crainic T.G., Li Y., Toulouse M., A first multilevel cooperative algorithm for capacitated multicommodity network design, Computers & Operations Research, 2006, 33, 2602-2622.

[12] Crainic T.G., Gendron B., Hernu G., A slope scaling/ Lagrangean perturbation heuristics with long-term memory for multicommodity capacitated fixed-charge network design, Journal of Heuristics, 2004, 10, 525-545.

[13] Katayama N., Chen M., Kubo M., A capacity scaling heuristics for the multicommodity capacitated network design problem, Journal of Computational and Applied Mathematics, 2009, 232, 90-101.

[14] Rodríguez-Martín I., Salazar-González J.J., A local branching heuristics for the capacitated fixed-charge network design problem, Computers & Operations Research, 2010, 37, 575-581.

[15] Hewitt M., Nemhauser G.L., Savelsbergh M., Combining exact and heuristics approaches for the capacitated fixed charge network flow problem, Journal on Computing, 2010, 22, 314-325.

[16] Chouman M., Crainic T.G., A MIP-Tabu search hybrid framework for multicommodity capacitated fixed-charge network design, CRT-2010-31, CIRRELT, Université de Montréal, 2010.