In the name of God

Network Flows

8. Capacitated Multicommodity Network Design Problems
8.1 Simplex-Based Tabu Search

Fall 2010

Instructor: Dr. Masoud Yaghini

1. Introduction

Introduction

- The goal of the paper (Crainic et al. (2000))is to present an efficient procedure to find good feasible solutions to fixed-charge, capacitated, multicommodity network design problem with linear costs (CMND).

Introduction

- The contribution of this paper is two-fold.
  - First, it provides an efficient heuristic for a difficult network optimization problem.
  - Second, it expands our understanding of the tabu search metaheuristic by studying the relationships among the tabu search framework, simplex pivoting, and column generation.

2. Formulation and Algorithmic Ideas

Formulations

- $\mathcal{G} = (\mathcal{N}, \mathcal{A})$: the graph underlying the network to be designed,
- $\mathcal{N}$: the set of nodes,
- $\mathcal{A} = \{a = (i, j) \mid i \in \mathcal{N}, j \in \mathcal{N}\}$ the set of arcs (for the sake of simplicity, we assume all arcs to be directed design arcs)
- $\mathcal{P}$: the set of commodities to be distributed.
- $p \in \mathcal{P}$ a commodity to be the flow between an origin node $r^p \in \mathcal{N}$ and a destination node $s^p \in \mathcal{N}$
- w^p : the corresponding demand is denoted w^p

Formulations

- c_p^a : the unit cost of moving commodity p through the arc a
- f_a : the fixed-cost of including the arc in the design of the network
- u_a : the total capacity of the arc.
- $\mathcal{L}^p$: all paths from r^p to s^p in $\mathcal{G}$.
- $\delta_{al}^p = 1$ if arc a is in the l^{th} path for commodity p, and is 0 otherwise,
- k_l^p : the variable (or per-unit) cost of path $l \in \mathcal{L}^p$

$$k_l^p = \sum_{a \in \mathcal{A}} c_a^p \delta_{al}^p.$$

Formulations

- y_a : the decision variables concern the selection of arcs for the final design of the network, $y_a = 1$ if arc a is included (opened) and 0, otherwise
- h_l^p : the flow of commodity p on path l
- x_a^p : the arc flows, can be obtained by

$$x_a^p = \sum_{l \in \mathcal{L}^p} h_l^p \delta_{al}^p.$$

Formulations

- The general form of the path-based formulation of the fixed-cost, capacitated multicommodity network design problem (PCMND) may then be written as:

Formulations

$$\text{Minimize } z(h, y) = \sum_{a \in \mathcal{A}} f_a y_a + \sum_{p \in \mathcal{P}} \sum_{l \in \mathcal{L}^p} k_l^p h_l^p$$

$$\text{Subject to } \sum_{l \in \mathcal{L}^p} h_l^p = w^p \quad \forall p \in \mathcal{P},$$

$$\sum_{p \in \mathcal{P}} \sum_{l \in \mathcal{L}^p} h_l^p \delta_{al}^p \leqslant u_a y_a \quad \forall a \in \mathcal{A},$$

$$h_l^p \geqslant 0 \quad \forall p \in \mathcal{P}, l \in \mathcal{L}^p,$$

$$y_a = \{0, 1\} \quad \forall a \in \mathcal{A}.$$

Algorithmic Ideas

- for any set of values $\tilde{y}$ of the design variables y, the path-based formulation of the CMND becomes a capacitated multicommodity minimum cost network flow (MCNF) problem in path flow variables

Algorithmic Ideas

- $H(\tilde{y})$: the set of all feasible path flows in PCMND for a given $\tilde{y}$.
- $H(\tilde{y})$: is also the set of all feasible path flows for the corresponding MCNF, for a given $\tilde{y}$.
- an optimal solution to MCNF problem can always be found at an extreme point of $H(\tilde{y})$, assuming $H(\tilde{y}) \# \emptyset$.
- $\tilde{H}(\tilde{y})$: the set of extreme points.

