II.1 Formulations, Applications and Approaches

Multicommodity Flow problems generalize the well-known single-commodity flow problems in that flows of different nature (*commodities*) co-exist in a given network: no commodity ever transforms into another commodity, so that each one has its own flow conservation equations, but using the same network they compete for the resources represented by the arc capacities. Multicommodity Flow problems share many characteristics of their single-commodity variants, most typically of being on the boundary between continuous and combinatorial problems, with the possibility of borrowing theoretical and algorithmic tools from both the sides: however, they do *not* inherit the integrality of the optimal solutions. Hence, most of the solution techniques developed over the years for *continuous* Multicommodity Flow problems are essentially based on their representation as (large-scale) linear or nonlinear problems, while *Integer* Multicommodity Flow problems, that retain most of the "combinatorial" flavour but are considered among the most difficult $\mathcal{NP}$-hard problems, call for the machinery of enumerative or polyhedral approaches.

Multicommodity Flow problems have attracted the researchers for several different reasons: from an applicative viewpoint, the Multicommodity Min-Cost Flow problem and its variants are (sub)models of a wide variety of transportation and scheduling problems, and their (repeated) solution is required within approaches to the original problem. Moreover, Multicommodity versions of the Max-Flow/Min-Cut relation are intimately related with $\mathcal{NP}$-hard graph partitioning problems, and have both theoretical and practical applications in the design of approximation algorithms for several problems arising e.g. in VLSI design. Finally, (MMCF) is the "archetype" of block-structured problems with "complicating" constraints, and has motivated many of the results for this broad and important class of optimization problems: for instance, the celebrated Dantzig-Wolfe decomposition approach was inspired by the work of Ford and Fulkerson on Multicommodity Max-Flow problems, but other (combinatorial or not) ε-approximation approaches to block-structured problems have been recently developed starting from (MMCF)-specific results.

In §II.1.1 and §II.1.2, the Min-Cost and Max-Flow formulations of Multicommodity Flow problems are discussed in details, with a stress on the characteristics that are especially relevant for the performances of the solution approaches; the other two paragraphs present bibliographic reviews respectively of some of the most important practical applications (§II.1.3) and of the main algorithmic approaches proposed so far (§II.1.4).

II.1.1 Multicommodity Min-Cost Flow formulations

Given a directed graph $G(\mathcal{N}, \mathcal{A})$, with n nodes and m arcs, and the set $\mathcal{K} = \{ 1 \dots k \}$, a *multicommodity flow* (or *multiflow*) on G is a vector $\boldsymbol{x} = [\, \boldsymbol{x}^1 \dots \boldsymbol{x}^k \,]$ of k distinct flow vectors $\boldsymbol{x}^h : \mathcal{A} \to \mathcal{R}^m$; with a little abuse of notation, "commodity" is used for indicating both $\boldsymbol{x}^h$ and its "name" h. The focus of Part II of the thesis is on the Linear Multicommodity Min-Cost Flow problem

$$\text{(MMCF)} \quad \begin{cases} \min \sum_{h \in \mathcal{K}} \sum_{(i,j) \in \mathcal{A}} c_{ij}^h x_{ij}^h & \\ \sum_{j:(i,j) \in \mathcal{A}} x_{ij}^h - \sum_{j:(j,i) \in \mathcal{A}} x_{ji}^h = b_i^h & \forall\ i \in \mathcal{N},\ \forall\ h \in \mathcal{K} \\ 0 \le x_{ij}^h \le u_{ij}^h & \forall\ (i,j) \in \mathcal{A},\ \forall\ h \in \mathcal{K} \\ \sum_{h \in \mathcal{K}} x_{ij}^h \le u_{ij} & \forall\ (i,j) \in \mathcal{A} \end{cases}$$

that requires to route the k commodities on G at minimal *total* (linear) *cost*, respecting the usual *node balance* constraints and *individual* (or *single-commodity*) capacity constraints $0 \le x_{ij}^h \le u_{ij}^h$ as well as *mutual* or *aggregate capacity* constraints $\Sigma_h x_{ij}^h \le u_{ij}$. When stated in matrix notation, using the *node-arc incidence matrix* $\boldsymbol{E}$ of G (Figure 1), the *node-arc formulation*

$$\text{(MMCF)} \quad \begin{cases} \min \Sigma_{h \in \mathcal{K}} \boldsymbol{c}^h \boldsymbol{x}^h & \\ \begin{bmatrix} \boldsymbol{E} & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{E} \\ \boldsymbol{I} & \cdots & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}^1 \\ \vdots \\ \boldsymbol{x}^k \end{bmatrix} \le \begin{bmatrix} \boldsymbol{b}^1 \\ \vdots \\ \boldsymbol{b}^k \\ \boldsymbol{u} \end{bmatrix} & \\ \boldsymbol{0} \le \boldsymbol{x}^h \le \boldsymbol{u}^h & \forall\ h \in \mathcal{K} \end{cases}$$

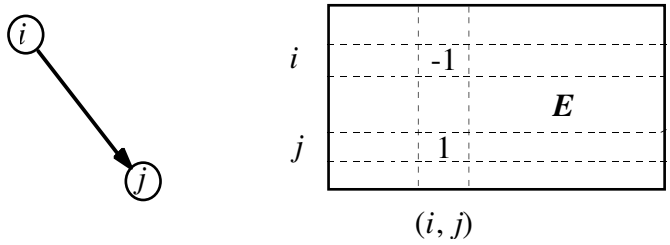

Figure 1: the node-arc incidence matrix

highlights the *block-structured* nature of the problem, i.e. that (MMCF) decomposes into k independent Min-Cost Flow subproblems if the mutual capacity (*linking*) constraints are relaxed: in this form, (MMCF) is a Linear Program with km variables and $kn + m$ (+ km) constraints. Although very sparse (each column has no more than three nonzeroes), such a (LP) becomes extremely large, even for small graphs, as the number of commodities increases; furthermore, the nice "combinatorial structure" of the (MCF) subproblems is in a large part destroyed by the mutual capacity constraints, since the optimal solution of the problem is *not* in general integral even though the data are. This is shown by the 3-commodities example of Figure 2, where one unit of commodity h has to be shipped from s_h to t_h on a (planar) graph with unit mutual capacities only in the "inner triangle": clearly, the optimal solution consists in sending 1/2 unit of each commodity along the path formed by three arcs with cost 1, and the other 1/2 unit along the single directed arc with cost 4.

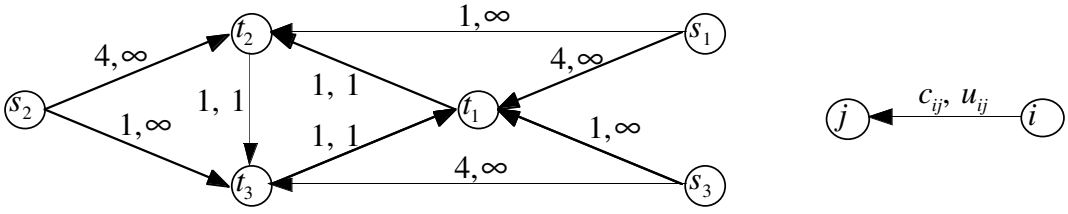

Figure 2: an instance with integral data and fractional optimal solution

Integrality is not the only property that is lost by passing from single-commodity to Multicommodity Flows: most notably, the celebrated Max-Flow/Min-Cut theorem is true only in some very special cases, as discussed in more details in the next paragraphs.

