In the name of God

Network Flows

8. Multicommodity Network Design Problems
8.2 Cycle-Based Neighbourhoods

Fall 2010

Instructor: Dr. Masoud Yaghini

1. Introduction

Introduction

- Reference:
  - Ghamlouche I., Crainic T.G., Gendreau, M. **Cycle-based neighbourhoods for fixed-charge capacitated multicommodity network design**. Operations Research 2003, 51:655–67.

Introduction

- The goal of this paper is to propose a new class of neighbourhoods for meta-heuristics aimed at the fixed-charge capacitated multicommodity network design formulation (CMND).
- The neighbourhood defines moves that may explicitly take into account the impact on the total design cost of potential modifications to the flow distribution of several commodities simultaneously.
- The fundamental idea is to explore the space of the arc design variables by re-directing flow around cycles and closing and opening design arcs accordingly.

Introduction

- **Compared to path-based neighbourhoods**
  - First, the move evaluations are more comprehensive (since all commodities on a cycle are explicitly considered)
  - Second, the range of moves is broader since flow deviations are no longer restricted to paths linking origins and destinations of actual commodities.
- To illustrate the quality of this neighbourhood, we implement two instances in a **tabu-based local search metaheuristic**.

Introduction

- Computational experiments on problems of various sizes (up to 700 arcs and 400 commodities) show that the tabu search algorithm based on these neighbourhood structures is quite powerful, outperforming already existing methods in solution quality for similar computational efforts.

Introduction

- **The contribution of this paper**
  - First, it introduces a new class of cycle-based neighbourhood structures for meta-heuristics aimed at the CMND, together with efficient procedures to identify good moves.
  - Second, to demonstrate the quality of the cycle-based neighbourhoods, it proposes a very simple tabu search procedure that offers the current best heuristic solutions for the CMND.

2. Mathematical Formulation

Mathematical Formulation

- Let $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ be a network with set of nodes $\mathcal{N}$ and set of directed arcs $\mathcal{A}$.
- We assume that all $(i, j \in \mathcal{A})$ are design arcs.
- Let $\mathcal{P}$ denote the set of commodities to move using this network, where each commodity p has a single origin $o(p)$, a single destination $s(p)$, and a flow requirement of w^p units between its origin and destination nodes.

Mathematical Formulation

- The arc-based formulation of the CMND can be written as follows:

$$\min\ z(x,y) = \sum_{(i,j)\in\mathcal{A}} f_{ij}y_{ij} + \sum_{p\in\mathcal{P}}\sum_{(i,j)\in\mathcal{A}} c_{ij}^{p}x_{ij}^{p}$$

$$\begin{aligned}
\text{subject to} \quad \sum_{j\in\mathcal{N}^{+}(i)} x_{ij}^{p} - \sum_{j\in\mathcal{N}^{-}(i)} x_{ji}^{p} &= d_i^{p} \quad \forall i\in\mathcal{N}, \forall\ p\in\mathcal{P}, \\
\sum_{p\in\mathcal{P}} x_{ij}^{p} &\le u_{ij}y_{ij} \quad \forall (i,j)\in\mathcal{A}, \\
x_{ij}^{p} &\ge 0 \quad \forall (i,j)\in\mathcal{A}, \forall p\in\mathcal{P}, \\
y_{ij} &\in \{0,1\} \quad \forall (i,j)\in\mathcal{A}
\end{aligned}$$

Mathematical Formulation

$\mathcal{N}^+(i)$: Set of outward neighbours of node i,
$\mathcal{N}^-(i)$: Set of inward neighbours of node i,
u_{ij} : Capacity of arc (i, j),
f_{ij} : Fixed cost applied on arc (i, j),
c_{ij}^p : Cost of one unit flow of commodity p on arc (i, j),

$$d_i^p = \begin{cases} w^p & if \quad i = o(p) \\ -w^p & if \quad i = s(p) \\ 0 & \text{otherwise.} \end{cases}$$

Mathematical Formulation

- For a given design vector $\bar{y}$, the arc-based formulation of the CMND becomes a capacitated multicommodity minimum cost flow problem (CMCF):

$$\min\ z(x(\bar{y})) = \sum_{p\in\mathcal{P}} \sum_{(i,j)\in\mathcal{A}(\bar{y})} c_{ij}^p x_{ij}^p$$

$$\text{subject to} \sum_{j\in\mathcal{N}^+(i)} x_{ij}^p - \sum_{j\in\mathcal{N}^-(i)} x_{ji}^p = d_i^p \quad \forall i \in \mathcal{N}, \forall\ p \in \mathcal{P},$$

$$\sum_{p\in\mathcal{P}} x_{ij}^p \leq u_{ij}\bar{y}_{ij} \quad \forall (i,j) \in \mathcal{A}(\bar{y}),$$

$$x_{ij}^p \geq 0 \quad \forall (i,j) \in \mathcal{A}(\bar{y}), \forall p \in \mathcal{P}.$$

- where $\mathcal{A}(\bar{y})$ stands for the set of arcs corresponding to the design $\bar{y}$.

3. Cycle-based Neighbourhoods

Cycle-based Neighbourhoods

- Consider the solution space of the design variables y.
- A solution to the CMND is then an assignment $\bar{y}$ of 0 or 1 to each design variable, plus the optimal flow of the corresponding multicommodity minimum cost flow problem $x^*(\bar{y})$.
- Similarly, the objective function value associated to a solution $(\bar{y}, x^*(\bar{y}))$ is the sum of the fixed cost of the open arcs in $\bar{y}$ and the objective function value of the CMCF associated to $x^*(\bar{y})$:

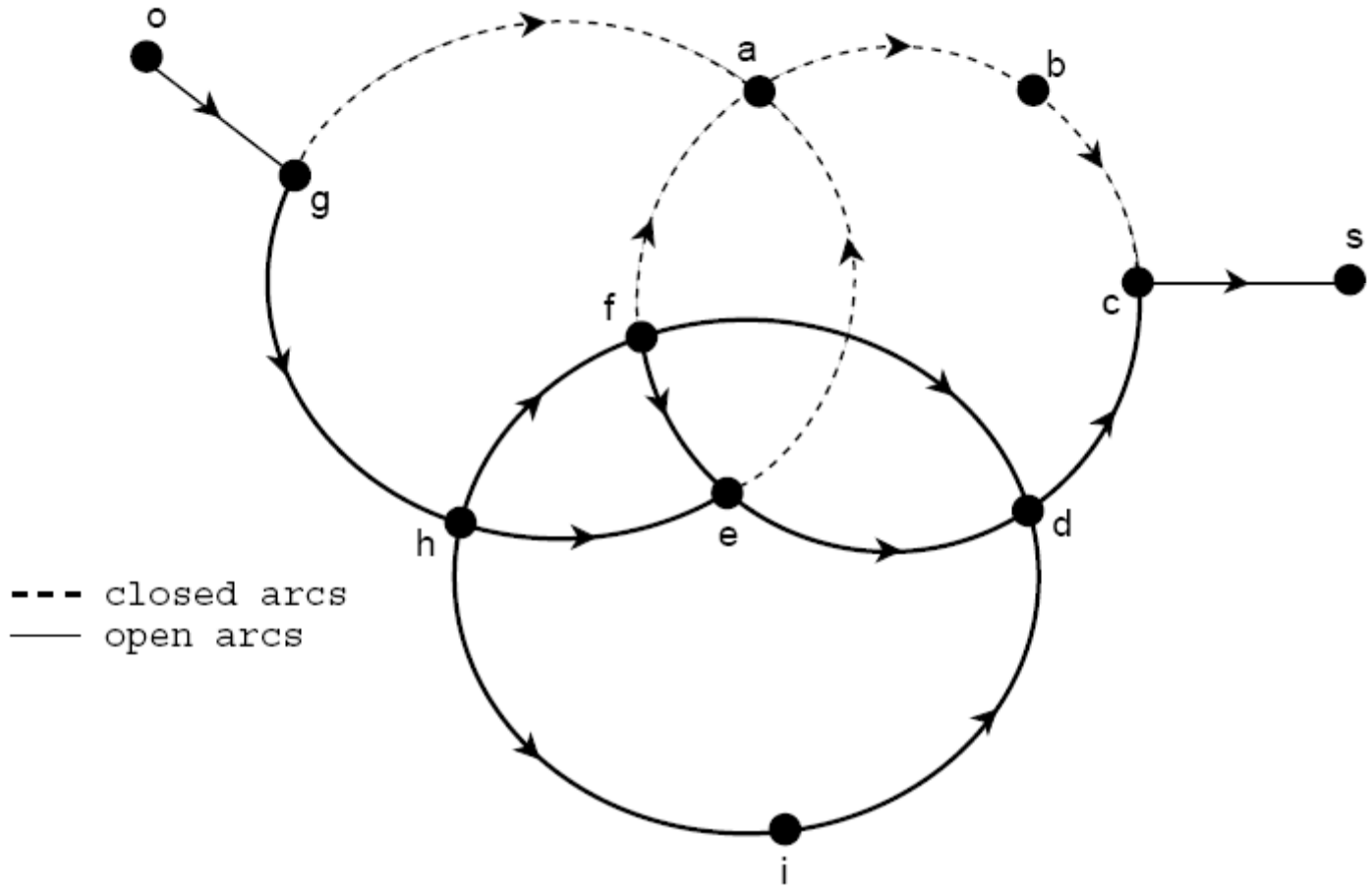
$$z(\bar{y}, x^*(\bar{y})) = \sum_{(i,j) \in \mathcal{A}(\bar{y})} f_{ij} \bar{y}_{ij} + z(x^*(\bar{y}))$$

Cycle-based Neighbourhoods

- **A Directed Path**:
  - A directed **path** from node i_0 to i_n is a sequence of closed or opened arcs f(i_0, i_1), (i_1, i_2), …, (i_{n-1}, i_n) such that the origin node of each arc is the destination node of the preceding arc in the sequence, and i_0, …, i_n are all distinct nodes.