Algorithmic Ideas

- Consider now $H = H(\tilde{y} = 1)$
- the set of feasible path flows when all the design arcs are open (i.e., $\tilde{y}_a = 1$, $a \in A$), and its associated set of extreme points $\tilde{H}$.
- H is defined by the set of constraints:

$$\sum_{l \in \mathcal{L}^p} h_l^p = w^p \quad \forall p \in \mathcal{P},$$

$$\sum_{p \in \mathcal{P}} \sum_{l \in \mathcal{L}^p} h_l^p \delta_{al}^p \leqslant u_a y_a \quad \forall a \in \mathcal{A},$$

$$h_l^p \geqslant 0 \quad \forall p \in \mathcal{P},\ l \in \mathcal{L}^p,$$

Algorithmic Ideas

- it is verified that

$$\tilde{\mathcal{H}} = \cup_{\tilde{y} \in \mathcal{Y}} \tilde{\mathcal{H}}(\tilde{y}),$$

- where y is the set of all possible design variable vectors.
- for any given path flow pattern $\tilde{h} \in \tilde{H}$, a design with minimal cost relative to the path flows, denoted $y(\tilde{h})$, can be obtained by

$$y_a(\tilde{h}) = \begin{cases} 0 & \text{if } \sum_{p \in \mathcal{P}} \sum_{l \in \mathcal{L}_a^p} \delta_{al}^p \tilde{h}_a^p = 0 \\ 1 & \text{otherwise} \end{cases}$$

Algorithmic Ideas

- Where L_a^p stands for the set of paths for commodity p that use the design arc a , $L_a^p \subset L^p$
- An optimal solution to the CMND can therefore be found by complete or implicit enumeration of the elements of $\tilde{H}$.

Algorithmic Ideas

- A solution procedure for the CMND based on enumerating the elements of $\tilde{H}$ is likely to be extremely time-consuming
- Alternatively, the previous property may be used to define local search heuristics for the CMND: the adjacency relationships in $\tilde{H}$ define a natural neighborhood structure, while the pivoting rules of the simplex method provide an efficient way to reach the neighbors of any given solution.
- Thus, the fundamental idea of the approach we propose for solving the CMND is to use a tabu search method to explore the solution space $\tilde{H}$,

Algorithmic Ideas

- In the proposed method the ***revised simplex algorithm*** offers the framework within which the neighborhood is constructed and explored, and moves are evaluated, selected, and implemented.

Algorithmic Ideas

- The algorithm exhibits
  - the main structure of the primal simplex method with basis partitioning and
  - column generation.

3. Tabu Search for Network Design

Tabu Search for Network Design

- TS method:

1. Obtain initial solution;
   $\mathscr{C}$: Set of candidates $-$ current paths;
   $Z_{best} = \infty$; $div = 0$.
2. Local Search.
3. If $div < max\_div$ then
   - Perform diversification moves;
   - $div = div + 1$;
   - Go to Step 2.
4. Else STOP.

Tabu Search for Network Design

- *max_div* : a predefined number of diversification phases
- Z_{best} : the best solution encountered during the search
- Computations stop once a predefined number of diversification phases (*max_div)* have been performed.

The Solution Space

- the solution space we want to explore is the set

$$\tilde{\mathcal{H}} = \cup_{\tilde{y} \in \mathcal{Y}} \tilde{\mathcal{H}}(\tilde{y})$$

  - of the extreme points of *H*, defined over the set of all possible network configurations (design variable vectors) *y* by equations:

$$\sum_{l \in \mathcal{L}^p} h_l^p = w^p \qquad \forall p \in \mathcal{P},$$

$$\sum_{p \in \mathcal{P}} \sum_{l \in \mathcal{L}^p} h_l^p \delta_{al}^p \leqslant u_a y_a \quad \forall a \in \mathcal{A},$$

$$h_l^p \geqslant 0 \qquad \forall p \in \mathcal{P}, l \in \mathcal{L}^p.$$

3.1 Neighborhoods and Moves

Neighborhoods and Moves

- Two neighborhoods are defined:
  - a ***continuous neighborhood*** for the local search phase
  - a ***discrete neighborhood*** used to diversify the search

Neighborhoods and Moves

- The ***continuous (or local) neighborhood***, $N(\tilde{h})$, of any element $\tilde{h} \in \tilde{H}$, is defined as
  - the subset of all extreme points of $\tilde{H}$ adjacent to $\tilde{h}$, that is, all extreme points that can be reached from $\tilde{h}$ by one simplex pivot.
  - Consequently, in linear programming terms, a ***local move*** corresponds to a transition from one basis of the system to an adjacent one: a basic path thus becomes unbasic, while a previously unbasic path takes its place in the basis.