Note that, in Figure 2, no arc has an individual upper bound and, for each commodity $h \in \mathcal{K}$, $d_h > 0$ units of flow have to be shipped from the unique source s_h to the unique sink t_h: in such a case, the k independent (MCF)s obtained by relaxing the mutual capacity constraints are Shortest Path problems. This suggests a "path-based" formulation of the problem, since, being $P(h)$ the set of all the simple path from s_h to t_h, it is possible to define the function $h = \mathcal{K}(p)$ that maps each path p in $P = \bigcup_{h \in \mathcal{K}} P(h)$ to a commodity $h \in \mathcal{K}$ and write the problem as

$$\text{(MMCF)} \qquad \begin{cases} \min \sum_{p \in P} c_p x_p & \\ \sum_{p \in P(h)} x_f = d_h & \forall\ h \in \mathcal{K} \\ \sum_{p:\,(i,j) \in f} x_p \le u_{ij} & \forall\ (i,j) \in \mathcal{A} \\ x_p \ge 0 & \forall\ p \in P \end{cases}$$

where the variables x_p are now associated to (simple) paths rather than to arcs, and c_p is the cost of sending one unit of flow from $s_{\mathcal{K}(p)}$ to $t_{\mathcal{K}(p)}$ along p. Note that the vector of the variables can still be partitioned according to the commodity, i.e. $\boldsymbol{x} = [\, \boldsymbol{x}^1 \ \ldots\ \boldsymbol{x}^k \,]$, so that the *arc-path* formulation can be restated in matrix form as

$$\text{(MMCF)} \qquad \begin{cases} \min \boldsymbol{c}\boldsymbol{x} & \\ \begin{bmatrix} \boldsymbol{e}^T & & \boldsymbol{0} \\ & \ddots & \\ \boldsymbol{0} & & \boldsymbol{e}^T \\ \multicolumn{3}{c}{\boldsymbol{F}} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}^1 \\ \vdots \\ \boldsymbol{x}^k \end{bmatrix} = \begin{bmatrix} d^1 \\ \vdots \\ d^k \\ \multicolumn{1}{c}{\boldsymbol{u}} \end{bmatrix} & \\ \boldsymbol{x} \ge \boldsymbol{0} \end{cases}$$

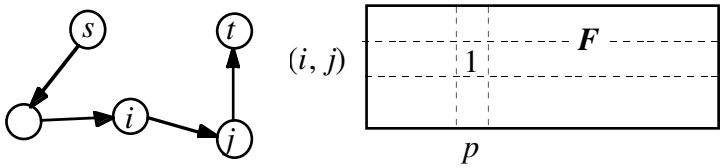

Figure 3: the arc-path incidence matrix

where $\boldsymbol{F}$ is the *arc-path* (or arc-chain) *incidence matrix* of G (Figure 3): this (LP) has exponentially many variables, but it has only $m + k$ constraints. This implies that an optimal solution exists (if any) that has at most $m + k$ strictly nonzero variables, i.e. no more than $m + k$ different paths are needed to deliver all the requested flow: hence, only a very small fraction of the variables will be needed in the optimal solution, clearly suggesting the use of column generation techniques.

The two formulations are essentially equivalent, but there are some differences that make one or the other more convenient for different purposes:

- the arc-path formulation is typically used when no individual upper bounds on the arcs are present: such bounds could be introduced as

  $$\sum_{p \in P(h):\,(i,j) \in p} x_p \le u_{ij}^h$$

  but this would increase the number of rows, possibly cancelling out the main advantage of this formulation; however, the special case $u_{ij}^h = 0$ is naturally accommodated (also in practice) within the arc-path formulation by deleting (i, j) from the graph $G^h(\mathcal{N}, \mathcal{A}^h)$ for commodity h;
- the arc-path formulation requires that all commodities have a single Origin/Destination pair, but it

can be extended as long as each path can be univocally assigned to a commodity: for instance, if all the source nodes are different but there is more than one sink, there will simply be one node balance constraint for each; in general, any problem (regardless of the structure of the demand vectors $\boldsymbol{b}^h$) can be transformed to one suitable for the arc-path formulation by augmenting the graph with at most k super-sources and k super-sinks (see Figure 4), but this requires having individual capacity b_j^h on the arc connecting any sink node j with the super-sink t_h, and $-b_i^h$ on (s_h, i) for any source node i;

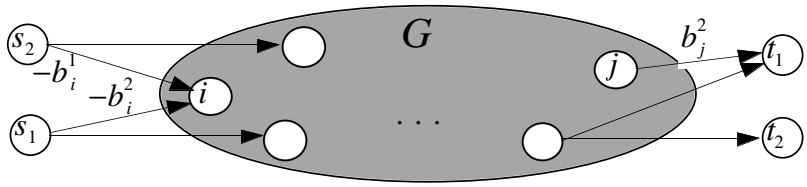

Figure 4: converting a general problem to one with $\mathcal{K} \subseteq \mathcal{N} \times \mathcal{N}$

- for each node-arc formulation, there exists an equivalent "extended" arc-path formulation where the cost of a path p is

$$c_p = \sum_{(i,j) \in P(\mathcal{K}(p))} c_{ij}^{\mathcal{K}(p)}$$

but the converse is true only if the cost of any path can be expressed as the sum of the costs of the arcs composing it: clearly, the arc-path formulation can have a more general structure of path costs, but, on the other hand, in the case of nonlinear separable arc cost functions $c_{ij}^h(x_{ij}^h)$ it has a nonseparable cost function;

- the arc-path formulation does not change if G is an *undirected* graph: the node-arc formulation can be adapted to this case by constructing both the arcs (i, j) and (j, i) relative to each *edge* $\{i, j\}$, and replacing each x_{ij}^h in the mutual capacity constraints with $x_{ij}^h + x_{ji}^h$, hence doubling the number of variables (but not the number of linking rows) in the resulting (LP).

An important, although apparently trivial, remark to be done is that the (MMCF) instances that arise from different applications can be very different under many respects:

- the number k of different commodities ranges from just a few, as in many distribution problems, to the number of all the possible O/D pairs and over, as in traffic models and routing applications: it is shown in Chapter II.2 that such a difference may influence the relative efficiency of alternative solution approaches, i.e. that efficient methods for small k may become less attractive when k grows and vice-versa;
- the mutual capacity constraints may be everything between very "loose" (a few percent being binding in the optimal solution) and very "strict" (30-50% and over): although tightly coupled (MMCF)s are generally harder to solve for all solution methods, the influence can be different for alternative approaches, as e.g. Cost Decomposition algorithms are likely to suffer on tightly coupled instances much more than Interior Point methods;
- the structure of the demand vectors has some influence on the difficulty of the problem, and in fact most of the existing theoretical results about Multicommodity Flows require the single source/sink assumption (if not stronger hypotheses on the demand pattern, such as that all pairs of nodes be source/sink for a commodity): for some solution methods this is also true in practice, as for

instance multiple sources *and* sinks for each commodity impair the possibility of using Shortest Path subproblems instead of Min-Cost Flow ones;

- analogously, individual capacities (that are needed in many real applications) seem to make the problem more "difficult" both in theory, invalidating many theoretical results, and in practice, since again the subproblems cease to be Shortest Paths: on the other hand, "tight" individual capacities may make many of the mutual capacity constraints redundant, diminishing the "coupling" of the problem - using an optimal solution $\boldsymbol{x}^*$ as arc capacities would completely decouple the blocks;
- *undirected* Multicommodity Flow problems are "easier" than *directed* ones, as many theoretical results show: for instance, the maximum inherent gap in the Max-Flow/Min-Cut relation is O(log n) for undirected graphs, but it is O(n) for directed ones (Cf. Sec. II.1.4.6).