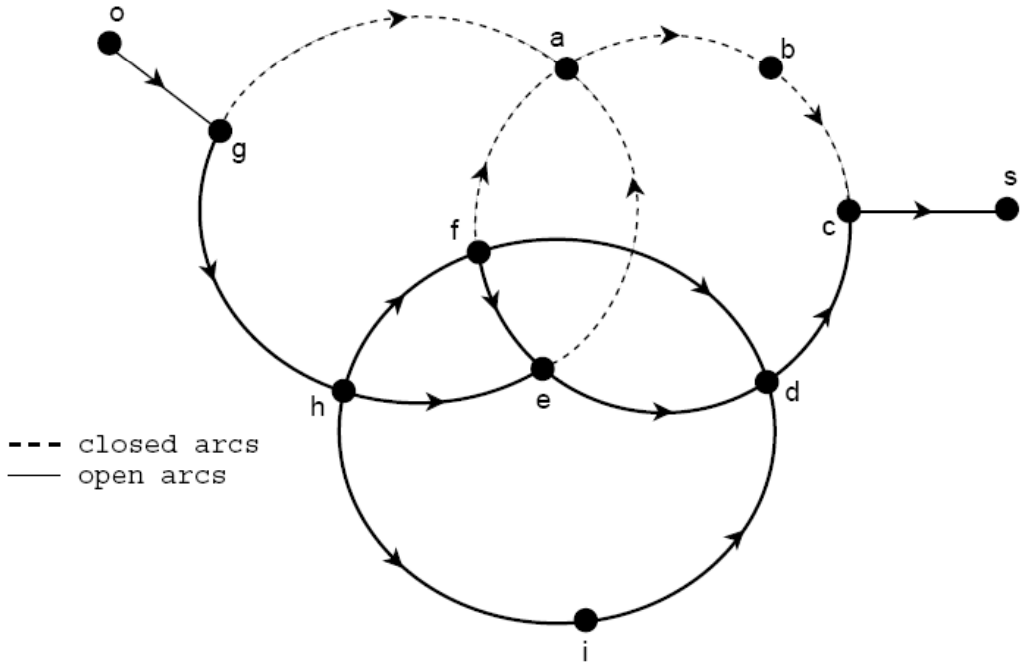
Cycle-based Neighbourhoods

- A path: $\{(h, f), (f, a), (a, b), (b, c), (c, s)\}$



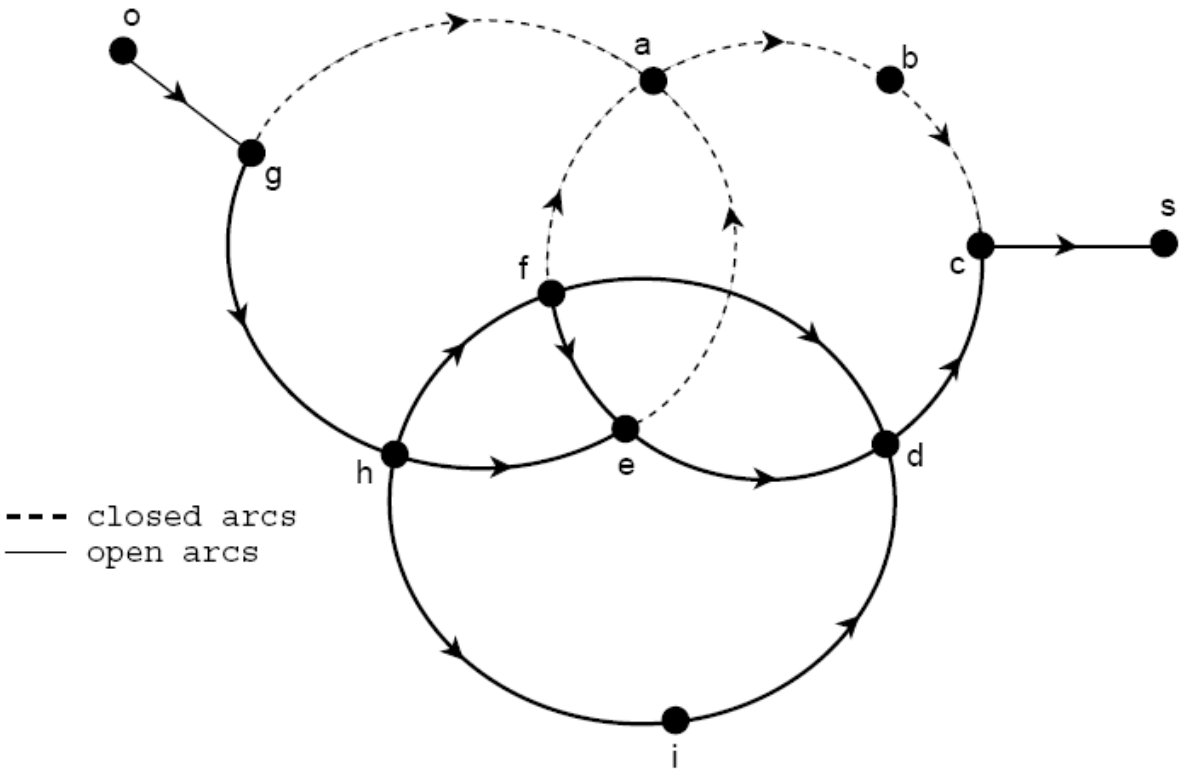
Cycle-based Neighbourhoods

- **A simple chain**
  - is similar to a path except that arcs are not restricted to follow the same direction.
  - A chain: $\{(a, b), (b, c), (c, d)\}$



Cycle-based Neighbourhoods

- **A cycle**
  - is a closed chain,
  - A cycle: $\{(a, b), (b, c), (c, d), (d, e), (e, f), (f, a)\}$



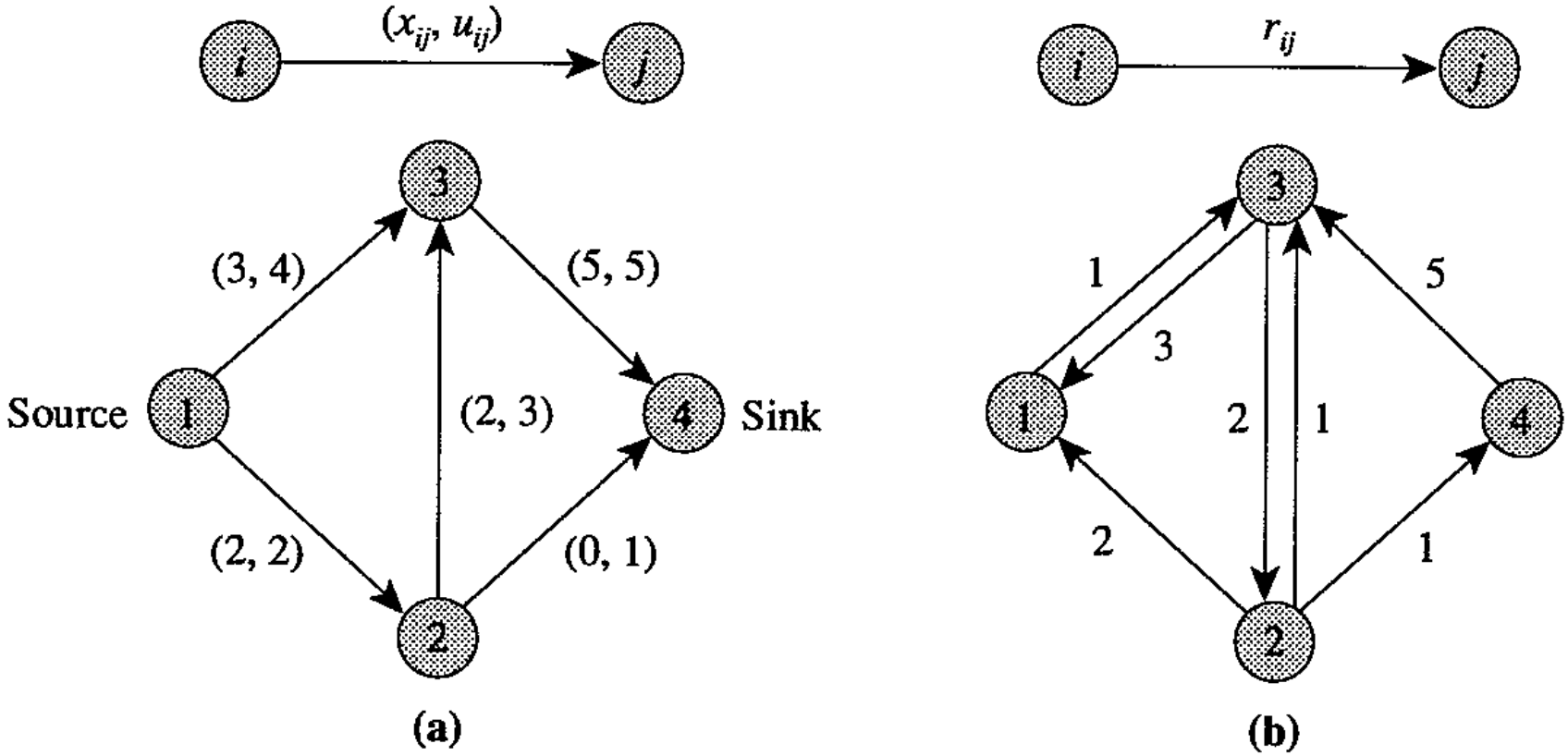
Cycle-based Neighbourhoods

- ***Residual network***
  - The concept of residual network plays a central role in the development of all the maximum flow algorithms we consider.
  - Given a flow x, the ***residual capacity*** r_{ij} of any arc $(i, j) \in A$ is the maximum additional flow that can be sent from node i to node j using the arcs (i, j) and (j, i).
  - The residual capacity r_{ij} has two components:
    - (1) u_{ij} - x_{ij}, the unused capacity of arc (i, j), and
    - (2) the current flow x_{ji} on arc (j, i), which we can cancel to increase the flow from node i to node j.

Cycle-based Neighbourhoods

- We refer to the network $G(x)$ consisting of the arcs with positive residual capacities as the **residual network** (with respect to the flow x).

Cycle-based Neighbourhoods



- (a) original network G with a flow x,
- (b) residual network $G(x)$, $\boldsymbol{r_{ij} = u_{ij} - x_{ij}, r_{ji} = x_{ij}}$

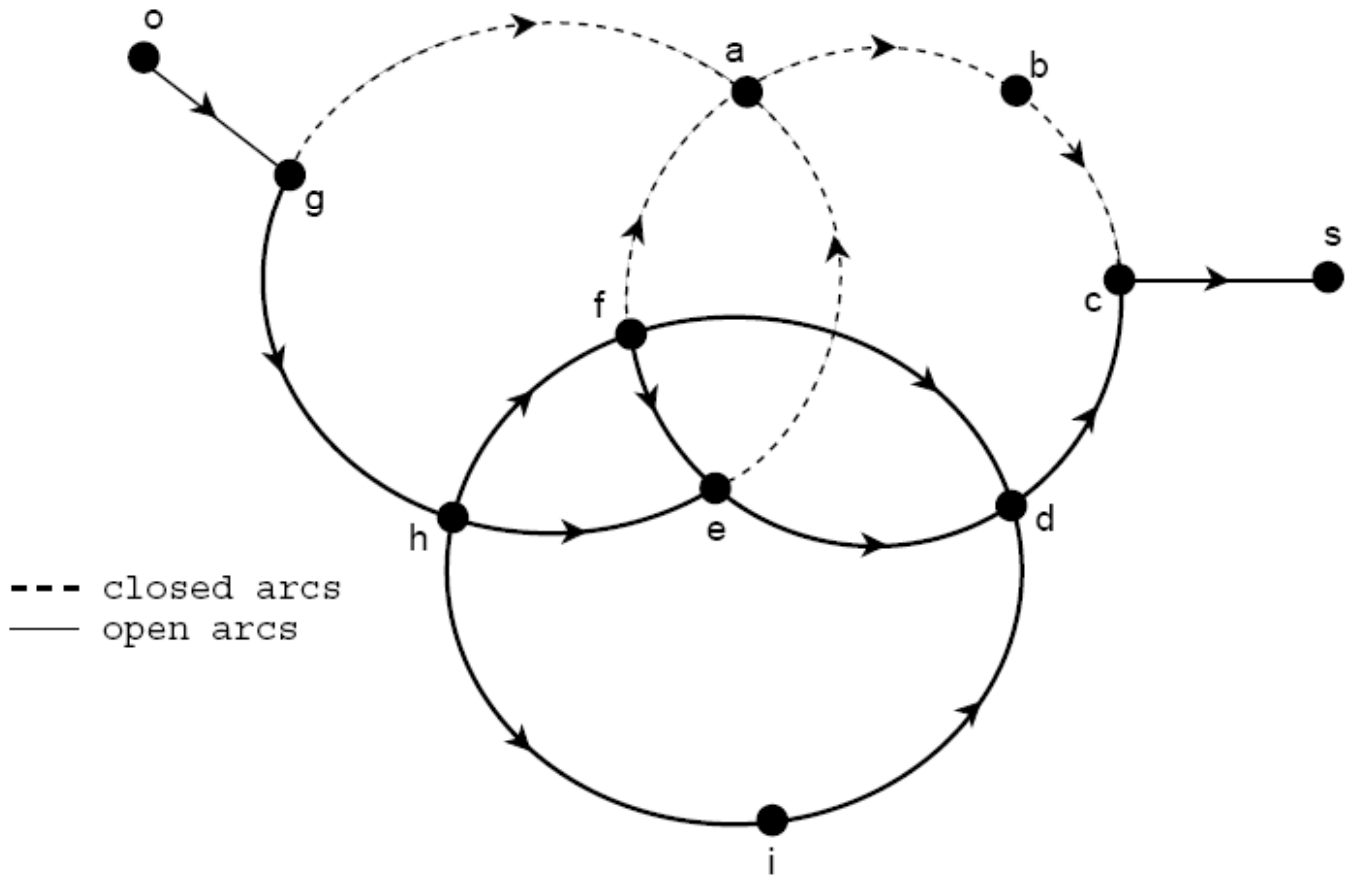
3.1 Neighbourhood Definition and Exploration

Neighbourhood Definition and Exploration

- The fundamental idea of the new neighbourhood class is that one may move from one solution to another by:
  - 1. Identifying two points in the network together with two paths connecting these points, thus closing a cycle,
  - 2. Deviating the total flow from one path to another such that at least one currently open arc becomes empty,
  - 3. Closing all previously open arcs in the cycle that are empty following the flow deviation and, symmetrically, opening all previously closed arcs that now have flow.