Neighborhoods and Moves

- Generally, not all the path variables are available at each iteration.
- Consider the sets $L_t^p = \{$available paths for commodity p at iteration $t\}$, for all $p \in P$, and H_t the corresponding polyhedron defined by:

$$\sum_{l \in \mathcal{L}_t^p} h_l^p = w^p \quad \forall p \in \mathcal{P},$$

$$\sum_{p \in \mathcal{P}} \sum_{l \in \mathcal{L}_t^p} h_l^p \delta_a^p + s_a = u_a \quad \forall a \in \mathcal{A},$$

$$h_l^p \geqslant 0 \quad \forall p \in \mathcal{P}, l \in \mathcal{L}_t^p,$$

$$s_a \geqslant 0 \quad \forall a \in \mathcal{A},$$

Neighborhoods and Moves

- where s_a are slack variables associated with arc a.
- Then, $\tilde{h}_t$, the current solution at iteration t, is an extreme point of H_t, and there exists a subset of neighbors, $N_t(\tilde{h}_t) \subseteq N(\tilde{h}_t)$, which are all exclusively made up of paths in

$$\cup_{p \in \mathcal{P}} \mathcal{L}_t^p, \forall p \in \mathcal{P},$$

- and thus are adjacent extreme points of H_t.

Neighborhoods and Moves

- To explore this neighborhood, we partition the h_t solution into its basic $\tilde{h}_B$ and *nonbasic* $\tilde{h}_N$ components.

$$\tilde{h}_B \subset \cup_{p \in \mathcal{P}} \mathcal{L}_t^p$$
$$\tilde{h}_N \subset \cup_{p \in \mathcal{P}} \mathcal{L}_t^p$$

- A ***candidate pivot-move*** is then simply a pair of variables (h_{in}, h_{out}), $h_{in} \in \tilde{h}_N$ *and* $h_{out} \in \tilde{h}_B$, on which a simplex pivot may be performed.
- The ***candidate list*** $\mathcal{I}$ is the set of all such pairs of variables.

Neighborhoods and Moves

- The ***discrete neighborhood*** is defined relative to the design variables and is used to modify drastically the network configuration and to diversify the search.
- A ***discrete neighbor*** $\hat{h} \in D(\tilde{h})$ *of* $\tilde{h} \in \tilde{H}$ at iteration t is the set of optimal paths in

$$\cup_{p \in \mathcal{P}} \mathcal{L}_t^p$$

- that corresponds to the network configuration $\hat{y}$ obtained by "closing" a given number of network arcs in $\tilde{y}$.

Tabu Search for Network Design

- The corresponding ***diversification move*** is thus a complex sequence of operations that usually includes several (primal and eventually dual) pivots.
- Several discrete neighborhoods may be defined (and several may be used in the same tabu search procedure) according to the number of arcs to be closed.
- The criteria used to select these arcs and determine the moment when diversification moves are to be performed are fundamental design parameters of the tabu search procedure.

3.2 Local Search

Local Search

- The **local search** phase aims to explore the space of the path flow variables according to the principles of the ***revised simplex method*** with ***column generation***.
- Two auxiliary MCNF formulations are defined over the constraint.

Local Search

- (1) The ***Linear component formulation***:

$$\text{Minimize } z(h) = \sum_{p \in \mathcal{P}} \sum_{l \in \mathcal{L}_t^p} k_l^p h_l^p$$

$$\sum_{l \in \mathcal{L}_t^p} h_l^p = w^p \quad \forall p \in \mathcal{P},$$

$$\sum_{p \in \mathcal{P}} \sum_{l \in \mathcal{L}_t^p} h_l^p \delta_a^p + s_a = u_a \quad \forall a \in \mathcal{A},$$

$$h_l^p \geqslant 0 \quad \forall p \in \mathcal{P}, l \in \mathcal{L}_t^p,$$

$$s_a \geqslant 0 \quad \forall a \in \mathcal{A},$$

Local Search

- Let α be the vector of dual variables associated with first and second constraints

Local Search

- The ***Linearized formulation***:

$$\text{Minimize } z(\tilde{h}) = \sum_{p \in \mathcal{P}} \sum_{l \in \mathcal{L}_t^p} \tilde{k}_l^p h_l^p$$

$$\sum_{l \in \mathcal{L}_t^p} h_l^p = w^p \quad \forall p \in \mathcal{P},$$

$$\sum_{p \in \mathcal{P}} \sum_{l \in \mathcal{L}_t^p} h_l^p \delta_a^p + s_a = u_a \quad \forall a \in \mathcal{A},$$

$$h_l^p \geqslant 0 \quad \forall p \in \mathcal{P},\ l \in \mathcal{L}_t^p,$$

$$s_a \geqslant 0 \quad \forall a \in \mathcal{A},$$

Local Search

- Where

$$\tilde{k}_l^p = \Sigma_{a \in \mathcal{A}} \, \delta_{al}^p (c_a^p + f_a / u_a)$$

- Let $\tilde{\alpha}$ be the vector of dual variables associated with first and second constraints

Local Search

- The local search phase steps are as follows:
- **Step 0**. Initialize short-term tabu lists;
  - l : Set of candidates - current paths;
  - Z_{best} : Value of the best overall solution obtained to date;
  - $Z_{local} = \infty$; $Z_{prev} = \infty$; $g = 0$.