Finally, the linear (MMCF) is only a special case - or a relaxation - of more complex problems that have a theoretical and practical interest of their own, such as

- the *Integer Multicommodity Min-Cost Flow* problem, where the integrality of the flow vectors $\boldsymbol{x}^h$ is required: being an $\mathcal{NP}$-hard problem, (IMMCF) calls for the entirely different machinery of enumerative, polyhedral or heuristic methods, that in turn may require the (repeated) solution of (MMCF)-like problems and/or the efficient computation of lower bounds to the integral optimal objective value; this is also true for other (MMCF)-structured $\mathcal{NP}$-hard problems such as the Fixed Charge [Integer] Multicommodity Min-Cost Flow problem discussed in §II.5.2;
- the *Generalised Multicommodity Flow problem*, where the mutual capacity constraints are replaced by a block of general linear constraints $\boldsymbol{D}[\, \boldsymbol{x}^1 \,..\, \boldsymbol{x}^k \,] = \boldsymbol{d}$: in turn, special cases of (GMF) are the *Nonsimultaneous Multicommodity problem* (Cf. §II.5.1), where the commodities are partitioned in p "blocks" $B(\,1\,)\,..\,B(\,p\,)$ and mutual capacity constraints are "restricted" within a block, i.e.

$$\sum_{h \in B(q)} x_{ij}^h \le u_{ij}^q \qquad q = 1 \,..\, p,$$

  and the *Equal Flow problem*, where there is only one commodity but constraints $x_{ij} = x_{uw}$ are imposed on a certain set $\{\, (\,(\,i\,,j\,)\,,\,(\,u\,,w\,)\,)\,\}$ of pairs of the arcs;
- various forms of the *Nonlinear* [Generalised] *Multicommodity Min-Cost Flow* problem, ranging from the "easy" ones with strictly convex separable cost functions $c_{ij}^h(\, x_{ij}^h\,)$ to the very "hard" ones with nonseparable and/or nonconvex functions and/or integrality constraints on the variables;
- the generic block-structured program

$$\text{(BS)} \qquad \min\Big\{ \sum_{h \in \mathcal{K}} c^h(\,\boldsymbol{x}^h\,) : \boldsymbol{x}^h \in X^h \;\forall\; h \in \mathcal{K}\,,\; \sum_{h \in \mathcal{K}} g^h(\,\boldsymbol{x}^h\,) \le \boldsymbol{u} \Big\}$$

  where optimization over a single block

$$\text{(B}_{\boldsymbol{y}}^h\text{)} \qquad \varphi^h(\,\boldsymbol{y}\,) = \min\Big\{\, c^h(\,\boldsymbol{x}^h\,) + \boldsymbol{y} g^h(\,\boldsymbol{x}^h\,) : \boldsymbol{x}^h \in X^h \Big\}$$

  is an "easy" problem for any choice of the dual vector $\boldsymbol{y} \ge \boldsymbol{0}$.

However, less general forms of Multicommodity Flow problems have also received attention in the past: the most important ones are (variants of) the Multicommodity Max-Flow problems discussed in the next paragraph.

II.1.2 Multicommodity Max-Flow formulations

Two different forms of Multicommodity Max-Flow problems have been studied in the literature: for the sake of exposition, their arc-path formulation will be presented first, therefore assuming no individual capacities and an unique source/sink pair (s_h, t_h) for each commodity h. The first version is the *Max-Sum Multicommodity Flow* problem

$$\text{(MSMF)} \qquad \begin{cases} \max \sum_{h \in \mathcal{K}} \sum_{p \in P(h)} x_p \\ \sum_{p:\,(i,j) \in p} x_p \le u_{ij} & \forall\ (i,j) \in \mathcal{A} \\ x_p \ge 0 & \forall\ p \in P \end{cases}$$

which is the direct generalization of the single-commodity Max-Flow problem, since $\Sigma_{p \in P(h)} x_p$ is the total quantity of commodity h (leaving s_h and reaching t_h) in the graph: obviously, for such a problem the uniqueness of the source/sink pair for each commodity can be assumed without loss of generality. The second form, known as the *Maximum Concurrent Multicommodity Flow* problem, is

$$\text{(MCMF)} \qquad \begin{cases} \max \lambda \\ \sum_{p \in P(h)} x_p = \lambda d_h & \forall\ h \in \mathcal{K} \\ \sum_{p:\,(i,j) \in p} x_p \le u_{ij} & \forall\ (i,j) \in \mathcal{A} \\ x_p \ge 0 & \forall\ p \in P \end{cases}$$

and it explicitly considers the demand d_h for each commodity h, trying to maximize the *throughput* λ, i.e. the common fraction of satisfied demand, subject to the capacity constraints: clearly, a multiflow satisfying all the demands exists if and only if the optimal throughput λ^* is at least 1. Equivalently, (MCMF) may be viewed as the search for the largest λ such that a feasible multiflow exists with respect to capacities $(1 / \lambda)\boldsymbol{u}$. Note that (MSMF) can be written in an analogous form by maximizing $\Sigma_{h \in \mathcal{K}} \lambda_h$ subject to

$$\sum_{p \in P(h)} x_p = \lambda_h \qquad \forall\ h \in \mathcal{K}$$

i.e. assuming $d_h = 1\ \forall\ h \in \mathcal{K}$ but allowing a different throughput for each commodity h: this shows that (MCMF) is the restriction of (MSMF) with the additional constraint $\lambda_1 = \lambda_2 = \ldots = \lambda_k$.

The duals are respectively

$$\text{(DMCMF)} \qquad \begin{cases} \min \sum_{(i,j) \in \mathcal{A}} y_{ij} u_{ij} \\ \sum_{(i,j) \in p} y_{ij} \ge z_h & \forall\ p \in P(h),\ \forall\ h \in \mathcal{K} \\ \sum_{h \in \mathcal{K}} z_h d_h \ge 1 \\ y_{ij} \ge 0 & \forall\ (i,j) \in \mathcal{A} \end{cases}$$

and

$$\text{(DMSMF)} \begin{cases} \min \sum_{(i,j) \in \mathcal{A}} y_{ij} u_{ij} \\ \sum_{(i,j) \in p} y_{ij} \geq 1 & \forall\ p \in P(h),\ \forall\ h \in \mathcal{K} \\ y_{ij} \geq 0 & \forall\ (i,j) \in \mathcal{A} \end{cases}$$

and for $k = 1$, $d = 1$ (the single-commodity Max-Flow) they are clearly equivalent: in this case, at least one optimal integral solution $\boldsymbol{y}^* \in \{0, 1\}^m$ exists that can be interpreted as the characteristic vector of a (*s*/*t*-)*cut* - a subset of arcs whose removal disconnects the source from the sink - of minimal capacity. Analogously, any feasible integral solution of (DMSMF) can be interpreted as a *multicut*, i.e. a set of arcs whose removal separates *all* source/sink pairs: therefore, (DMSMF) is the continuous relaxation of the *Minimum Capacity Multicut* problem, and a feasible fractional $\boldsymbol{y}$ is a *fractional multicut* (a convex combination of multicuts) that "covers" all the commodities. Hence, the capacity of *any* multicut provides an upper bound to z(MSMF), but since the integrality property does not hold, a nonzero gap can be expected. Similar relations can be derived for (MCMF): given a *cutset* $\mathcal{S} \subseteq \mathcal{N}$ and the set $\mathcal{K}(\mathcal{S}) = \{ h \in \mathcal{K}: s_h \in \mathcal{S} \,\&\, t_h \notin \mathcal{S} \}$ of the commodities whose source and sink are separated by $\mathcal{S}$, the *capacity* of the induced cut and the *separated demand* are respectively

$$c(\mathcal{S}) = \sum_{i \in \mathcal{S}, j \notin \mathcal{S}} u_{ij} \qquad \text{and} \qquad d(\mathcal{S}) = \sum_{h \in \mathcal{K}(\mathcal{S})} d_h$$