Neighbourhood Definition and Exploration

- **Example:** paths $\{(f, e), (e, d), (d, c)\}$ and $\{(f, a), (a, b), (b, c)\}$ in the network form a cycle and flow from the former may be deviated to the latter.



Neighbourhood Definition and Exploration

- Three arcs will thus be open, (f, a), (a, b), and (b, c) and, provided sufficient flow may be deviated, at least one previously open arc will be closed.
- The general neighbourhood structure we propose for the CMND may then be written as:
  - $\mathcal{V}(\bar{y}) = \{y$: obtained from $\bar{y}$ by complementing the status of a number of arcs following the deviation of flow in a given cycle in $\mathcal{A}(\bar{y})\}$

Neighbourhood Definition and Exploration

- Such neighbourhoods are huge and their explicit and exhaustive exploration is not practical in most situations.
- Moreover, the complete evaluation of any design modification involves the resolution of a capacitated multicommodity network flow problem, which rapidly becomes extremely computation intensive.
- Thus, in order to select the best move out of a given solution, one cannot simply identify explicitly all neighbours, evaluate them, and retain the best one.

Neighbourhood Definition and Exploration

- A more efficient procedure must be implemented that
  - 1) avoids the complete evaluation of every examined move and
  - 2) generates a limited number of cycles that include the "good" moves.

Neighbourhood Definition and Exploration

- Note that not all cycles are of equal interest.
- We seek, in particular, moves that modify the status of several arcs and that lead to a significant modification of the flow distribution.
- Therefore, moves that **close at least one arc** and open new paths for a group of commodities appear attractive.

Neighbourhood Definition and Exploration

- To close an arc (i, j), one must be able to deviate all its flow.
- **Residual capacity of a cycle**
  - Let the **residual capacity of a cycle** denote the **maximum flow** one can deviate around the cycle.
- The residual capacity of any cycle that includes (i, j) must then be at least equal to the total flow on arc (i, j):

$$\sum_{p \in \mathcal{P}} x_{ij}^{p}$$

Neighbourhood Definition and Exploration

- The cycles of interest to us are those that display a residual capacity equal to one of the values in $\mathbf{\Gamma}$ defined as the set of the total (positive) volumes on the open arcs of the corresponding network:

$$\Gamma(\bar{y}) = \left\{ \sum_{p \in \mathcal{P}} x_{ij}^{p} > 0 \ : \ (i,j) \in \mathcal{A}(\bar{y}) \right\}$$

- We thus progressively build a set of good candidate neighbours (cycles) among which the best move is then selected.

Neighbourhood Definition and Exploration

- Let $\mathcal{C} \subseteq \mathcal{A}$ denote the set of candidate links, where each arc $(i, j) \in \mathcal{C}$ will be considered as the starting point of cycles.
- $\mathcal{C}$ may include all the arcs in $\mathcal{G}$.
- Let $\mathcal{C}(\gamma) \subseteq \mathcal{C}$ include all arcs $(i, j) \in \mathcal{C}$ that can support a movement of γ units of flow.
- A closed arc may belong to $\mathcal{C}(\gamma)$ only if its capacity is at least γ.
- For an open arc, either its flow or its residual capacity must be at least γ units.
  - i.e. its flow can not be less than γ

Neighbourhood Definition and Exploration

- The heuristic procedure to explore the neighbourhood $\mathcal{V}(\bar{y})$ of a given solution $\bar{y}$, given a set of candidate links $C(\bar{y})$:

- Build set $\Gamma(\bar{y})$,
- For each $\gamma \in \Gamma(\bar{y})$
  - Build the γ-residual network,
  - For each arc $(i, j) \in C(\gamma)$ find the lowest cost cycle by using the network optimization procedure,
- Select and implement the best move,
- Evaluate new solution.

3.2 The γ-Residual Network

The γ-Residual Network

- The objective is to build a network such that, given a flow value γ, the network optimization procedure may
  - 1) identify low cost cycles that support the deviation of at least γ units of flow and
  - 2) explicitly consider the impact of a potential closure or opening of an arc due to the deviation of flow.

The γ-Residual Network

- To build the γ-residual network corresponding to a flow variation of γ units, replace each arc (i, j) of the original network by at most two arcs $(i, j)^+$ and $(j, i)^-$
- Arc $(i, j)^+$ is included only if an additional amount of γ units of flow may pass on arc (i, j), that is if its residual capacity

$$u_{ij} - \sum_{p \in \mathcal{P}} x_{ij}^p$$

- is greater or equal to γ.

The γ-Residual Network

- The cost c_{ij}^{+} associated to $(i, j)^{+}$ approximates the cost of routing γ units of additional flow on arc (i, j).
- It equals the average commodity routing costs on the arc, plus the fixed cost if it is currently closed:

$$c_{ij}^{+} = \frac{\sum_{p \in \mathcal{P}} c_{ij}^{p}}{\mid P \mid} \gamma + f_{ij}(1 - (\lceil \min(1, \sum_{p \in \mathcal{P}} x_{ij}^{p}) \rceil))$$

The γ-Residual Network

- Symmetrically, arc $(j, i)^-$ is not included in the γ-residual network if the total flow on arc (i, j):

$$\sum_{p \in \mathcal{P}} x_{ij}^p$$

- is less than γ.
- Otherwise, we associate to $(j, i)^-$ the cost c_{ji}^- that approximates the value of a reduction of γ units of the total flow (all commodities) currently using arc (i, j).
- This approximation is computed as the weighted average of the routing costs of the commodities currently using arc (i, j).
- The fixed cost of (i, j) is then subtracted if the reduction of γ units of flow leaves the arc empty.

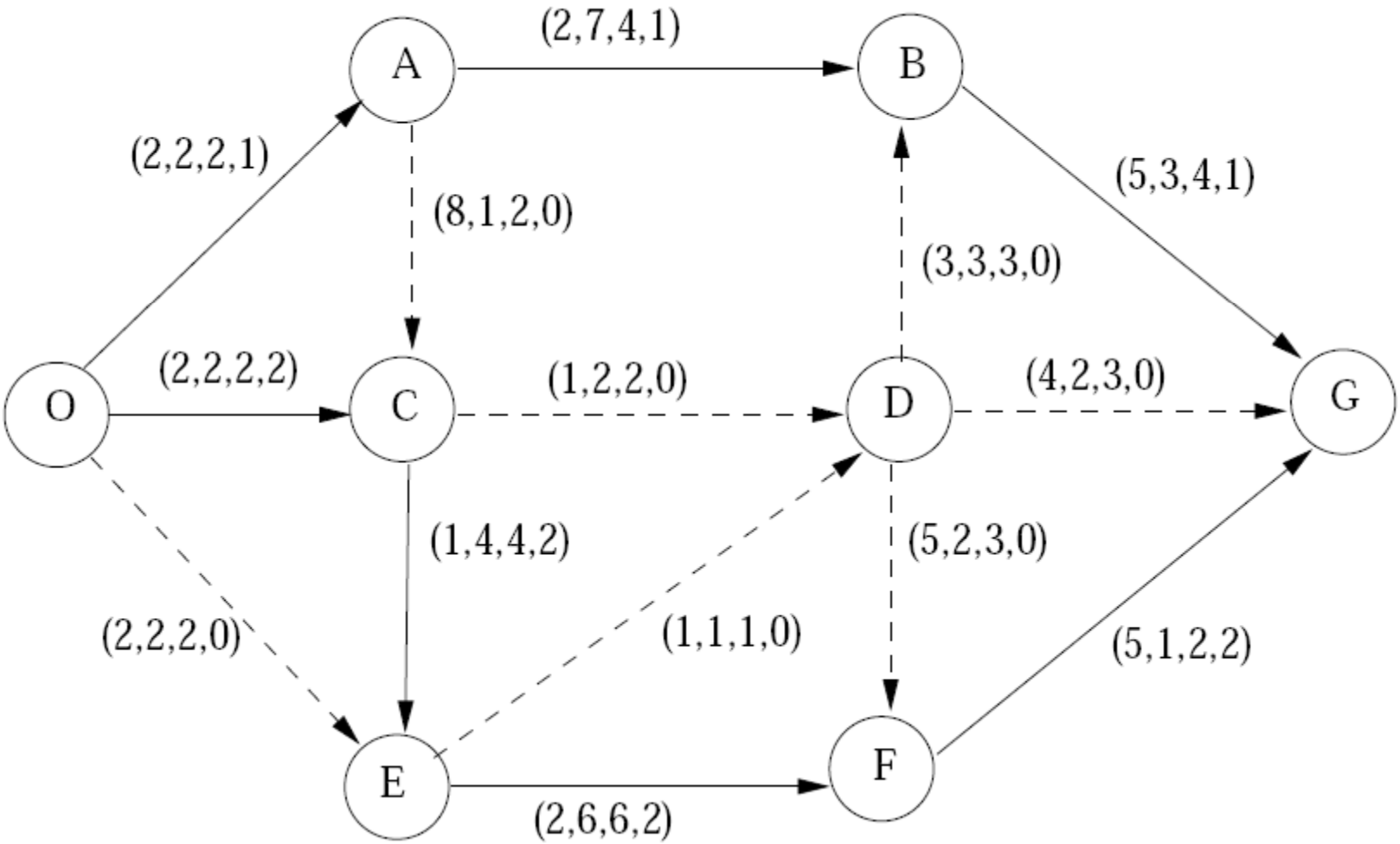
The γ-Residual Network

- The cost c_{ji}^{-}:

$$c_{ji}^{-} = \begin{cases} -\dfrac{\sum_{p\in\mathcal{P}} c_{ij}^{p} x_{ij}^{p}}{\sum_{p\in\mathcal{P}} x_{ij}^{p}}\gamma - f_{ij} & \text{if} \quad \sum_{p\in\mathcal{P}} x_{ij}^{p} = \gamma \\ -\dfrac{\sum_{p\in\mathcal{P}} c_{ij}^{p} x_{ij}^{p}}{\sum_{p\in\mathcal{P}} x_{ij}^{p}}\gamma & \text{if} \quad \sum_{p\in\mathcal{P}} x_{ij}^{p} > \gamma \end{cases}$$

The γ-Residual Network

- Example: (fixed cost, routing cost, capacity, flow)

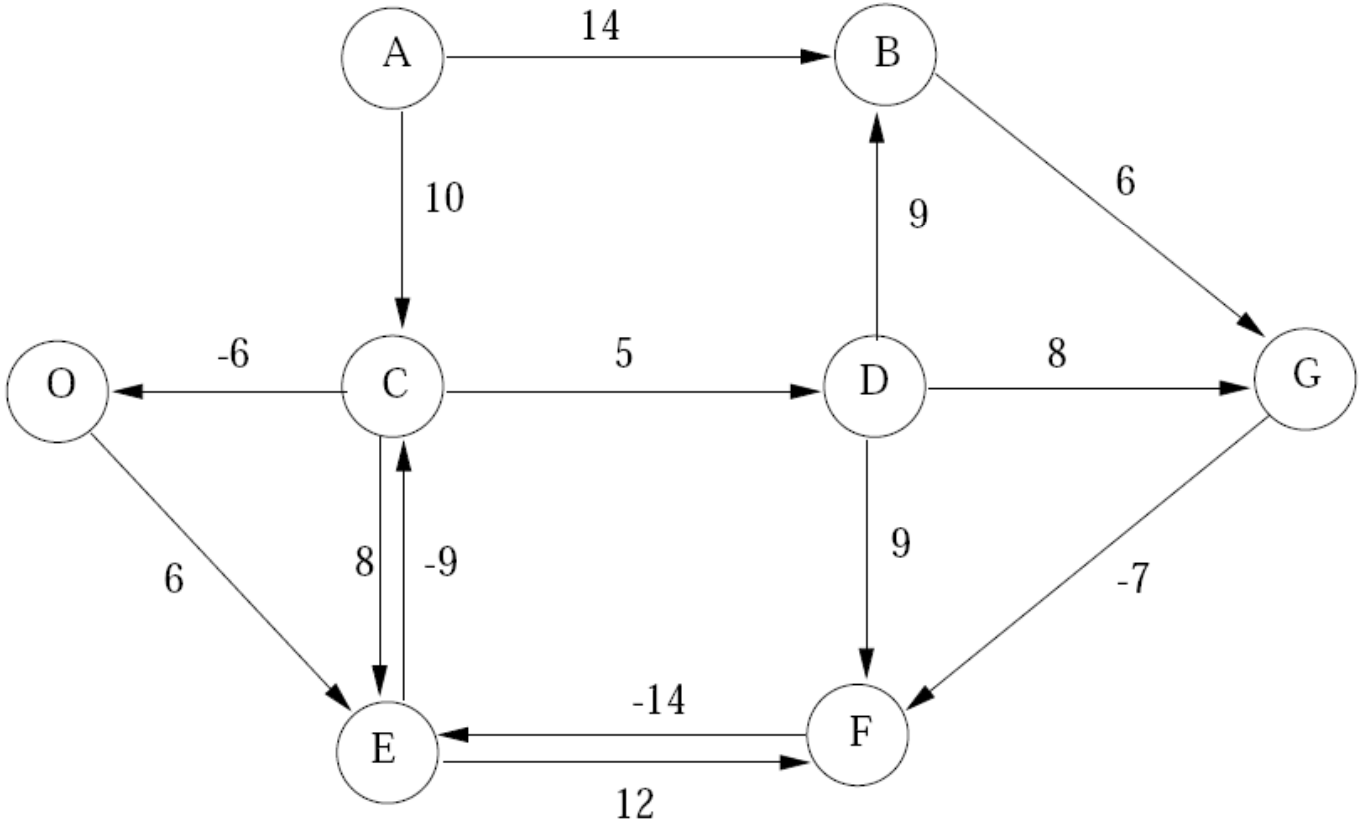


The γ-Residual Network

- Example:
  - In this example, arcs are labeled by the fixed cost, the routing cost (only one commodity is considered), the capacity, and the total flow.
  - We build a 2-residual network and look for the lowest cost cycle that passes through arc (C, D).