Local Search

- ***Neighbor evaluation and move selection.***
- **Step 1.** Repeat steps 2 to 7 until *max_move* consecutive pivots (moves) have been executed with no improvement in the value of Z_{local}.
- **Step 2.** Let B identify the basis corresponding to the current solution;
  - Compute the values of the dual variables: $\alpha B = k_l^p$
  - Pivot in any slack variable with negative marginal cost.
- **Step 3.** Compute the reduced costs $\overline{k}_l^p$ for all nonbasic paths.

Local Search

- **Step 4.** For each nonbasic path variable i (potential h_{in} variable):
  - Determine the corresponding exiting (basic) path j (potential h_{out}) variable and associated value $\boldsymbol{\beta}$ of the entering variable;
  - Determine the corresponding modification (if any) to the vector of design variables y and compute the associated variation in the total fixed cost of the network $\boldsymbol{\Delta_{i,j}(Z(y))}$;
  - Compute the value of the potential move as the variation in the evaluation of $\boldsymbol{z(h, y)}$, the objective function of the original network design problem:

$$\boldsymbol{\Delta_{i,j}(Z) = \bar{k}_i\, \beta + \Delta_{i,j}(Z(y))}$$

Local Search

- **Step 5.** Identify (h_{in}, h_{out}) $argmin\{\Delta_{i,j}(Z)\}$.
- ***Pivot-move***
- **Step 6.** Move implementation.
  - If the selected move is not tabu, implement it.
  - If it is tabu, implement only if it improves Z_{best};
  - Otherwise, select next best candidate move.
- **Step 7.** Establish and update tabu tenures;
  - Update long-term memories;
  - If improved, update Z_{local} and, eventually, Z_{best}.

Local Search

- ***Column generation***
- **Step 8.**
  - If $Z_{local} < Z_{prev}$ then $Z_{prev} = Z_{local}$ and $g = 0$;
  - Else $g = g + 1$ and if $g >$ *max_col_gen* EXIT.
- **Step 9.**
  - Compute the values of the dual variables for the linearized formulation:

$$\tilde{\alpha}B = \tilde{k}_l^p;$$

  - Generate new paths and add them to l;
  - Go to Step 1.

Tabu List

- Several approaches are possible to prevent the tabu search procedure from cycling, such as:
  - forbidding the reversal of the last n_p pivots, might become computationally expensive
  - forbidding the last n_v paths to have entered the basis to leave it, might drive the procedure into a gridlock.
  - forbidding a exiting variable that recently got out of the basis to enter back into the basis (this is selected)

Tabu List

- The ***tabu tenure*** of each exiting variable is randomly chosen in a [*low_tab_mv*, *high_tab_mv*] interval according to a discrete uniform distribution.

Column Generation

- In a classical column generation scheme, the algorithm proceeds to price out existing nonbasic variables and then pivots until a local optimum is attained with respect to these variables, i.e. until all nonbasic variables display positive reduced costs.
- Then, a new, “improving” variable is generated, and the algorithm stops when this variable exhibits a positive reduced cost.
- In this paper, authors also call upon column generation only when a local optimum has been reached relative to the paths already generated (i.e., presently in l).

Column Generation

- However, in the present setting, one cannot use marginal costs to determine that no further improvement is possible by using the existing paths.
- Therefore, local optimality is defined as no improvement in the value of Z_{local} for *max_move* consecutive moves.

Column Generation

- A ***shortest path algorithm*** over the links of the network is used to generate new paths.
- To capture the interplay between the fixed-costs and capacities of the arcs, the reduced arc costs used by the column generation subproblem correspond to the ***linearized*** multicommodity capacitated network flow problem with the surrogate arc costs:

$$\tilde{c}_a^p = c_a^p + f_a / u_a, \forall a \in \mathcal{A}, p \in \mathcal{P}.$$

- All arcs are available for the generation of new paths, the dual variables associated with the capacity constraints keeping the procedure away from saturated arcs.