and $c(\mathcal{S}) / d(\mathcal{S})$ is an upper bound on the maximum throughput λ^*. Hence, the "most restraining" cut is the one with the minimum *capacity-to-separated-demand* ratio: this can be shown with simple combinatorial arguments, but it is instructive to derive it from the (LP) formulations. The coefficient of 1 in the objective function of (MCMF), and therefore the right hand side in the constraint $\Sigma_{h \in \mathcal{K}} z_h d_h \geq 1$ of (DMCMF), can be turned to any $\alpha > 0$ with the only effect of multiplying by α the optimal objective value: then, for any feasible $(\bar{\boldsymbol{z}}, \bar{\boldsymbol{y}}) \in \{0, 1\}^{m+k}$, $\bar{\boldsymbol{z}}$ is the characteristic vector of the set of commodities that are separated by the multicut induced by $\bar{\boldsymbol{y}}$, since for any h such that $\bar{z}_h = 1$ the constraints $\Sigma_{(i,j) \in p} \bar{y}_{ij} \geq 1\ \forall\ p \in P(p)$ ensure that at least one arc on any s_h-t_h path is selected. Since $(\bar{\boldsymbol{z}}, \bar{\boldsymbol{y}})$ is still feasible, by setting α equal to the "separated demand" $\bar{\boldsymbol{z}}\boldsymbol{d}$ of the cut $\bar{\boldsymbol{y}}$, the bound $\bar{\boldsymbol{y}}\boldsymbol{u} / \bar{\boldsymbol{z}}\boldsymbol{d} \geq \lambda^*$ is easily established: again, the bound is tight for $k = 1$ but a gap can exist for larger k, an example being the $K_{2,3}$ (undirected) graph depicted in Figure 5 with unitary capacities over all edges and unitary demands between all pair of vertices not directly connected by an edge, that has a minimum capacity-to-separated-demands ratio of 1 (two cuts achieving such a ratio are shown) but for which clearly no feasible multiflow exists. In fact, a stronger path-based upper bound can be obtained by observing that, since each $p \in P(h)$ is an s_h-t_h path, the quantity $\delta_p(\boldsymbol{y}) = \Sigma_{(i,j) \in p} y_{ij}$ is the distance between s_h and t_h along path p if y_{ij} is the length of arc (i, j): for any fixed $\bar{\boldsymbol{y}}$, the "optimal" variables $\boldsymbol{z}(\bar{\boldsymbol{y}})$ are clearly

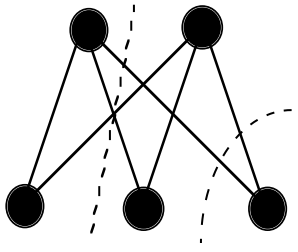

Figure 5: example of Max-Flow / Min-Cut gap

$$\varphi^h(\bar{\boldsymbol{y}}) = z(\bar{\boldsymbol{y}})_h = \min\{ \delta_p(\bar{\boldsymbol{y}}) : p \in P(h) \}$$

i.e. the length of the *shortest path* between s_h and t_h w.r.t. $\bar{\boldsymbol{y}}$, and (DMCMF) can be restated as the problem of finding arc lengths $\boldsymbol{y}$ minimizing $\boldsymbol{y}\boldsymbol{u}$ and such that the weighted (minimal) distance from all the source/sink pairs be at least 1. Analogously, (DMCMF) can be interpreted as the problem of finding arc lengths $\boldsymbol{y}$ minimizing $\boldsymbol{y}\boldsymbol{u}$ and such that the distance between each s_h and t_h is at least 1:

hence, the bound

$$\lambda^* \le \frac{\bar{\boldsymbol{y}}\,\boldsymbol{u}}{\sum_{h \in \mathcal{K}} d_h \varphi^h(\,\bar{\boldsymbol{y}}\,)}$$

can be easily obtained as the previous one for any (non necessarily integral) choice of the distances $\bar{\boldsymbol{y}}$.

The node-arc formulations of the Multicommodity Max-Flow problems are respectively

$$\text{(MSMF)} \quad \begin{cases} \max \sum_{h \in \mathcal{K}} \lambda_h \\ \begin{bmatrix} \boldsymbol{E} & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{E} \\ \boldsymbol{I} & \cdots & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}^1 \\ \vdots \\ \boldsymbol{x}^k \end{bmatrix} - \begin{bmatrix} \boldsymbol{b}^1 & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{b}^k \\ \boldsymbol{0} & \cdots & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_k \end{bmatrix} = \le \begin{bmatrix} \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \\ \boldsymbol{u} \end{bmatrix} \\ \boldsymbol{0} \le \boldsymbol{x}^h \le \boldsymbol{u}^h \quad \forall\, h \in \mathcal{K} \end{cases} \qquad \text{where } \boldsymbol{b}^h = \begin{array}{|c|c|} \hline & \\ -1 & s_h \\ & \\ & \\ +1 & t_h \\ & \\ \hline \end{array}$$

$$\text{(MCMF)} \quad \begin{cases} \max \lambda \\ \begin{bmatrix} \boldsymbol{E} & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{E} \\ \boldsymbol{I} & \cdots & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}^1 \\ \vdots \\ \boldsymbol{x}^k \end{bmatrix} - \lambda \begin{bmatrix} \boldsymbol{b}^1 \\ \vdots \\ \boldsymbol{b}^k \\ \boldsymbol{0} \end{bmatrix} = \le \begin{bmatrix} \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \\ \boldsymbol{u} \end{bmatrix} \\ \boldsymbol{0} \le \boldsymbol{x}^h \le \boldsymbol{u}^h \quad \forall\, h \in \mathcal{K} \end{cases} \qquad \text{where } \boldsymbol{b}^h = \begin{array}{|c|c|} \hline & \\ -d_h & s_h \\ & \\ & \\ +d_h & t_h \\ & \\ \hline \end{array}\ .$$

The first formulation is analogous to that of the single-commodity Max-Flow problem, each variable λ_h being the flow of the artificial "return arc" (t_h, s_h) (Figure 6); the second is different in that the unique variable λ can be interpreted as the flow of an unique "hyperarc with gains" (Figure 7), where each commodity has a different gain d_h, connecting all the sinks to all the sources (the mutual capacity constraints are not represented in the Figures).

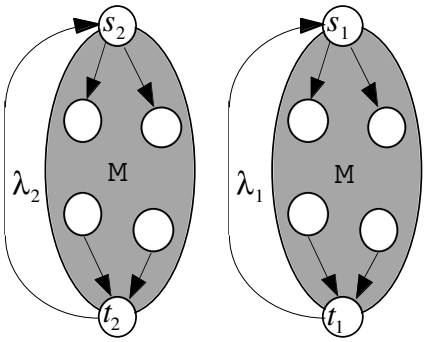

Figure 6: the Max-Sum problem

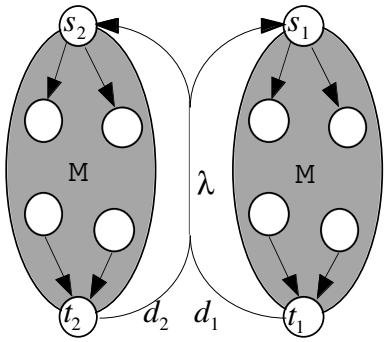

Figure 7: the Concurrent problem

As for (MMCF), the two different Multicommodity Max-Flow problems are special cases of more general classes of problems: (MSMF) is a *Fractional Packing* problem [PST93]

(FP) $\qquad \exists?\, \boldsymbol{x} : \boldsymbol{x} \in X \;\&\; \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{u}$

where $\boldsymbol{A} \ge \boldsymbol{0}$ and optimization over X is "easy", that is $\min\{\ (\boldsymbol{y}\boldsymbol{A})\boldsymbol{x} : \boldsymbol{x} \in X\ \}$ can be efficiently solved for any $\boldsymbol{y} \ge \boldsymbol{0}$ (*Fractional Covering* problems have $\boldsymbol{A}\boldsymbol{x} \ge \boldsymbol{u}$ instead). The idea behind this definition is that X is a polyhedron given by an "oracle" and one wants to find a feasible fractional packing (covering) of its extreme points: note that in (MSMF) *flows* are packed, while in its dual *cuts*

are used to cover. (MCMF) is a *Resource Sharing* problem [GK94a]

$$\text{(RS)} \qquad \min\left\{ \lambda : \boldsymbol{x}^h \in X^h \ \forall\ h \in \mathcal{K},\ \sum_{h \in \mathcal{K}} \boldsymbol{g}^h(\boldsymbol{x}^h) \le \lambda \boldsymbol{u} \right\}$$

where each X^h is a nonempty convex compact set, each $\boldsymbol{g}^h()$ is a *nonnegative* continuous convex (vector) function and optimization over a single block - the (B_y^h) problem defined at the end of §II.1.2 - is "easy": here, the idea is that the $\boldsymbol{g}^h()$ functions measure the consumption of some resources shared among the (otherwise independent) blocks X^h, and the problem is to find a point $\boldsymbol{x} = [\ \boldsymbol{x}^1\ ..\ \boldsymbol{x}^k\ ]$ minimizing the total resource consumption. Some efficient ε-approximation algorithms have been recently devised for both (FP) and (RS) on the basis of results initially obtained for Multicommodity Max-Flow problems, as discussed in more details in §II.1.4.