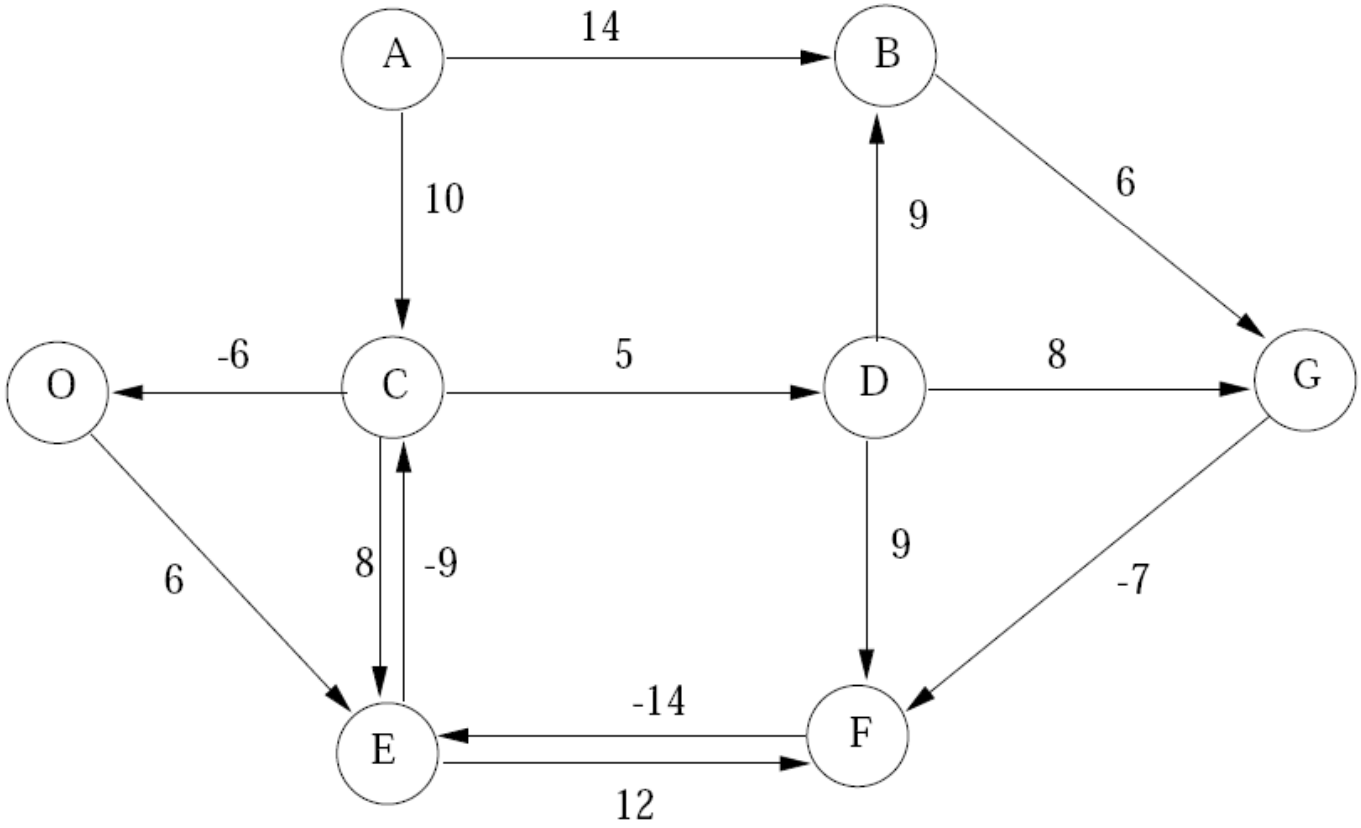
The γ-Residual Network

- One unit of flow passes on arc (O, A) and its residual capacity is equal to 1. Consequently, one can neither send nor take out 2 units of flow and, thus, there are no arcs between O and A.



The γ-Residual Network

- Arc (O, C) is saturated, so no more flow can be routed from O to C and the associated arc does not appear in

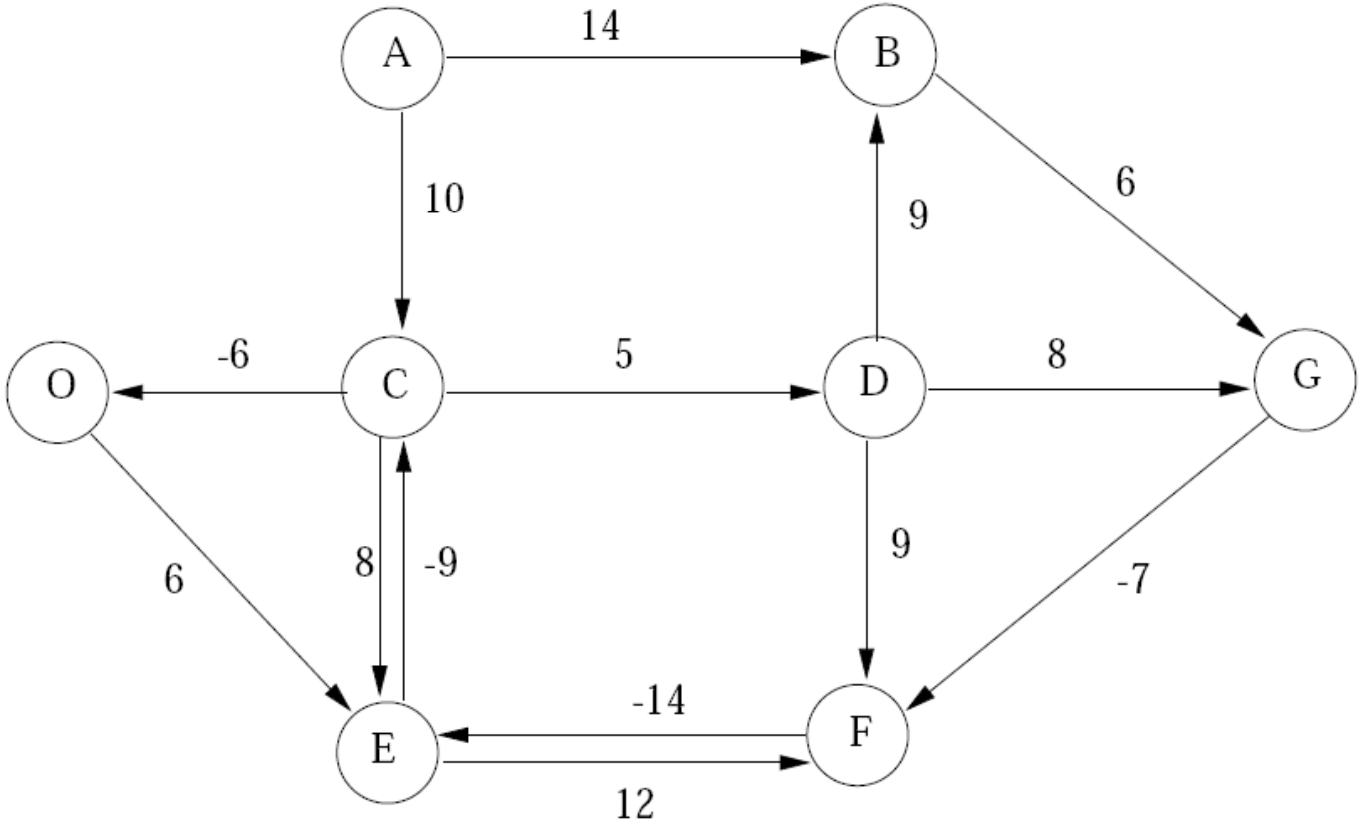


The γ-Residual Network

- On the other hand, reducing the flow of arc (*O*, *C*) by 2 units of flow leaves the arc empty, and thus the (*C*, *O*) arc is included in the 2-residual network with a cost -6 representing the savings in routing and fixed costs.
- The rest of the arcs are treated similarly.

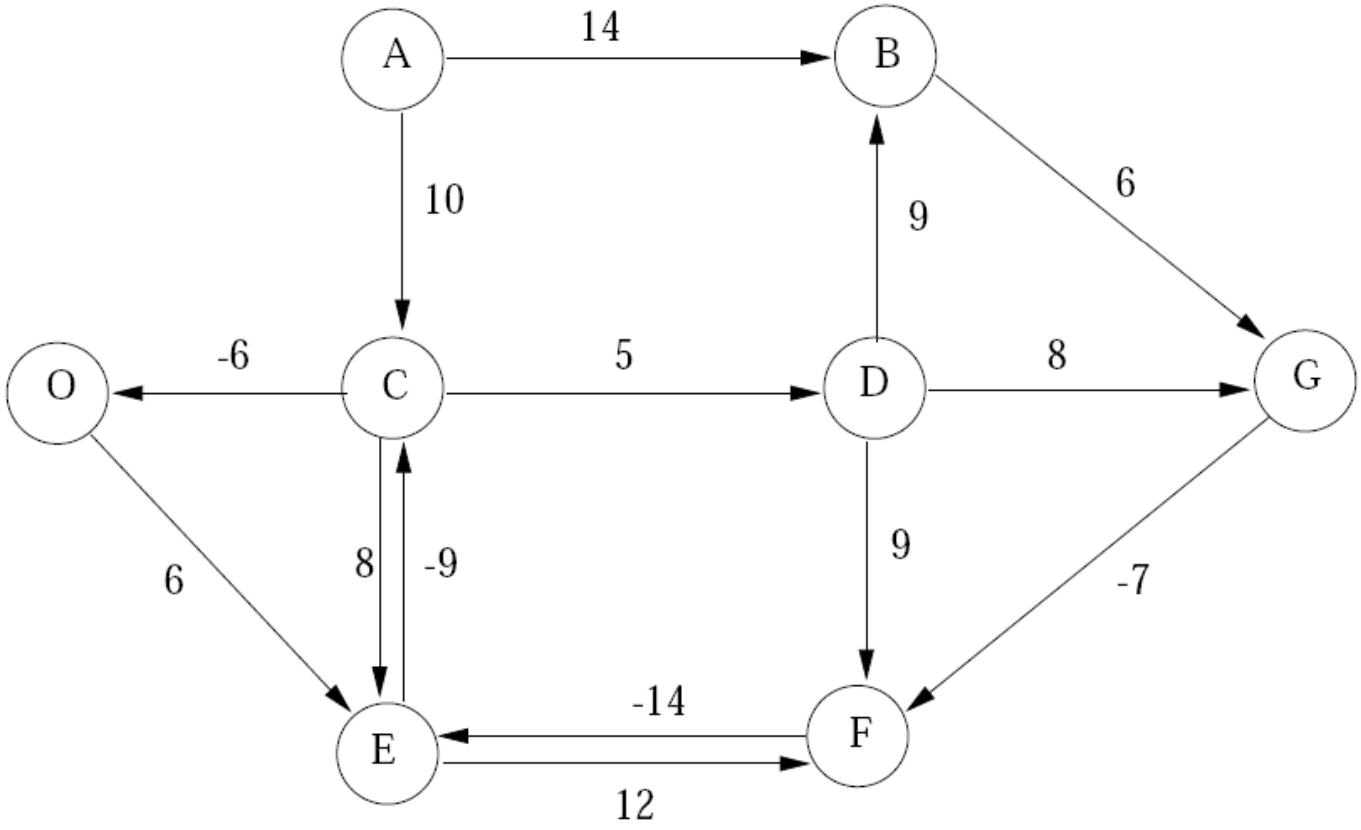
The γ-Residual Network

- Two directed cycles pass through arc (*C*, *D*) in the 2-residual network: (D, F), (F,E), (E,C), (C,D) and (D,G), (G, F), (F,E), (E,C), (C,D).