Column Generation

- Several strategies are possible relative to “how many” and to “which” path variables to generate each time the column generation phase is called.
  - One could, for example, generate one or several paths (in the latter case, one could generate a fixed or a variable number of paths)
  - One could do it for all or only for specific origins, destinations, or origin-destination pairs.
  - The choice of a strategy affects the efficiency of the search
  - Authors chose to generate a fixed number (identified as ***k_gen***) of paths for all origin-destination pairs, and to attempt to generate new paths for all O-D pairs each time the procedure is executed.

Column Generation

- When several paths have to be generated for an O-D pair, we want to obtain paths that are as dissimilar as possible, without having to implement a time-consuming procedure.
- Our approach iteratively starts from the successors of the origin (other than the one on the shortest reduced cost path) and computes the shortest marginal cost path from each of these nodes to the destination.
- If all such successors are used and still the required number of paths has not been generated, one proceeds to the successor node on the initial shortest path and applies the same procedure.

Column Generation

- In all cases, the path, or paths, with the lowest reduced cost(s) is (are) then added to the set of paths *l*.
- Note that, since this is not a classical simplex case where only variables with negative reduced cost are generated, we are not guaranteed that a newly generated path is not already in *l*.

Column Generation

- ***Column Generation Cycle***
  - A sequence of local neighborhood explorations (by pivoting) followed by a column generation phase is called a ***column generation cycle***.
  - Local search is terminated when no improvement to the current local best solution is obtained after a number of consecutive ***column generation cycles***.
  - The method then either stops or proceeds to diversify the search.
  - The parameter ***max_col_gen*** controls the termination of the local search procedure.

3.3 The Initialization Phase

The Initialization Phase

- Initially, a shortest path is generated for each demand using the surrogate arc costs.
- Demand is then sequentially loaded on these paths, the overflow being assigned to artificial paths with arbitrarily high costs.
- Starting from this principle, two basic initialization procedures may be devised.

The Initialization Phase

- ***The first method***
  - The first starts with all arcs open and performs simplex pivots without the tabu mechanisms until a first feasible solution is obtained to the PCMND.
  - This method is thus equivalent to a classical Simplex Phase I.
- ***The second method***
  - The second mechanism adopts the same Phase I philosophy, but makes use of the local search routine when looking for the first feasible solution, thus initiating the tabu logic from the very start.

3.4 Diversification

Diversification

- Once the completion of a local search sequence is signaled by ***max_col_gen*** consecutive column generation phases without improvement, the algorithm proceeds to a ***diversification phase***.
- This step aims to take the search out of an apparent local optimum and toward some promising region.

Diversification

- The diversification strategy implemented is based on the observation that when using column generation, a number of “good” arcs (with respect to capacity or to fixed-cost-to-capacity ratio) appear again and again in the paths used to satisfy demand.

Diversification

- ***Two simple long-term frequency memory structures*** have been implemented
  - The first records for how many iterations an arc has been in the basis, that is for how long it belonged to at least one basic path.
  - The second frequency vector records how often an arc was part of an entering path variable.
- In both cases, one attempts to capture a persistence type of attribute of the design arcs, in order better to move away from current solutions.

Diversification

- The experimental results have confirmed that the **first approach** to provide a more effective measure since it also captures a consistency attribute (for how long an arc was present “in” the basis) and somewhat filters out the arcs that are prone to get in and out of the basis in rapid succession.

Diversification

- Arcs that score high in any one of these two memories will tend to belong to the group of arcs often used in the solutions already explored.
- Therefore, a solution that does not use some of them will be in a different region than the ones already visited.
- Hence, to diversify, one selects a small number of often-used arcs and closes them.
- This is done via a procedure (**similar to dual simplex pivots**) that removes from the basis any path that contains at least one of the closed arcs.

Diversification

- This may result in an infeasible solution with flows on artificial paths; we will try to regain feasibility during the following pivoting operations.
- During the tabu tenure of these arcs, no paths that contain them are allowed to enter the basis unless, of course, the ***aspiration criterion*** overrides the tabu status.
- Furthermore, the closed arcs are not available during the column generation phases.
- These arcs and paths are tabu for ***tabu_cycle*** column generation cycles.

4. Calibration of the Tabu Search Metaheuristic

Calibration of the Tabu Search Metaheuristic

- The calibration process aims to determine values, or value ranges, for the search parameters such that the metaheuristic performs well over a broad range of problem types.
- A common pitfall of metaheuristic calibration phases is to tune too finely the procedure to a particular set of problems (or even to report performance results on the same set of problems used for calibration).