II.1.3 Applications

Multicommodity Flow problems have a wide area of application, far beyond the "obvious" one as models for transportation problems: however, this usage alone would justify the interest in efficient solution methods, since many practical problems can be formulated as (Generalised) Multicommodity Flow Probles [To71] [GG71] [Go75] [PS83] [CG84] [ZTT84] [AKS88] [AKL93] [BG95] [CGN95] [EGM95] [HBL95] (and many more). In these models, a commodity may represent different things, such as skills of personnel to be assigned to new jobs [AKS88], depots to which vehicles belong [CG84], data to be exchanged between antennas [BG95] or types of aircrafts to be assigned to scheduled flights [HBL95]: however, in some relevant models a commodity is just the type of some sort of merchandise flowing from its source to its destination along the network. An interesting representative of such models is the Freight Transportation Planning problem [FPL93], that can be briefly described as follows: on a physical transportation network (highway, railway …), represented by a *graph of terminals* $G(\mathcal{N}, \mathcal{A})$, a fixed set of *trips* (of trains, trucks ...) have already been scheduled within a given *planning horizon* $T = \{\ t_1\ ..\ t_p\ \}$. Demands for transporting different goods along the network are given, each one with its origin and destination terminals and the (desired) time window, that must be satisfied optimally with the available *services*, i.e. trips. The problem can be formulated as a (MMCF) on the *time-space graph* $G'(\mathcal{N} \times T, \mathcal{A}')$ (Figure 8), where each node I_t represents the terminal I at time instant t, there is the *service* arc $(I_t, J_s) \in \mathcal{A}'$ if and only if a service starts from terminal I at time instant t and ends at terminal J at time instant s and there is the *holding* arc $(I_t, I_{t+1}) \in \mathcal{A}'$ whenever goods are allowed to wait at the terminal I from t to $t + 1$.

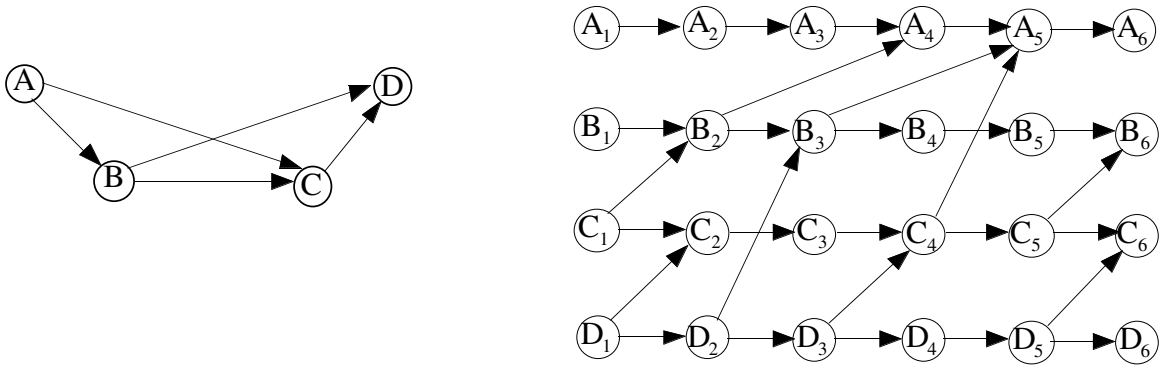

Figure 8: a graph with four terminals and the corresponding space-time graph for six time periods

Transport costs are associated to service arcs, while inventory costs are associated to holding arcs: using the technique of node splitting, costs can be assigned also to nodes for representing operations to be performed in terminals, such as rearranging the cars of a train. Typically, real-world models have hundreds of commodities, medium-sized graphs and up to thousands of time instants, giving rise to (MMCF) of very large size: furthermore, (FTP) is actually only a subproblem of the Freight Services Scheduling problem where the optimal set of services is requested, so that reasonable solution times are necessary to allow a reasonable number of alternatives be tested, either in an automated optimzation routine or in an interactive mode.

In many practical cases, "good" integer solutions are required: being $\mathcal{NP}$-hard, the Integer (MMCF) calls for the machinery of heuristic methods [BS93] [Ba93], implicit enumeration methods (Branch & Bound, Branch & Price) [BHV96] or polyhedral methods (Cutting Plane, Branch & Cut) [BCF95]. However, all the exact approaches require the solution of a sequence of continuous (MMCF)s - Generalized (MMCF)s in a Branch & Cut - in order to obtain an integer primal solution; furthermore, efficient heuristics may benefit of dual (cost) information, that can be obtained by solving the continuous relaxation or some Lagrangean Dual.

An important variant of the Integer (MMCF) is the *Fixed Charge Multicommodity* (*Min-Cost*) *Flow* problem [GC95], belonging to the broad class of Network Design problems [KBY94] [Mi89], in which the arcs must be constructed - at a fixed cost - before the flow of any commodity can be routed along them: (FCMF) is known to be among the most difficult (mixed-)integer problems (the single-commodity version is already $\mathcal{NP}$-hard [HS89]), but it has a large number of practical and theoretical applications. In fact, other than being a model for various problems in telecommunications [LSW95] [PR96], (FCMF) is used as a tool for reformulating and approaching a large class of difficult problems such as the Steiner Tree problem [NW88 §II.6.4], the Constrained Minimal Spanning Tree problem [Ka94] and a number of Production Scheduling and Lot Sizing problems [ZTT84] [PW95]: for all such problems, (FCMF) provides both "stronger" (LP) formulations and some large classes of strong valid inequalities [NW88 §II.2.4].

An interesting application of (possibly nonlinear) Multicommodity Flow problems is the design of routing algorithms: in fact, they provide a general analytic model for describing a variety of basic routing problems [RH89], such as to satisfy communication demands subject to the network capacity, to minimize communication delays or to maximize network utilization in presence of congestion. In the "basic" version of these models, commodities are just communicating pairs of nodes with (estimated) demands, i.e. $k \approx O(n^2)$, but they can also represent different classes of traffic, such as video/voice, flow control and standard data packets: multi-class traffic control will be one of the major characteristics of the next generation of routing protocols, especially after the advent of ATM communication networks. In this framework, Multicommodity Flows can provide (off-line) "optimal" routings against with to evaluate the effectiveness of actual routing protocols [CWH92]; furthermore, any (exact or approximate) solution method for Multicommodity Flow problems that is suitable for fine-grain, node-based distributed implementation (such as those of Sec. II.1.4.5) can be "interpreted" as an on-line routing protocol, and vice-versa [FGK73] [Ga77] [St77] [CC91] [RE92].