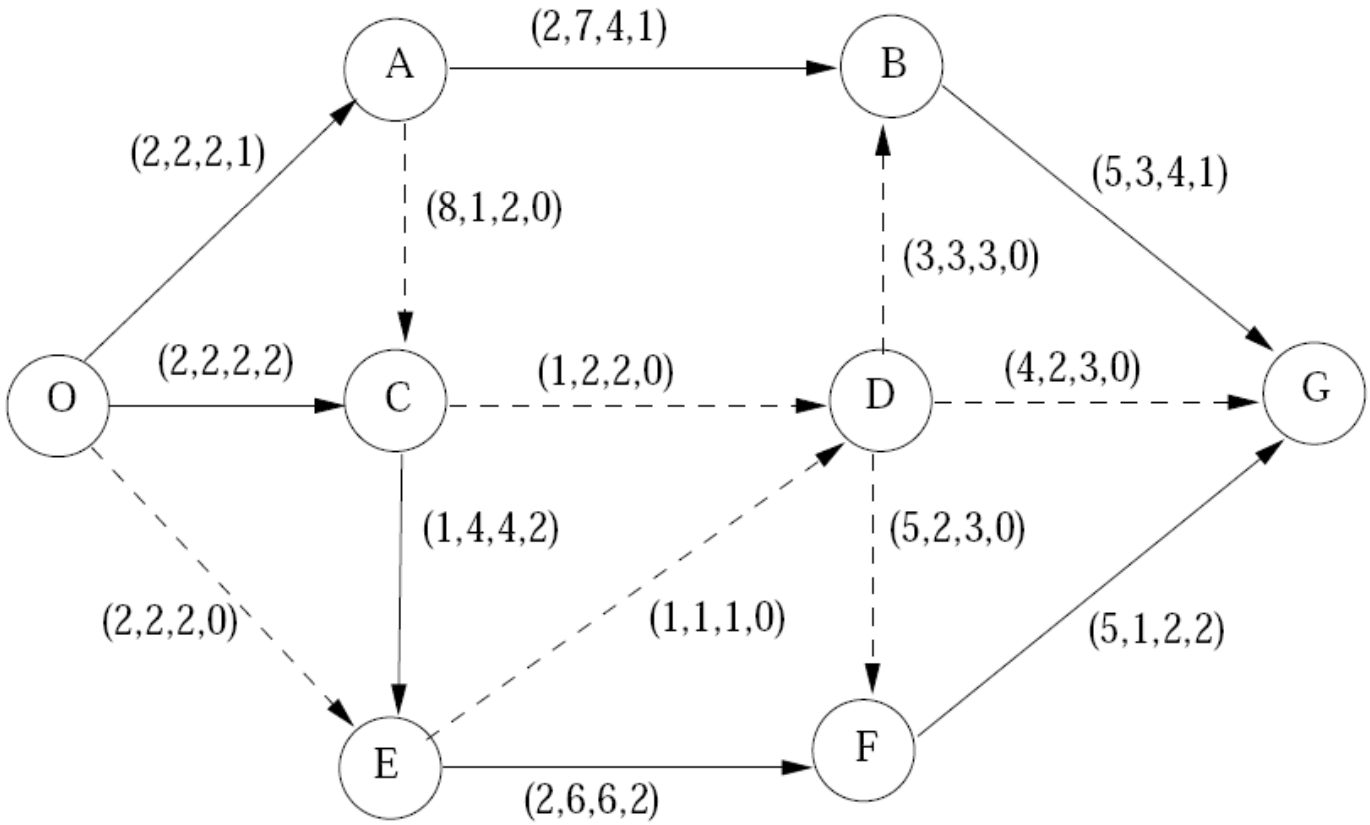
The γ-Residual Network

- The (D,G), (G, F), (F,E), (E,C), (C,D) has the lowest cost with value -17.



The γ-Residual Network

- This cost represents the net change in the objective function of the design formulation when moving 2 units of flow from path (C,E), (E, F), (F,G) to path (C,D), (D,G)



The γ-Residual Network

- This cost represents the net change in the objective function of the design formulation when moving 2 units of flow from path (C,E), (E, F), (F,G) to path (C,D), (D,G)
- This neighbour is thus reached by a complex move that involves opening two arcs ((C,D), (D,G)) and closing three others ((C,E), (E, F), (F,G)).

3.3 Heuristic to Identify Low Cost Cycles

Heuristic to Identify Low Cost Cycles

- The enumeration of all cycles passing through a given arc in a γ-residual network is expensive.
- Since one is interested in the lowest cost cycle only, one would like to find this particular cycle without enumerating all the others.
- A **network optimization procedure** is proposed for this purpose.

Heuristic to Identify Low Cost Cycles

- The main idea is to find for a given arc (i, j) the shortest path from j to i to complete the cycle.
- Yet, since the γ-residual network may contain negative cost directed cycles, finding a shortest path is NP-hard.
- Consequently, we do not attempt to solve the problem exactly.
- We rather develop a heuristic based on a modification of the shortest path label-correcting algorithm that avoids getting trapped in negative directed cycles and allows to find low-cost cycles.

4 Tabu Search with Cycle-based Neighbourhoods

Tabu Search with Cycle-based Neighbourhoods

- To evaluate the concepts introduced in the previous section, we developed a simple tabu search-based local search procedure that integrates two versions of the cycle-based neighbourhood:
  - First one that considers the flow of all commodities when determining cycles
  - Second one that refines the search by implementing moves resulting from the deviation of the flow of one commodity only at a time.

4.1 The Tabu Search Procedure

The Tabu Search Procedure

- Following an initialization phase, the tabu search procedure explores the *y* solution space
- At each iteration, the best non-tabu move is determined and implemented regardless whether it improves the overall solution or not.
- A short-term tabu memory is used to record characteristics of visited solutions to avoid cycling.
- When a particularly good solution is encountered, the search is intensified using a particular implementation of cycle-based neighbourhoods that consider the flow distribution of one commodity only.

The Tabu Search Procedure

- A solution is considered particularly good when it either improves the objective value of the best known solution or is within a pre-defined percentage of this value.
- The method terminates whenever a predefined stopping criterion (number of iterations, CPU time, etc.) is met.

The Tabu Search Procedure

- To select the **best move** in the neighbourhood of a given solution $\bar{y}$, the procedure first determines the set of the flow deviation values $\Gamma(\bar{y})$ as defined by:

$$\Gamma(\bar{y}) = \left\{ \sum_{p \in \mathcal{P}} x_{ij}^{p} > 0 \ : \ (i,j) \in \mathcal{A}(\bar{y}) \right\}$$

- To reduce the computational burden, the set of candidate arcs $\mathcal{C}$ is restricted to a random subset of closed arcs.

The Tabu Search Procedure

- Then, a γ-residual network is built for each value
$$\gamma \in \Gamma(\bar{y})$$
and a low-cost cycle is found for each arc $(i, j) \in C(\gamma)$.
- The lowest overall cycle is then considered as the **best local search move** from the current solution.

The Tabu Search Procedure

- The tabu search procedure:
- **Initialization**
  - Generate an initial feasible solution and initiate the *BestSolution* and *CurrentSolution* to its objective value.
- **Main local search loop**
  - While a stopping criterion is not met
  - Determine sets C , $\Gamma(\bar{y})$, and $C(\gamma)$ for the current solution $\bar{y}$.
  - Determine the best overall cycle
  - Move to the new solution by opening and closing the appropriate arcs,
  - Determine the solution value of the new solution by solving exactly the associated capacitated multicommodity network flow problem,

The Tabu Search Procedure

- **Main local search loop (cont.)**
  - Assign a tabu status to each complemented arc,
  - If the solution is infeasible, perform a restoration phase,
  - If,

$$\frac{CurrentSolution - BestSolution}{BestSolution} \leq IntensGap$$

  - the current solution is a “good” solution and an intensification phase is performed.
  - If CurrentSolution < BestSolution update BestSolution

The Tabu Search Procedure

- **Short-term tabu memory**
  - is used to prevent the search from cycling.
  - Thus, the arcs that are opened or closed receive a tabu status that forbids the reversal of the move for a given number of iterations.
  - The tabu memory is also updated following local search and intensification moves.

The Tabu Search Procedure

- **Initial Feasible Solution**
  - An initial feasible solution is produced by opening all arcs and solving the corresponding flow problem.
  - All arcs with flow are then considered open, while all unused arcs are closed.
  - An intensification phase is then performed until negative cycles are no longer detected for any commodity.

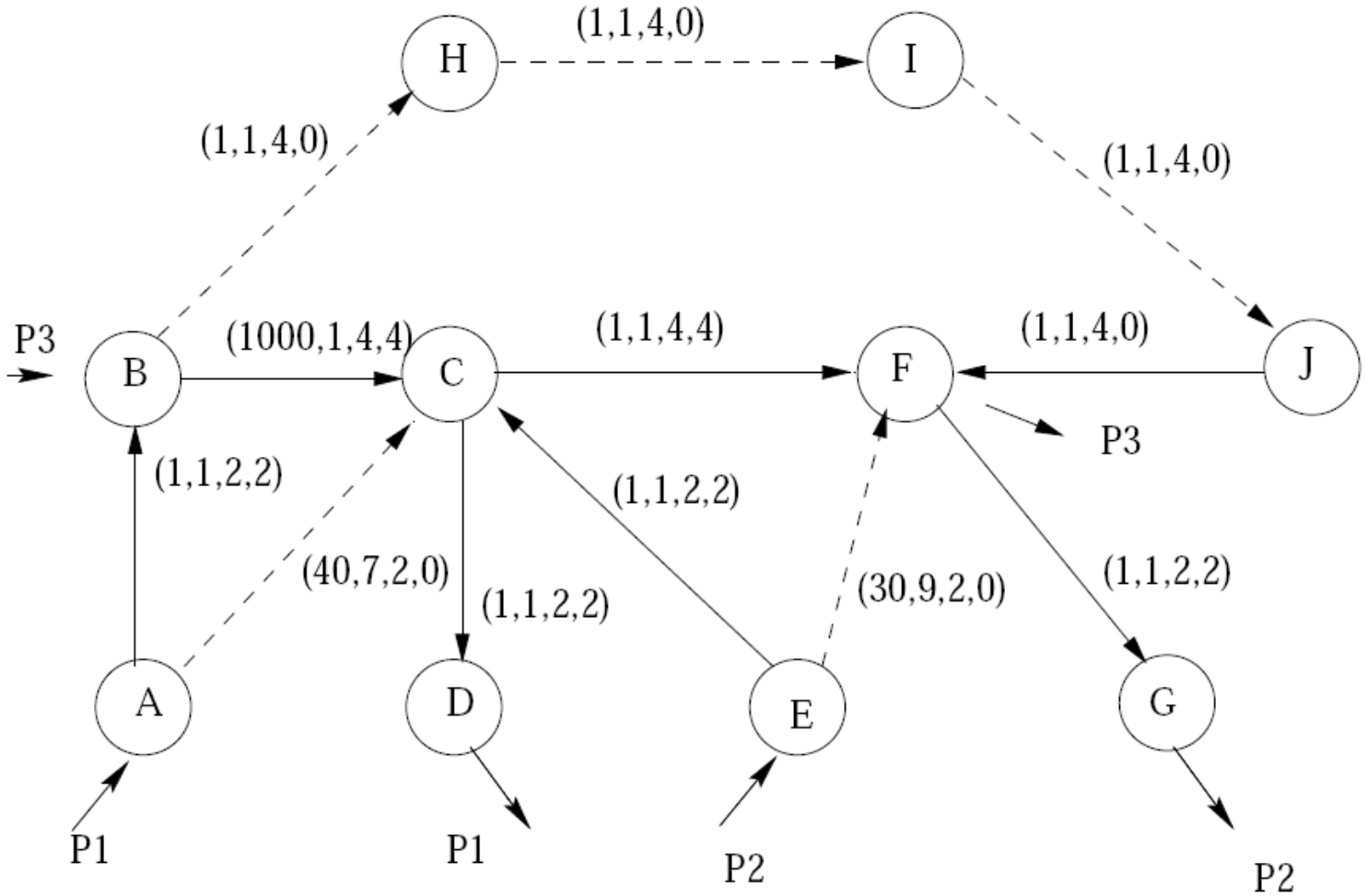
4.2 Restoration from Infeasible Moves

Restoration from Infeasible Moves

- The solution produced by a local search move might be infeasible.
- This follows from the fact that commodities are aggregated when local search moves are determined on γ-residual networks.
- Consequently, in the new network configuration that follows the closing and opening of arcs in the original network, commodities might follow different paths than the expected ones (i.e., those in the local search cycle-move).