Calibration of the Tabu Search Metaheuristic

- To avoid this problem, we look for a set of parameters that is robust for a small set of representative problems, and then use the resulting parameter settings to experiment with two different sets of problem instances.

4.1 Local Search Calibration

Local Search Calibration

- The initial experiments did show that tight capacities and dominant fixed-costs make problems more difficult to solve.
- Therefore, for the calibration phase, we selected 10 problems (out of the initial 18) that display these attributes and cover the entire range of network sizes: from 100 to 700 design arcs and from 10 to 400 commodities.
- We were also able to fix the value of the ***max_move*** parameter to twice the value of the maximum tabu tenure ***high_tab_mv***.

Local Search Calibration

- We tested the following combinations of parameter values, without activating the diversification feature, and fixing ***max_col_gen*** at 10 (which allows for a long local search phase):
  - Tabu tenure interval for exiting path variables [***low_tab_mv, high_tab_mv***]*:* [5,10], and [10,20];
  - The type of initial solution: simplex Phase I without tabu mechanisms (*YES*) or no independent Phase I and the tabu mechanisms of the local search start from the possibly infeasible initial flow allocation (*NO*);
  - The number of paths generated for each commodity (origin-destination pair) during the column generation cycle: ***k_gen*** = 1, 2, 3, and 5.

Local Search Calibration

- The success of each parameter combination was measured by the number of first, second, and third best solutions it found over the set of 160 runs (10 problem instances and 16 parameter combinations).

Local Search Calibration

- **Individual Parameter Performance**

Parameter	1st Place	2nd Place	3rd Place
	Initialization procedure		
YES	5	4	3
NO	7	11	7
	low_tab_mv, high_tab_mv, max_move		
5,10,20	6	6	4
10,20,40	9	6	6
	k_gen		
1	4	3	5
2	3	2	1
3	4	5	2
5	4	5	2

Local Search Calibration

- Table I allows us to gain some insight into the behavior induced by each parameter:
  - Activation of the tabu logic (NO option) from the very beginning of the initialization phase appears best.
  - A tabu tenure randomly chosen from a [10,20] interval performs marginally better than when the [5,10] is used.
  - Generating 3 or 5 paths appears to be preferable to the other two options. However, since the computation time of this phase increases with the number of generated paths, we prefer to generate only 3 paths for each commodity.

Local Search Calibration

- To make the final decision, we weighted each first, second, and third places with three, two, and one points, respectively.
- The resulting relative performance of the various combinations of parameters is displayed in:

Setting	*k_gen=1*	*k_gen=2*	*k_gen=3*	*k_gen=5*
10,20,40,Yes	3	0	1	12
10,20,40,No	6	8	12	9
5,10,20,Yes	6	3	3	1
5,10,20,No	8	3	8	2

Local Search Calibration

- One combination stands out and was used in all the tests that followed: *10,20,40,No,k_gen = 3*, that is,
  - no separate initialization phase,
  - tabu tenure selected from a [10,20] iterations interval,
  - 40 consecutive unimproving moves to decide that a current local best solution is a local optimum with respect to the existing paths, and
  - 3 paths generated for each commodity during a column generation phase.

Local Search Calibration

- In the present case, the combination *10,20,40,Yes,k_gen = 5* scored as high as the one we selected (but required the generation of five paths for each product), while several others are not far behind.
- The selected set of parameters performs very satisfactorily over the range of test problems:
  - the average gap between the solutions obtained by using the selected set of parameters and the best solutions is 1.2%,
  - the maximum observed difference is of the order of 4%.

4.2 Calibration of Diversification

Calibration of Diversification

- The diversification phase requires that two more parameters be considered and jointly fixed:
  - ***max_col_gen***, which controls when the local search terminates and when a diversification phase is initiated,
  - ***tabu_cycle***, the tabu tenure of the arcs closed by the diversification move.
- After a number of tests, we decided to use *max_col_gen* = *3* and *tabu_cycle* = *2*.

Calibration of Diversification

- The diversification mechanisms are based on the idea of forcing the search in a direction where some of the arcs currently used heavily are not part of the solution
- we tried to close an important number of arcs of the order of 1% and 5%.