Finally, Multicommodity Max-Flow problems have been used as a tool for approximating a variety of $\mathcal{NP}$-hard graph problems arising in VLSI circuits layout [KAR90] [Tr91], including the Minimum

Feedback Arc Set, the Minimum Cut Linear Arrangement and the Minimum Area Layout: the idea is that undirected (MCMF) with unit demands and a commodity for each node pair (the *All-Pair Unit Demands* problem) provide a mean for (approximately) constructing cuts with "low" capacity and "balanced" sides. In fact, the cutset $\mathcal{S}^*$ achieving the minimum capacity-to-separated-demand ratio solves

$$\mathcal{S}^* = \text{argmin} \{ c(\mathcal{S}) / (|\mathcal{S}| \cdot |\mathcal{N} - \mathcal{S}|) : \mathcal{S} \subseteq \mathcal{N} \}$$

and the denominator $|\mathcal{S}| \cdot |\mathcal{N} - \mathcal{S}|$ is large when both sides have about the same size: in the case of unitary capacities, the cut induced by $\mathcal{S}^*$ is referred to as the *sparsest cut* and it is clearly a good way of evenly dividing the node set $\mathcal{N}$ by cancelling the minimal number of arcs. In [LR88], the first $O(\log n)$-approximation algorithm for (APUD) is presented and recursively used to construct an $O(\log^2 n)$-optimal solution to the Minimal Capacity 1/3 - 2/3 Separator problem, i.e.

$$\text{argmin} \{ c(\mathcal{S}) : \mathcal{S} \subseteq \mathcal{N}, n / 3 \leq |\mathcal{S}| \leq 2n / 3 \}.$$

In turn, such a Minimal Capacity "balanced" Cut can be successfully used to drive a Divide and Conquer strategy [Le83] [BL84], where two smaller subproblems are (recursively) solved on the two sides of the (approximatedly) optimal cut and the two partial solutions are pasted together: the almost balanced sides ensure at most a logarithmic number of recursion steps, while the small capacity limits the amount of work required to paste the two partial solutions. Interestingly enough, the original algorithm in [LR88] uses the *dual* optimal solution of (APUD), and it was also the seminal work for the ε-approximation algorithms for Multicommodity Max-Flow problems discussed in Sec. II.1.4.6.

II.1.4 Algorithmic approaches

Historically, (MMCF) has motivated many important algorithmic developments that have later found broader applications: while its arc-path formulation inspired the first column-generation approaches [FF58], the node-arc formulation is the archetype of the block-structured Linear Program, and the celebrated Dantzig-Wolfe decomposition algorithm [DW60] was originally devised for solving Multicommodity Flow problems. As more broadly discussed below, this "pushing" effect is lasting until today, and a number of interesting recent developments have been motivated by the need of solving, at least approximately, (MMCF)s of very large size.

Several algorithmic approaches have been proposed for (MMCF), but unfortunately the available surveys [As78] [Ke78] [AMO93 Chapter 17] only present the more "classical" techniques, sometimes dating back to the sixties, and this is also true for computational studies [As76] [AHK79]: the only recent surveys are limited to the fast-growing areas of approximation algorithms [AL93] [Ta94] and parallel approaches [Ze93] [DKM94]. The main algorithmic approaches to (MMCF) will be presented in the following, with a particular stress on Cost Decomposition methods: the focus will be on *sequential* approaches, since the issues relative to *parallel* algorithms will be discussed in §II.3.1.

II.1.4.1 Basis Partitioning methods

Basis Partitioning methods [HL72] [Ke77] [CN95] are specialized versions of the (usually Primal) Simplex method for Linear Programming that exploit the network structure of the basis for reducing the cost of the primal/dual basic solutions calculation and of pivoting: matrix factorizations or

inversions cannot be completely avoided (as it is the case for the single-commodity case), but they can be limited to a working matrix of a much smaller dimension. Although the details of the method are rather involved, the idea is quite simple: in each basis of (MMCF), the arcs associated with basic variables for a commodity can be partitioned into two sets, one forming a spanning tree of the graph G and the other being that of "extra" arcs corresponding to the mutual capacity constraints. By introducing slack variables $\boldsymbol{s}$ in the mutual capacity constraints, i.e. $\sum_{h \in \mathcal{K}} \boldsymbol{x}^h + \boldsymbol{s} = \boldsymbol{u}$, the problem can be turned into a standard equality form and every basis $\boldsymbol{B}$ can be partitioned as

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{L}_1 & \boldsymbol{R}_1 & \boldsymbol{0} \\ \boldsymbol{L}_2 & \boldsymbol{R}_2 & \boldsymbol{0} \\ \boldsymbol{L}_3 & \boldsymbol{R}_3 & \boldsymbol{I} \end{bmatrix} \qquad \text{where} \qquad \boldsymbol{L}_1 = \begin{bmatrix} \boldsymbol{B}_1 & \boldsymbol{0} & \mathtt{L} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{B}_2 & \mathtt{L} & \boldsymbol{0} \\ \mathtt{M} & \mathtt{M} & \mathtt{O} & \mathtt{M} \\ \boldsymbol{0} & \boldsymbol{0} & \mathtt{L} & \boldsymbol{B}_k \end{bmatrix}$$

and each $\boldsymbol{B}_h$ is the node-arc incidence matrix of a spanning tree T_h: the second and third set of rows correspond to active mutual capacity constraints and active slacks (i.e. inactive mutual capacity constraints) respectively, and the second set of columns corresponds to non-tree basic arcs. The solution of linear systems on $\boldsymbol{B}$ can be performed by first efficiently solving k smaller systems on the submatrices $\boldsymbol{B}_h$ (visiting T_h) and then solving a system on the *working matrix* $\boldsymbol{W} = \boldsymbol{R}_2 - \boldsymbol{L}_2\boldsymbol{L}_1^{-1}\boldsymbol{R}_1$ that can be efficiently computed and updated (after pivots) by exploiting the structural properties of $\boldsymbol{L}_1$: Primal Partitioning implementations [Ke78b] [Ca94] usually outperform "standard" simplex codes due to the greatly reduced cost of the core matrix operations. Moreover, the structure of the problem can be further exploited by performing "advanced starts": a typical way is to solve the k (MCF) subproblems independently to obtain a primal unfeasible (but "good") base, then achieve feasibility by solving an appropriately modified problem and finally use the resulting primal feasible base as a starting point - the computational results in §II.2.4 will show the importance of such details.

These techniques can also be used to efficiently implement *Reduced Gradient* methods for the Nonlinear (nonseparable) (MMCF), analogously to the single-commodity case [ZM88]; in general, many algorithms for solving the equality-constrained Nonlinear Problem

$$\min \{ f(\boldsymbol{x}) : \boldsymbol{E}\boldsymbol{x} = \boldsymbol{b}\ ,\ \boldsymbol{0} \le \boldsymbol{x} \le \boldsymbol{u} \}$$

are based on the *active set* philosophy, where only a subset of the variables (the *active* ones) are allowed to vary at any given iteration: the Reduced Gradient methods further subdivide the active variables into *Basic* and *Superbasic*, where the superbasic variables are the independent ones while the basic variables are adjusted in order to preserve the feasibility. Assuming for simplicity that the matrix $\boldsymbol{E}$ has full row rank, and that the upper bounds $\boldsymbol{u}$ are not present, $\boldsymbol{E}$ and $\boldsymbol{x}$ are partitioned as

$$\boldsymbol{E} = [\, \boldsymbol{E}_B \boldsymbol{E}_S \boldsymbol{E}_N \,] \qquad \text{and} \qquad \boldsymbol{x} = [\, \boldsymbol{x}_B \boldsymbol{x}_S \boldsymbol{x}_N \,]$$

where $\boldsymbol{E}_B$ is a nonsingular square matrix and $\boldsymbol{x}_N$ (the Nonbasic variables) are fixed to $\boldsymbol{0}$: the Superbasic variables $\boldsymbol{x}_S$ are "free", while the Basic variables $\boldsymbol{x}_B$ are set to $\boldsymbol{E}_B^{-1}(\, \boldsymbol{b} - \boldsymbol{E}_S\boldsymbol{x}_S \,)$ to satisfy the equality constraints, so that the *Reduced Gradient*

$$\nabla f_R = \nabla f(\, \boldsymbol{x} \,)_S - \boldsymbol{E}_S^T(\, \boldsymbol{E}_B^{-1} \nabla f(\, \boldsymbol{x} \,)_B \,)$$

gives a descent direction in the space of the $\boldsymbol{x}_S$; furthermore, a properly defined *Reduced Hessian* can be used to extract suitable second-order information, allowing the implementation of rapidly

convergent Newton or Quasi-Newton methods. Some fundamental details cannot be described here, such as the use of dual information (Lagrangean multipliers) for determining whether the current active set $B \cup S$ has to be updated by inserting some nonbasic variable (pricing) or eliminating some redundant active one: in the linear case, for instance, the only Superbasic variable is the Nonbasic one entering at the beginning of any iteration, and it is transformed into Basic to replace the leaving one. However, it is not hard to see that the necessary tasks can be efficiently dealt with by exploiting the structural properties of $\boldsymbol{E}_B$: hence, as in the linear case, Basis Partitioning techniques greatly reduce the cost of many of the core operations of standard optimization algorithms. However, while the complexity of each step is about $\max\{ O(k), O(m^2) \}$, the number of steps can be expected to increase with km, so that such methods may not be very attractive as k increases.