Restoration from Infeasible Moves

- **Example:** consider this network



(fixed cost, routing cost, capacity, flow)

Restoration from Infeasible Moves

- **Example: (cont.)**
  - Three commodities p1, p2, and p3 share the network.
  - Let (x, y, z, t) denote respectively the fixed cost, routing cost, capacity
  - Flow on each arc and suppose that (A, D, 2), (E, G, 2) and (B, F, 2) represent the origin, destination, and demand of commodity p1, p2, and p3, respectively.
  - The current solution routes 2 units of flow of commodity p1 on path (A, B), (B, C), (C, D), 2 units of flow of commodity p2 on path (E,C), (C, F), (F,G), and 2 units of flow of commodity p3 on path (B,C), (C, F).
  - The objective function value associated to the current solution equals 1021.

Restoration from Infeasible Moves

- **Example: (cont.)**
  - In this example

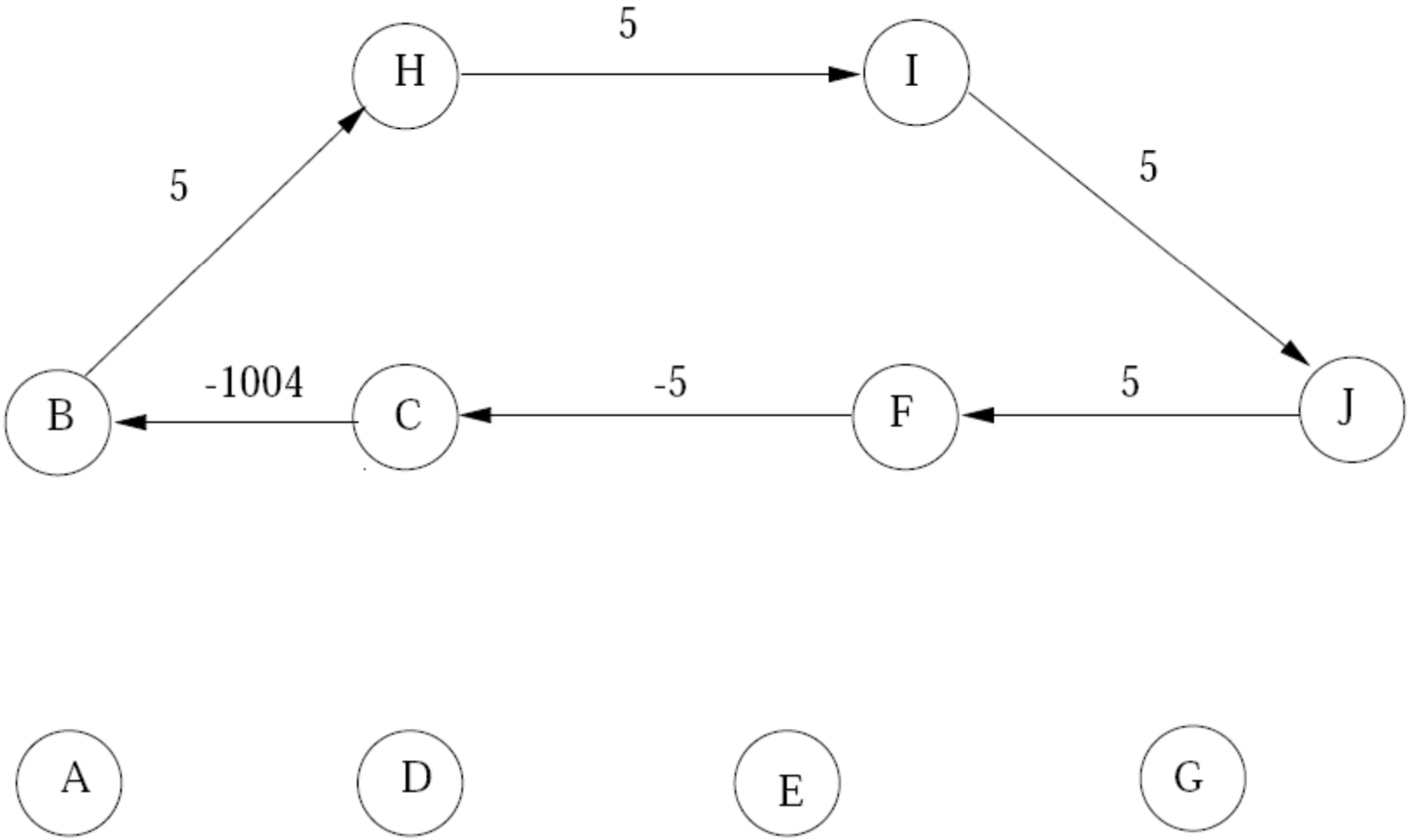
$$\Gamma(\bar{y}) = \left\{ \sum_{p \in \mathcal{P}} x_{ij}^{p} > 0 \ : \ (i,j) \in \mathcal{A}(\bar{y}) \right\}$$

$$\Gamma = \{2, 4\}$$

  - Assume all closed arcs are in C and an empty tabu list.

Restoration from Infeasible Moves

- The lowest cost cycle in the 4-residual network shows an improvement of -989.

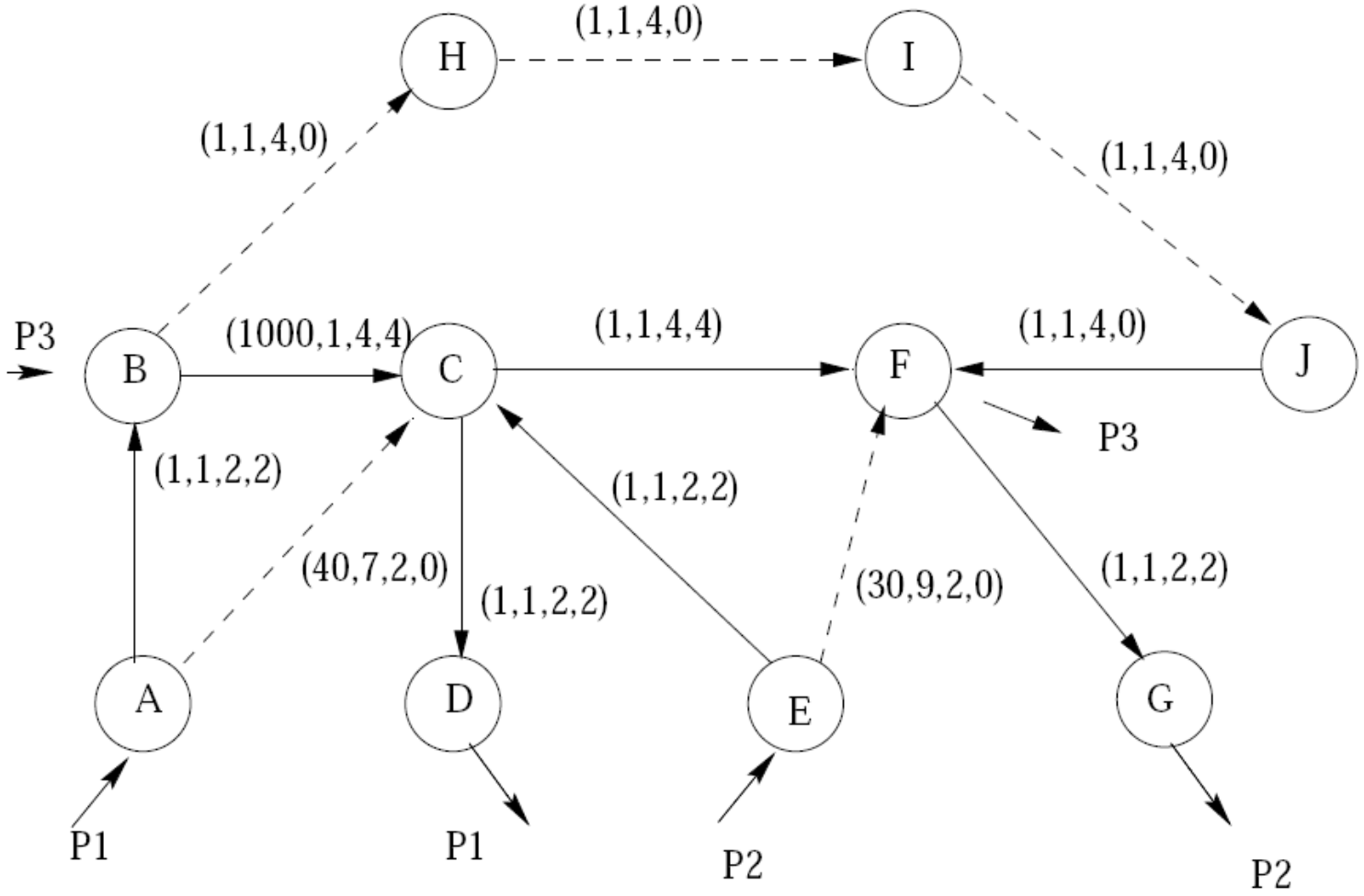


Restoration from Infeasible Moves

- **Example: (cont.)**
  - Consequently, the procedure would retain the cycle (B, H), (H, I), (I, J), (J, F), (F, C), (C, B) and would move to the best neighbour
    - by opening arcs (B, H), (H, I), (I, J), and (J, F) and
    - closing the arcs without flow (B, C) and (C, F).
  - The resulting network is infeasible since the demands of commodities p1 and p2 may no longer be satisfied.

Restoration from Infeasible Moves

- **Example: (cont.)**



Restoration from Infeasible Moves

- To detect infeasible solutions, artificial arcs between the origin and destination nodes of each commodity may be added to the network.
- These arcs receive capacities equal to the demand of the associated commodity and **high routing costs** to ensure that an artificial arc will bear flow only if the solution is infeasible.

Restoration from Infeasible Moves

- When the solution produced by a local search move is infeasible, a **restoration phase** is undertaken in order to reach a solution in the feasible domain.
- This phase is similar to the neighbourhood local search, except that the set C is made exclusively of artificial arcs with a positive flow, while the set Γ is restricted to the flow of commodities routed on these artificial arcs.

4.3 Intensification

Intensification

- The intensification phase is called each time a local search move yields a “good” solution that is, a solution that improves the best overall solution or is close to it.
- This phase searches to improve the solution further by iteratively modifying the flow distribution of one commodity only.

Intensification

- Adoption of the neighbourhood for the intensification phase:
  - Moves are detected by letting the flow of one commodity move around negative directed cycles while the flow of the other commodities is kept fixed.
  - The set Γ is instantiated for each commodity (denoted Γ^p) to contain the flow of the particular commodity that exist on each arc of the network.
  - Moreover, only improving moves are accepted during this phase to ensure that the chosen neighbour will always yield a feasible solution better than the current one.