5. Experimentation and Analyses

Experimentation and Analyses

- The main objectives of the experimentation phase are:
  - (i) to characterize the behavior of the search algorithm where pivoting and column generation are driven according to tabu search principles
  - (ii) to gain insight into the performance of the method relative to three main problem instance characteristics: dimensions, relative importance of fixed-costs compared to variable costs, and the degree of capacity tightness
  - (iii) to establish how well the proposed algorithm solves the type of problems of interest here

Experimentation and Analyses

- ***Test Problems***
  - Two sets of problems have been generated.
  - These are general transshipment networks, with no parallel arcs and one commodity per origin-destination pair.
  - On each arc, the same unit cost is used for all commodities.
  - Problems differ in the number of nodes, arcs (all of which are design arcs), and commodities.
  - Several instances have been generated for each problem dimension by varying the relative importance of fixed versus variable costs and the capacity of the network compared to the total demand.

Experimentation and Analyses

- The tabu search metaheuristic is programmed in *FORTRAN77*.
- The experiments reported have been performed on a SUN UltraSparc-II workstation (two CPUs, but only one allocated to our experiments), with a 296 MHz clock, 2 MB of cache memory, and 2 GB of RAM.

Performance Analysis

- To analyze the behavior and performance of the tabu search heuristic, we compare its output to the optimal solution obtained by using the **branch-and-bound algorithm** of **CPLEX version 4.0**, with primal simplex-based bounding
- We also compare the tabu search to two other heuristics.

Performance Analysis

- **The first heuristic method (GREEDY)**
  - accepts only improving (pivot) moves in Step 5 of local search.
  - This corresponds to a classical greedy descent procedure.
  - Contrasted to the results of the tabu search method, it acts as a measure of the impact on the solution quality of the imposition of a metaheuristic (tabu search) logic over the greedy search.

Performance Analysis

- **The second heuristic method (UBR)**
  - The upper bound heuristic makes use of the information yielded by the lower bounding procedure and combines projection and resource decomposition methods

Performance Analysis

- **Computational Results**

PROB	GREEDY	UBR	OPT	TABU	GAP
20,230,40,V,L	431179	431701	423848 (9.46)	425046 (71.29)	0.28%
20,230,40,F,T	4.3e+10	404638	371475 (82.27)	371816 (90.28)	0.09%
20,230,40,F,T	3.6e+09	679539	643036 (1639.34)	644172 (121.79)	0.17%
20,230,200,V,L	146464	164770	94752 (t)	122592 (504.50)	29.38%
20,230,200,F,L	232817	299590	139888 (t)	188590 (491.63)	34.82%
20,230,200,V,T	136978	204486	98051 (t)	118057 (548.36)	20.40%
20,230,200,F,T	6.8e+06	277365	137796 (t)	182829 (889.69)	32.68%
20,300,40,V,L	431839	447235	429398 (1.25)	429912 (71.05)	0.12%
20,300,40,F,L	624978	734217.1	586077 (188.53)	589190 (113.44)	0.53%
20,300,40,V,T	3.2e+10	481738	464509 (176.91)	464509 (145.33)	0%
20,300,40,F,T	1.2e+11	650874	604198	606364	0.35%

Performance Analysis

20,300,200,V,L	104195	168510	75460 (t)	88398 (982.21)	17.15%
20,300,200,F,L	205254	300507	116810 (t)	151317 (1316.75)	29.54%
20,300,200,V,T	101772	162044	75090 (t)	82724 (938.29)	10.17%
20,300,200,F,T	2.5e+08	302376	112650 (t)	135593 (1065.88)	0.40%
25,100,10,V,L	14763	14828	14712 (0.12)	14712 (5.60)	0%
25,100,10,F,L	17073	20182	14941 (91.47)	15889 (8.37)	6.35%
25,100,10,F,T	1.2e+07	57314	49899 (441.86)	51654 (17.10)	3.52%
25,100,30,V,T	6.1e+09	382923	365272 (20.35)	365272 (16.57)	0%
25,100,30,F,L	6.9e+06	48782	37055 (15274.21)	38804 (33.01)	4.72%
25,100,30,F,T	3.0e+08	99896	85530 (2283.71)	86445 (71.84)	1.07%

Performance Analysis

- Problems are identified with a quintuplet that indicates
  - (i) the number of nodes
  - (ii) the number of arcs
  - (iii) the number of commodities
  - (iv) if fixed-costs are relatively high (F) or low (V) compared to variable costs
  - (v) if the problem is tightly (T) or somewhat loosely (L) capacitated.