II.1.4.2 Cost Decomposition methods

Among the approaches to (MMCF), Cost Decomposition methods are perhaps the most intriguing, as demonstrated by the constant stream of work dedicated to this class of algorithms: a natural derivation of such methods can be obtained from the dual of the arc-path formulation of (MMCF)

$$\text{(DMMCF)} \quad \begin{cases} \max \sum_{h \in \mathcal{K}} z_h d_h - \sum_{(i,j) \in \mathcal{A}} y_{ij} u_{ij} \\ c_p + \sum_{(i,j) \in p} y_{ij} \geq z_h \qquad \forall\ p \in P(h),\ \forall\ h \in \mathcal{K} \\ y_{ij} \geq 0 \qquad\qquad \forall\ (i,j) \in \mathcal{A} \end{cases} .$$

As for the Maximum Concurrent Multicommodity Flow problem, $c_p + \sum_{(i,j) \in p} y_{ij} = \sum_{(i,j) \in p} (c^h_{ij} + y_{ij}) = \delta_p(\boldsymbol{c}^h + \boldsymbol{y})$ is the distance between s_h and t_h along path p with arc lengths $\boldsymbol{c}^h + \boldsymbol{y}$, hence at optimality z_h is the length of the *shortest path* between s_h and t_h and the problem can be rewritten as

$$\text{(DMMCF)} \qquad \max \left\{ \sum_{h \in \mathcal{K}} \varphi^h(\boldsymbol{y}) - \boldsymbol{y}\boldsymbol{u} : \boldsymbol{y} \geq \boldsymbol{0} \right\}$$

where $\varphi^h(\boldsymbol{y}) = d_h \min\{ \delta_p(\boldsymbol{c}^h + \boldsymbol{y}) : p \in P(h) \}$ is the cost of sending d_h units of flow from s_h to t_h along the shortest path: hence, the dual of (MMCF) is naturally viewed as the maximization of a function involving Shortest Path computations w.r.t. the modified costs $\boldsymbol{c}^h + \boldsymbol{y}$.

More in general, (MMCF) is a special case of the block-structured program (BS) introduced at the end of §II.1.1, and it is well-known [Ge69] that, without essentially any assumption over the sets X^h and the functions $c^h()$ and $\boldsymbol{g}^h()$, the *dual function*

$$\varphi(\boldsymbol{y}) = \sum_{h \in \mathcal{K}} \varphi^h(\boldsymbol{y}) - \boldsymbol{y}\boldsymbol{u}$$

is concave; moreover, if all the blocks X^h are closed convex sets, and all the $c^h()$ and $\boldsymbol{g}^h()$ functions are convex and finite everywhere over X^h, then the *dual* (concave, max-min) *problem*

$$\text{(DBL)} \qquad \max\{ \varphi(\boldsymbol{y}) : \boldsymbol{y} \geq \boldsymbol{0} \}$$

is equivalent to (BL) in the sense that

- each feasible solution of (DBL) provides a lower bound to z(BL), and each feasible solution of (BL) provides an upper bound to z(DBL): therefore, if (DBL) is nonempty then (BL) is either empty or bounded, and if (BL) is nonempty then (DBL) is either empty or bounded;

- under mild conditions, implied by all the known constraint qualifications, (BL) and (DBL) have the same optimal objective function value whenever (BL) is nonempty and bounded; in particular, such conditions are always true when X^h are polyhedra;
- the *value function* of (BL)

$$v(\boldsymbol{z}) = \min\left\{ \sum_{h \in \mathcal{K}} c^h(\boldsymbol{x}^h) : \boldsymbol{x}^h \in X^h \ \forall\ h \in \mathcal{K}, \sum_{h \in \mathcal{K}} g^h(\boldsymbol{x}^h) \le \boldsymbol{z} \right\}$$

  is a convex function over the (convex) set of all the *feasible* right hand sides $\boldsymbol{z}$;
- for any optimal solution $\bar{\boldsymbol{y}}$ of (DBL), $-\bar{\boldsymbol{y}}$ is a subgradient of $v()$ in $\boldsymbol{z} = \boldsymbol{u}$; moreover, *any single* $\bar{\boldsymbol{y}}$ characterises the set of *all* optimal solutions of (BL), in that $\bar{\boldsymbol{x}}$ is optimal for (BL) if and only if

$$\bar{\boldsymbol{x}}^h = \operatorname{argmin}\left\{ c^h(\boldsymbol{x}^h) + \bar{\boldsymbol{y}} g^h(\boldsymbol{x}^h) : \boldsymbol{x}^h \in X^h \right\} \qquad \forall\ h \in \mathcal{K}$$

$$\sum_{h \in \mathcal{K}} g^h(\bar{\boldsymbol{x}}^h) \le \boldsymbol{u}$$

$$\bar{\boldsymbol{y}}\left(\sum_{h \in \mathcal{K}} g^h(\bar{\boldsymbol{x}}^h) - \boldsymbol{u} \right) = 0 \qquad \text{(} \textit{complementary slackness} \text{).}$$

Hence, duality provides an attractive way of solving (BL) by attacking its dual (DBL), i.e. the constrained maximization of a concave nondifferentiable function $\varphi(\boldsymbol{y})$: for any given $\boldsymbol{y}$, the computation of $\varphi(\boldsymbol{y})$ and of a subgradient requires the solution of k independent block problems (B_y^h), that for (MMCF) are just

$$(\mathrm{RMF}_y^h) \qquad \min\left\{ (\boldsymbol{c}^h + \boldsymbol{y})\boldsymbol{x}^h : \boldsymbol{E}\boldsymbol{x}^h = \boldsymbol{b}^h,\ \boldsymbol{0} \le \boldsymbol{x}^h \le \boldsymbol{u}^h \right\}$$

i.e. single-commodity Min-Cost Flow (or Shortest Path) subproblem with the modified costs $\boldsymbol{c}^h + \boldsymbol{y}$, for which several efficient algorithms exist. Hence, the solution of (DBL) can be tackled with the methods discussed in Part I: surprisingly, the first attempts in this direction are recent [Me90] [Me94] [FG96], maybe because of the lack of widely available general-purpose Bundle codes. However, a number of alternative approaches have been proposed: most of these methods could essentially work with any (BL) problem, even though they have been mostly applied, or even initially proposed, in the context of Multicommodity Flows. This is the case of the *Dantzig-Wolfe decomposition method* for Linear Problems [DW60], that is well-known [Le91] to be equivalent to a *Cutting Plane* approach [Ke60] to the maximization of the dual function $\varphi(\boldsymbol{y})$: an extensive description of the method and its relations with NDO algorithms is given in §I.1.2. Since its proposal, about 40 years ago, many sophisticated implementations, both sequential [HL81] [HL83] [JLF93] and parallel [HLS88] [GH93] [DKM94], have been developed: however, the results have not always been encouraging, motivating the substantially negative opinion of many experts that "... the folklore is that generally such [Cost Decomposition] schemes take a long time to converge so that they're slower than just solving the model as a whole, although research continues. For now my advice, unless [...] your model is so huge that a good solver can't fit it in memory, is to not bother decomposing it. It's probably more cost effective to upgrade your solver, if the algorithm is limiting you, than to invest your time ..."◊. However, if Cost Decomposition approaches are of any value, Multicommodity Flow problems are probably the application where such a value could show up, and in fact some of the recent results appear to be more encouraging, despite the enormous progresses that standard (LP)