Intensification

- **The intensification phase:**
  - Repeat until no negative cycle is detected in any γ-residual network
  - For each commodity $p \in \mathcal{P}$ and flow value $\gamma \in \Gamma^p$
    - Build the γ -residual network
    - Find a low cost cycle for each arc $(i, j) \in C(\gamma)$, $\gamma \in \Gamma^p$
    - Select the best overall cycle;
    - Denote γ_{best} the associated volume of deviated flow;

Intensification

- **The intensification phase:**
  - If the selected cycle is improving (negative cost)
    - 1. Update the solution:
      - Modify the flow around the selected cycle by a quantity γ_{best} ;
      - Open/close appropriate arcs;
      - Update the cost of the current solution by adding the cost of the cycle (without solving the CMCF).
    - 2. Assign a Tabu status to each complemented arc.
  - Solve exactly the CMCF associated to the current design.

5 Computational Results

Computational Results

- To ensure meaningful comparisons, we experiment on the same two sets of problem instances also used by Crainic, Gendreau, and Farvolden (2000).
- The computer code is written in C++.
- The exact evaluation of the capacitated multicommodity network flow problems is done using the linear programming solver of CPLEX 6.5.
- Unless indicated otherwise, tests were conducted on a Sun Ultra-60/2300 workstation with 2 Gigabyte of RAM, operating under Solaris 2.6.
- Computing times are reported in seconds.

5.1 Calibration

Calibration

- Key parameters of the tabu search meta-heuristic:
  - The size of the neighbourhood set explored at each iteration.
    - This size depends on the dimension of the candidate set *C* that is, on the percentage of closed arcs randomly selected.
    - Two values, 50% and 70% of the closed arcs, have been tested.
  - The length of the tabu tenure of arcs opened and closed following a local search or an intensification move.
    - Four values 1, 2, 3, and 5 were considered initially.
    - The 1 and 5 values were rapidly dropped, however, since cycling was observed for a tabu tenure length of 1, while for a value equal to 5 the quality of the solutions started to decrease.
  - The threshold used to determine a "good" solution:
    - IntensGap equal to 7%, 9%, and 11% of relative improvement.

Calibration

- Calibration experiments were conducted the same problem instances used by Crainic, Gendreau, and Farvolden (2000) to calibrate their path-based tabu search meta-heuristic.
- The 10 problems cover network sizes from 100 to 700 design arcs and from 10 to 400 commodities.
- They also display relatively high fixed costs compared to routing costs and tightly capacitated.
- Moreover, since a random number generator controls the selection of arcs in C, each run was repeated 3 times.

Calibration

- The impact of the random seed used was not major, but large enough to make it unsafe to draw conclusions from a single run.

Calibration

- Each parameter combination was ranked for each problem instance
  - according to the average gap (over 3 runs) relative to the best known solution
  - that of the branch-and-bound procedure of CPLEX 6.5, when available, or that obtained by Crainic, Gendreau, and Farvolden 2000, otherwise

Calibration

- A score of 10, 9, 8, .. , 2, 1 is assigned to each of the first ten places, respectively.
  - The performance of each parameter setting is then aggregated over all runs
  - Scores are summed up, while average gaps and CPU solution times are averaged.
  - Over the set of 360 runs: 10 problems tested 3 times for 12 parameter settings.

Calibration

- Parameter Setting Performances

Parameter Setting	Average Gap	Average CPU	Score
50%, 2, 0.09	1.93%	10814.34	68
70%, 2, 0.11	2.07%	13068.72	68
70%, 2, 0.09	2.09%	13011.97	63
70%, 2, 0.07	1.84%	13168.14	62
70%, 3, 0.07	1.96%	12427.75	59
50%, 3, 0.07	2.27%	10984.97	46
50%, 2, 0.07	2.30%	11480.44	45
70%, 3, 0.09	2.62%	12826.49	40
70%, 3, 0.11	2.95%	13528.20	35
50%, 3, 0.11	2.83%	11530.40	26
50%, 2, 0.11	2.72%	10929.73	24
50%, 3, 0.09	2.81%	11172.91	14

Calibration

- The parameter setting performances table
  - displays for each parameter combination these aggregated results
  - The first column holds the parameter setting, the second and third columns present the global average gaps and CPU times, respectively, while the last column displays the total score.
  - The final choice was then made on the other performance measures and the parameter combination C=50%, tabu tenure= 2, IntensGap= 9% was selected since it offers the lowest global average gap and CPU time.

Calibration

- To complete the calibration phase, we examined if the impact of the **intensification phase** and the **trim procedure** on the solution quality.
- **Intensification phase**
  - Recall that the **intensification phase** is based on the idea of focusing the search on promising regions, signaled by solutions that either improve the overall best solution or are close to this value, in order to avoid bypassing solutions of good quality.

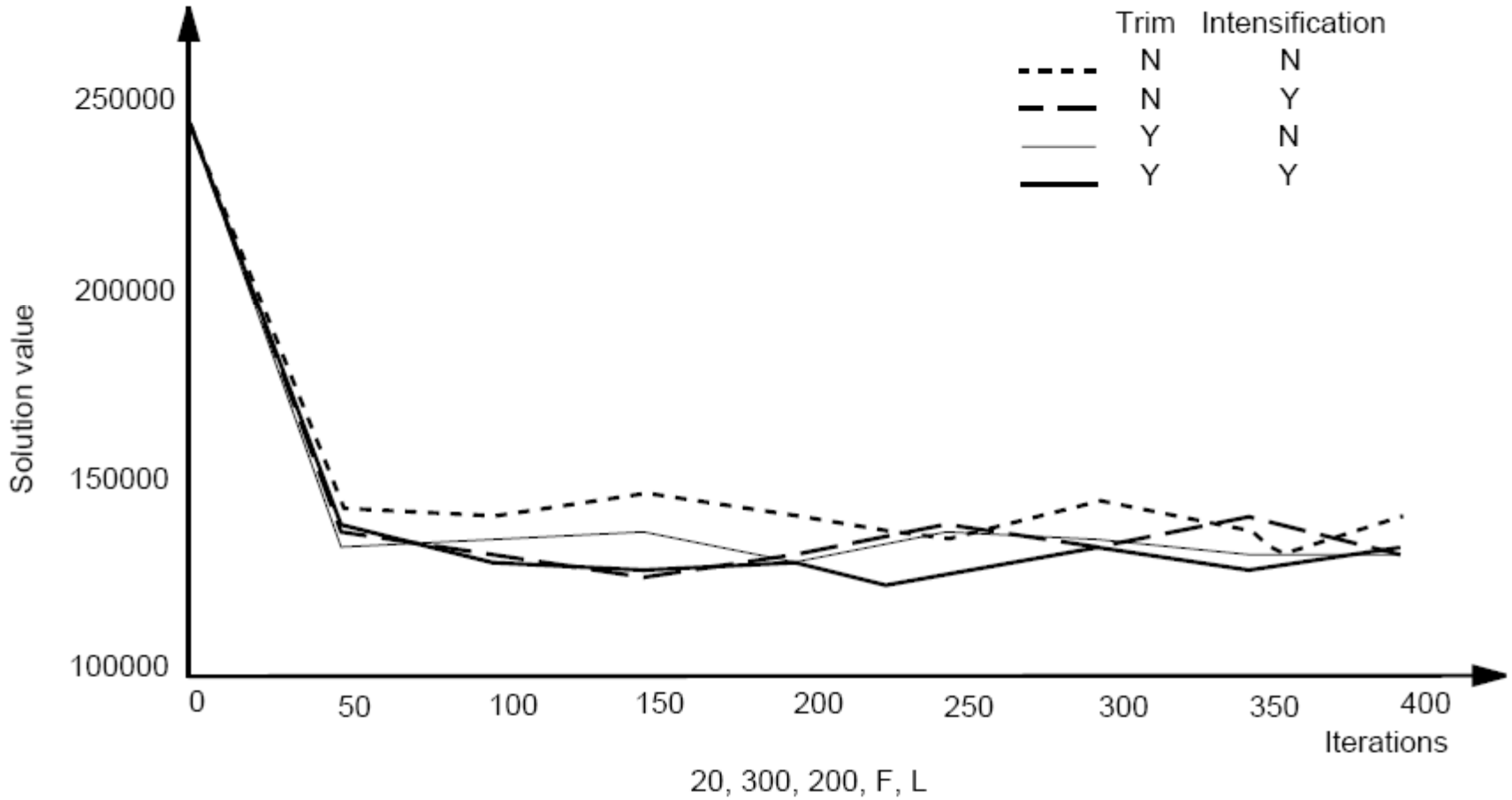
Calibration

- **Trim procedure**
  - The exact evaluation of a new configuration performed by solving the associated minimum cost network flow problem, may result in arcs that are open but do not carry flow.
  - The **trim procedure** consists in closing these arcs.
  - This reduces the solution value, but also modifies the basic move definition and impacts the search trajectory.

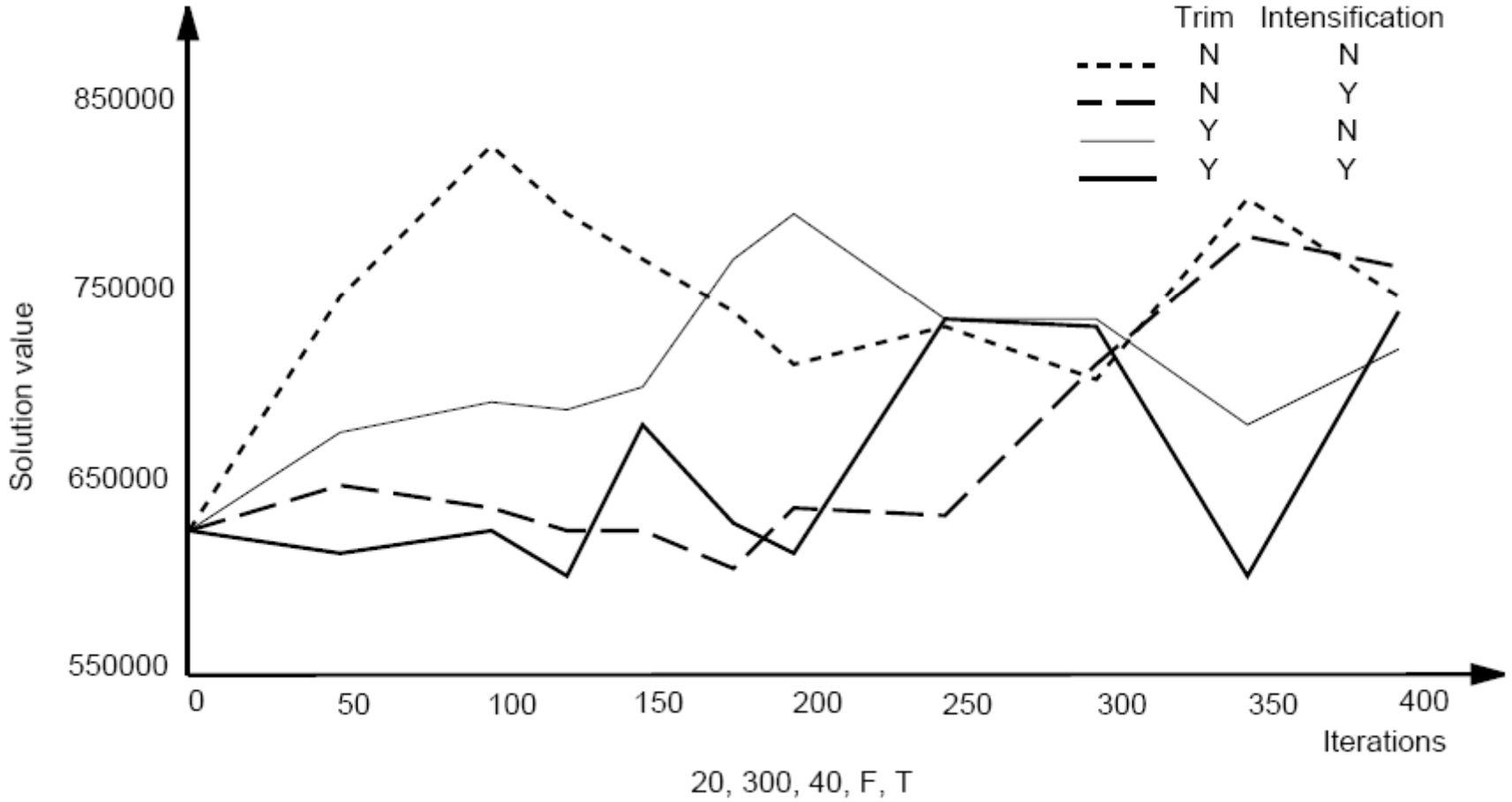
Calibration

- We investigated by running five problems using all four possible combinations of including or not the two procedures.
- The results indicate that including the intensification and trim procedures is beneficial for quality of the search.
- The following figure illustrates the behaviour of the four versions of the tabu search procedure on two problem instances,
  - 20,300,200,F,L and 20,300,40,F,T (identified by the number of nodes, design arcs, commodities, a letter indicating high fixed costs, and rather loose and tight capacities, respectively).