Performance Analysis

- **Table information:**
  - the **GREEDY** and **UBR** columns display the value of the best feasible solution obtained by the greedy descent and the resource-decomposition heuristic, respectively.
  - The **OPT** column corresponds to the branch-and-bound algorithm of CPLEX version 4.0, with primal simplex-based bounding.
  - In the **TABU** column, the results of tabu search methods with tuned parameters are depicted

Performance Analysis

- Table information:
  - **X** indicates that the procedure failed to find a feasible solution.
  - The figures in parentheses in the **OPT** and **TABU** columns represent total computation times, in CPU seconds;
  - **t** indicates that the procedure stopped due to a time limit condition.
  - When branch-and-bound has identified a feasible solution, the **GAP** column displays the optimality gap of the tabu procedure relative to it.
  - A limit of 6 hours (21000 CPU seconds) computation time was imposed on the branch-and-bound.

Experimentation and Analyses

PROB	GREEDY	UBR	OPT	TABU	GAP
30,520,100,V,L	4.6e+08	67699	54104 (t)	56426 (995.64)	4.29%
30,520,100,F,L	1.0e+09	221188	96450 (t)	104117 (939.24)	7.94%
30,520,100,V,T	2.6+09	75757	52258 (t)	53288 (1218.52)	1.97%
30,520,100,F,T	115244	185746	99870 (t)	107894 (670.29)	8.03%
30,520,400,V,L	1.9e+10	226070	113021 (t)	125831 (5789.27)	11.33%
30,520,400,F,L	3.0+08	389636	X (t)	177409 (6406.62)	—
30,520,400,V,T	3.1e+11	X	X (t)	125518 (6522.23)	—
30,520,400,F,T	2.0e+10	336248	X (t)	174526 (8415.24)	—
30,700,100,V,L	51962	59411	47603 (1775.69)	48984 (1265.11)	2.90%
30,700,100,F,L	73892	114603	60559 (t)	65356 (1479.59)	7.92%
30,700,100,V,T	2.8e+09	60918	46564 (t)	47083 (2426.02)	1.12%
30,700,100,F,T	64003	97023	55709 (t)	58804 (1735.72)	5.56%

Experimentation and Analyses

30,700,400,V,L	1.3e+10	215014	X (t)	110000 (12636.20)	—
30,700,400,F,L	1.8e+10	387566	X (t)	165484 (11367.70)	—
30,700,400,V,T	2.0e+10	202740	X (t)	103768 (15879.50)	—
30,700,400,F,T	2.12e+10	375586	X (t)	150919 (11660.40)	—
100,400,10,V,L	6.4e+07	30273	28433 (775.40)	28485 (32.66)	0.18%
100,400,10,F,L	28279	45188	25166 (t)	24912 (33.00)	−1.00%
100,400,10,F,T	2.4e+07	109244	71439 (t)	71128 (81.23)	−0.44%
100,400,30,V,T	9.0e+09	407085	385102 (383.77)	385185 (277.50)	0.02%
100,400,30,F,L	63000	110401	52368 (t)	58773 (100.16)	12.23%
100,400,30,F,T	1.7e+09	226054	145083 (t)	149282 (215.71)	2.89%

Experimentation and Analyses

- ***Second set of problems***
  - A second set of problems has been generated, therefore, in order to study these relationships further.
  - The problem instances in this set, identified with the letter R, have been designed to facilitate the evaluation of the impact on performance of one problem characteristic only.

Experimentation and Analyses

- **Distribution of Relative Gaps**

Prob. Set	X	Opt	Imp	(0%, 1%]	(1%, 5%]	(5%, 10%]	(10%, 20%]	>20%
C	7	3	2	9	8	5	4	5
R	15	16	14	19	52	24	8	5
All	22	19	16	28	60	29	12	10

  - Column **X :** for 22 problem instances, branch-and-bound did not find a feasible solution
  - Column **Opt** : the tabu search heuristic identified 19 optimum solutions
  - Column **Imp** : the tabu search heuristic improved the branch-and-bound feasible solution (no optimum known) for 16 other problems.
  - The next columns corresponds to the (0%, 1%] gap interval, and so on.

Experimentation and Analyses

- **Aggregate Relative Gaps**

Prob. Set	Improvement Gap	Strict Opt Gap
C	0.72% ($n = 2$)	8.97% ($n = 31$)
R	8.95% ($n = 14$)	5.45% ($n = 108$)
All	6.56% ($n = 16$)	6.24% ($n = 139$)

  - The figures in the **Improvement Gap column** correspond to the 16 problems for which the tabu search improved the branch-and-bound feasible solution.
  - For the 139 other problem instances, the tabu search method achieved an average gap relative to the branch-and-bound solution of 6.24% (the **Strict Opt Gap column).**

The End