Calibration



Calibration



5.2 Result Analysis

Result Analysis

- To evaluate the performance of the tabu search algorithm proposed in this paper, we compare its output to
  - the optimal solution obtained using the branch-and-bound algorithm of CPLEX 6.5,
  - as well as to the results of the TABU-PATH procedure presented by Crainic, Gendreau, and Farvolden (2000)

Result Analysis

- The same two data sets used by Crainic, Gendreau, and Farvolden (2000) are also used in this paper.
- Problems in both sets are
  - general transshipment networks with no parallel arcs,
  - single origin-destination pair for each commodity,
  - and unique but arc-specific commodity routing costs.
  - Problem instances have been generated to offer for each network size a variety of fixed to routing cost and capacity to demand ratios.

Computational Results, C problems

PROB	OPT	TABU-PATH	GAP	TABU-CYCLE	GAP	TC(400)	GAP
25,100,10,V,L	14712	14712	0%	14712	0%	14712	0%
	(0.36)	(5.60)		(5.60)		(48.88)	
25,100,10,F,L	14941	15889	6.34%	14941	0%	14941	0%
	(53.64)	(8.37)		(8.37)		(53.77)	
25,100,10,F,T	49899	51654	3.52%	50767	1.74%	49899	0%
	(40.58)	(17.10)		(17.10)		(51.21)	
25,100,30,V,T	365272	365272	0%	365385	0.03%	365385	0.03%
	(16.62)	(16.57)		(16.57)		(223.67)	
25,100,30,F,L	37055	38804	4.72%	38679	4.38%	37583	1.42%
	(1727.96)	(33.01)		(33.01)		(215.38)	
25,100,30,F,T	85530	86445	1.07%	86296	0.90%	86296	0.90%
	(534.18)	(71.84)		(71.84)		(224.64)	
20,230,40,V,L	423848	425046	0.28%	425091	0.29%	424778	0.22%
	(10.43)	(71.29)		(71.29)		(370.33)	
20,230,40,V,T	371475	371816	0.09%	372583	0.30%	371893	0.11%
	(52.37)	(90.28)		(90.28)		(435.61)	
20,230,40,F,T	643036	644172	0.18%	646786	0.58%	645812	0.43%
	(671.76)	(121.79)		(121.79)		(423.32)	
20,300,40,V,L	429398	429912	0.12%	429837	0.10%	429535	0.03%
	(3.76)	(71.05)		(71.05)		(611.5)	

Computational Results, C problems

20,300,40,F,L	586077	589190	0.53%	593323	1.24%	593322	1.24%
	(145.55)	(113.44)		(113.44)		(581.94)	
20,300,40,V,T	464509	464509	0%	466525	0.43%	464724	0.05%
	(83.88)	(145.33)		(145.33)		(589.64)	
20,300,40,F,T	604198	606364	0.36%	609285	0.84%	607100	0.48%
	(59.88)	(123.42)		(123.42)		(560.38)	
20,230,200,V,L	94386	122592	29.88%	103930	10.11%	98995	4.88%
	(t)	(504.50)		(504.50)		(2663.24)	
20,230,200,F,L	141737.4	188590	33.06%	154404	8.94%	146535	3.38%
	(t)	(491.63)		(491.63)		(2718.31)	
20,230,200,V,T	97914	118057	20.57%	108274	10.58%	104752	6.98%
	(t)	(548.36)		(548.36)		(2565.71)	
20,230,200,F,T	137271	182829	33.19%	153455	11.79%	147385	7.37%
	(t)	(889.69)		(889.69)		(3120.14)	
20,300,200,V,L	74972.4	88398	17.91%	81628	8.88%	80819	7.80%
	(t)	(982.21)		(982.21)		(4086.83)	
20,300,200,F,L	117306	151317	28.99%	129600	10.48%	123347	5.15%
	(t)	(1316.75)		(1316.75)		(4367.88)	
20,300,200,V,T	74991	82724	10.31%	80510	7.36%	79619	6.17%
	(t)	(938.29)		(938.29)		(3807.93)	
20,300,200,F,T	108252	135593	25.26%	122547	13.21%	114484	5.76%
	(t)	(1065.88)		(1065.88)		(4657.54)	

Computational Results, C problems

PROB	OPT	TABU-PATH	GAP	TABU-CYCLE	GAP	TC(400)	GAP
100,400,10,V,L	28423	28485	0.22%	28708	1.00%	28677	0.89%
	(84.81)	(32.66)		(32.66)		(336.34)	
100,400,10,F,L	24436	24912	1.95%	24576	0.57%	23949	- 1.99%
	(t)	(33.00)		(33.00)		(306.79)	
100,400,10,F,T	66364	71128	7.18%	68004	2.47%	67014	0.98%
	(t)	(81.23)		(81.23)		(626.46)	
100,400,30,V,T	385544	385185	- 0.09%	385508	- 0.01%	385508	- 0.01%
	(t)	(277.50)		(277.50)		(1975.34)	
100,400,30,F,L	50496	58773	16.39%	55401	9.71%	51552	2.09%
	(t)	(100.16)		(100.16)		(1300.56)	
100,400,30,F,T	141278	149282	5.67%	146455	3.66%	145144	2.74%
	(t)	(215.71)		(215.71)		(1869.98)	
30,520,100,V,L	53966	56426	4.56%	55358	2.58%	54958	1.84%
	(t)	(995.64)		(995.64)		(3355.96)	
30,520,100,F,L	95294	104117	9.26%	103747	8.87%	99586	4.50%
	(t)	(939.24)		(939.24)		(4032.35)	
30,520,100,V,T	52085	53288	2.31%	52985	1.73%	52985	1.73%
	(t)	(1218.52)		(1218.52)		(3481.08)	
30,520,100,F,T	98357	107894	9.70%	106877	8.66%	105523	7.29%
	(t)	(670.29)		(670.29)		(3927.35)	
30,700,100,V,L	47603	48984	2.90%	48824	2.56%	48398	1.67%
	(1736.05)	(1265.11)		(1265.11)		(4396.42)	

Computational Results, C problems

30,700,100,F,L	60525	65356	7.98%	63302	4.59%	62471	3.22%
	(t)	(1479.59)		(1479.59)		(4755.01)	
30,700,100,V,T	45944.5	47083	2.48%	47025	2.35%	47025	2.35%
	(t)	(2426.02)		(2426.02)		(4560.11)	
30,700,100,F,T	55709	58804	5.56%	57886	3.91%	57886	3.91%
	(t)	(1735.72)		(1735.72)		(4866.11)	
30,520,400,V,L	112997.5	125831	11.36%	122048	8.01%	120652	6.77%
	(t)	(5789.27)		(5789.27)		(36530.8)	
30,520,400,F,L	X	177409	-	167837	- 5.40%	161098	- 9.19%
	(t)	(6406.62)		(6406.62)		(429296)	
30,520,400,V,T	X	125518	-	121603	- 3.12%	121588	- 3.13%
	(t)	(6522.23)		(6522.23)		(28214)	
30,520,400,F,T	X	174526	-	171641	- 1.65%	167939	- 3.77%
	(t)	(8415.24)		(8415.24)		(40010.9)	
30,700,400,V,L	X	110000	-	108029	- 1.79%	106777	- 2.93%
	(t)	(12636.2)		(12636.2)		(24816.8)	
30,700,400,F,L	X	165484	-	154215	- 6.81%	148950	- 9.99%
	(t)	(11367.70)		(11367.70)		(69540.1)	
30,700,400,V,T	X	103768	-	102468	- 1.25%	101672	- 2.02%
	(t)	(15879.50)		(15879.50)		(34974.9)	
30,700,400,F,T	X	150919	-	148243	- 1.77%	142778	- 5.39%
	(t)	(11660.40)		(11660.40)		(51877.9)	

Result Analysis

- The problems are identified in the first column by the
  - number of nodes, arcs, and commodities,
  - as well as two letters summarizing the fixed cost and capacity information:
    - a relatively high or low fixed cost relative to the routing cost is signaled by the letter F or L, respectively,
    - while letters T and L indicate respectively if the problem is tightly or somewhat loosely capacitated compared to the total demand.

Result Analysis

- The **opt column**
  - corresponds to the solution of the the branch-and-bound algorithm solved using CPLEX 6.5 on one 400MHz processor of a 64-CPU Sun Enterprise 10000 with 64 Gigabyte of RAM, operating under Solaris 2.7.
  - A limit of 10 hours was imposed.
  - An X indicates that the procedure has failed to produce a feasible solution within this time limit, while a t indicates that the procedure stopped due to a time limit condition.

Result Analysis

- The **Column TABU-PATH**
  - holds the results found by the path-based tabu search of Crainic, Gendreau, and Farvolden (2000) on the same workstations.
  - For a valid comparison between the two approaches, two results are displayed for the cycle-based tabu search meta-heuristic.
- The **tabu-cycle column**
  - displays solutions found by our approach using the same CPU time reported for the path-based method.

Result Analysis

- **The column TC(400)**
  - To illustrate and analyze the behaviour of the algorithm we propose when longer runs are performed, **column tc(400)** displays computational results after 400 iterations.
- **CPU time**
  - For all methods, the figures in parentheses represent total computational time in CPU seconds on the appropriate computers.

Result Analysis

- **GAP relative**
  - For the three meta-heuristic results, the next column displays the optimality gap relative to the branch-and-bound solution.
  - When branch-and-bound has failed to identify a feasible solution, the gap relative to the path-based tabu search is displayed instead.

Result Analysis

- **Conclusion #1:**
  - is that fixed cost, capacitated, multicommodity network design problems are indeed difficult to solve, as indicated by the performance of a state-of-the-art mixed integer programming solver.
- **Conclusion #2:**
  - tabu-cycle outperforms tabu-path for an equivalent computational effort in almost all experiments.
  - Thus, out of 43 problem instances, only for 9, relatively small, problems are the solutions of tabu-path better than those of tabu-cycle.

Result Analysis

- **Conclusion #3:**
  - The superiority of tabu-cycle appears to be greater when fixed cost are high
  - the optimality gap of tabu-cycle for these problems is at most 14%, while it reaches some 33% for tabu-path.
- **Conclusion #4:**
  - The average percentage improvement of tabu-cycle compared to tabu-path is around 3.36%, with a maximum improvement of 18.13%.
  - These results are a first indication of the effectiveness of the cycle-based neighbourhood structures to generate good solutions for the CMND.

Result Analysis

- **Conclusion #5:**
  - The results displayed in Tables also support the hypothesis that our algorithm may identify higher quality solution given longer search times.
  - Using 400 iterations, improved solutions were found for 35 out of the 41 problems (2 solutions were already optimal).
  - Moreover, the benefits obtained from increasing the computational effort appear to become greater for larger instances.
  - Thus, for example, for problems with 200 commodities the maximum optimality gap is decreased from 13.21% to 7.80%.

Result Analysis

- Distribution of Relative Gaps

P.Set	X	OPT	IMP	(0-2%]	(2-4%]	(4-6%]	(6-8%]	(8-10%]	>10%
C	7	3	2	15	6	4	6	-	-
R	-	28	-	46	33	20	11	8	7

  - the distribution of the optimality gap relative to branch-and-bound for the two sets of problem instances, obtained by the cycle-based tabu search after 400 iterations.

Result Analysis

- **Distribution of Relative Gaps**
  - The first column identifies the problems set,
  - the second column indicates the number of problems where branch-and-bound did not find a feasible solution (in 10 hours),
  - the third indicates the number of optimal solutions found by the meta-heuristic,
  - the fourth displays the number of problems where the heuristic solution is better than the one found by branch-and-bound after 10 hours of computation.
  - The next six columns correspond to the indicated gap intervals.

Result Analysis

- **Distribution of Relative Gaps**
  - For problems in set C, the optimality gap never exceed 8% and is often much smaller than 2%, for an average within 2.10% of the best solutions found by branch-and-bound.

The End