◊ John W. Gregory, in the "Linear Programming FAQ" monthly posted to the UseNet newsgroup sci.op-research.

codes have attained in the last few years. The most significant recent findings concern the effect of the number and the "quality" of the proposals (new vertices) that are sent by the block solvers to the Master Problem: providing multiple solutions at each iteration can help in reducing the convergence problems that are often experienced by Decomposition methods, although it seems to be important that the vertices be at least near-optimal for the current subproblems [HLS88]. For the Multicommodity Flow case, a somewhat analogous effect has been observed by analysing the (relevant) impact that the formulation has on the performances of the DW method [JLF93]: path-based formulations, where commodities are origin/destination pairs, seem to converge much faster than tree-based formulations where commodities are aggregated by common origin. This is not surprising, since at any iteration a tree-based formulation produces (for each commodity) a tree with a single associated multiplier, whereas a path-based formulation produces an equivalent set of paths, each one having its independent multiplier: hence, the "disaggregated" Master Problem is much more "flexible", and fewer points are likely to be needed in order to generate the optimal solution; in other words, the number of different trees of a graph is much larger than the number of different paths. Furthermore, the running time of the Master Problem does not grow too much, despite the greatly increased dimensions: this is both due to the reduced number of overall iterations and to the higher "sparsity" of the columns of the Master Problem w.r.t. those of the tree-based formulation.
Such an implementation of the DW method is essentially a *Column Generation* approach applied to the arc-chain formulation of (MMCF), with "multiple pricing" at each iteration: actually, the general DW method was inspired by the previous work of Ford and Fulkerson [FF58] on CG approaches to Multicommodity (Max-)Flow problems, that have been [Je66b] [CD74] and continue to be [Po92] [BHJ95] an active area of research. One interesting recent result is the Primal Partitioning technique for the CG method proposed in [FPL93]: it is based on the observation that (Cf. §II.1.1) there are at most $m + k$ "active" paths along which a nonzero flow is sent, so that, if $k > m$, there are at least $k - m$ commodities sent along an unique path. Then, every basis $\boldsymbol{B}$ can be partitioned as

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{I} & \mathbf{0} & \mathbf{0} \\ \Delta_0 & \boldsymbol{P} & \mathbf{0} \\ \Delta_1 & \Delta_2 & \boldsymbol{I} \end{bmatrix}$$

where the first set of rows corresponds to *unsplit* commodities, the second to *split* commodities and saturated arcs and the third to unsaturated arcs (active slack variables): the three sets of columns correspond respectively to path variables for unsplit commodities, path variables for split commodities and basic slack variables. Hence, any system involving $\boldsymbol{B}$ can be efficiently solved as long as the inverse or a factorization of the submatrix $\boldsymbol{P}$ is available: as for the "standard" Primal Partitioning methods, the fine details of the data structures update are rather involved (there are 12 different types of pivot steps according to the type of the entering and leaving variable, out of which only 9 are feasible), but it is clear that such an implementation can contribute in reducing the computational cost of the Column Generation method.

Seen from the dual viewpoint, however, all those methods are still Cutting Plane approaches to the maximization of $\varphi()$: Bundle methods are not the only "modern" alternative to CP, others being the *Analytic Centre Cutting Plane Methods* [GHV92] [GHV93] described in some detail in §I.1.1. Ad-hoc implementations of ACCPM for (MMCF) [GGS94] have shown to be effective for the solution

of large instances: however, the "standard" ACCPM can be significantly slower than a carefully implemented CP algorithm. This behaviour is inherent in the "conservative" viewpoint of Analytic Centre Methods, that have been designed for obtaining good complexity bounds and therefore does not make use of valuable information, such as the optimum of the CP model, to avoid the case where it may be misleading: in fact, these methods are resistant to "difficult" instances, where the CP fails, but can be slower on the "easy" ones where the CP is efficient. Hence, a promising alternative is that of "hybrid" methods [MGV96], that follow the standard CP strategy as long as the speed of convergence is sufficient, and turn to the ACCPM strategy when the going gets tough; alternatively, it is possible to evaluate φ() twice per iteration, one in the (approximate) Analytic Centre and one in the (approximate) maximizer of the CP-model.

The major drawback of Cost Decomposition methods is certainly the need of maximizing a NonDifferentiable function, where the efficient smooth optimization machinery cannot be used: an attractive alternative is therefore to replace φ() with a "nicer" (smooth) function, such as the *Augmented Lagrangean function*

$$(\mathrm{AL}_{y,t}) \quad \tilde{\varphi}_t(\boldsymbol{y}) = \begin{cases} \min \sum_{h \in \mathcal{K}} \boldsymbol{c}^h \boldsymbol{x}^h + \boldsymbol{y}\left(\sum_{h \in \mathcal{K}} \boldsymbol{x}^h - \boldsymbol{u} \right) + t \left\| \left[\sum_{h \in \mathcal{K}} \boldsymbol{x}^h - \boldsymbol{u} \right]_+ \right\|_2^2 \\ \boldsymbol{E}\boldsymbol{x}^h = \boldsymbol{b}^h \,,\ \boldsymbol{0} \le \boldsymbol{x}^h \le \boldsymbol{u}^h \qquad \forall\ h \in \mathcal{K} \end{cases}$$

where $[\cdot]_+$ is the projection operator over the nonnegative hortant of $\mathbb{R}^m$. It is well-known that $\tilde{\varphi}_t()$ is differentiable, and that there exists a sufficiently large $\bar{t} < \infty$ such that

$$(\mathrm{DAL}_{\bar{t}}) \qquad \max\{\ \tilde{\varphi}_{\bar{t}}(\boldsymbol{y}) : \boldsymbol{y} \ge \boldsymbol{0}\ \}$$

is equivalent (DMMCF); although the treatment is restricted here to Multicommodity Flows, the results are valid for a much broader class of block-structured problems. Being a smooth problem, $(\mathrm{DAL}_{\bar{t}})$ can be efficiently solved by means of very simple methods: in fact, the customary *Method Of Multipliers* amounts at computing $\tilde{\varphi}_t(\bar{\boldsymbol{y}})$ and its gradient $\nabla\tilde{\varphi}_t(\bar{\boldsymbol{y}}) = t\left(\sum_{h \in \mathcal{K}} \boldsymbol{x}(\bar{\boldsymbol{y}}, t)^h - \boldsymbol{u} \right)$, where $\boldsymbol{x}(\boldsymbol{y}, t)$ is the (unique) primal optimal solution of $(\mathrm{AL}_{y,t})$, and update the multipliers as

$$\bar{\boldsymbol{y}} := \left[\bar{\boldsymbol{y}} + 2t\nabla\tilde{\varphi}_t(\bar{\boldsymbol{y}}) \right]_+$$

therefore being just a (projected) gradient method. Clearly, the *penalty parameter t* - that can also be replaced by a vector of parameters, one for each constraint - must be appropriately managed, and second-order methods can be used to accelerate the speed of convergence: however, the MOM in its simplicity often reveals to be efficient enough. The drawback of AL-based methods, at least in the context of Multicommodity Flows, is that $(\mathrm{AL}_{y,t})$ is a large nonseparable (QP): hence, it may be exceedingly costly to solve it with standard packages [DP95], even though the number of MOM iterations is sufficiently small. However, $(\mathrm{AL}_{\bar{y},t})$ can be solved in a *block Gauss-Seidel* fashion, i.e. (approximately) solving

$$\min\left\{ \left(\boldsymbol{c}^{\bar{h}} + \boldsymbol{y} \right)\boldsymbol{x}^{\bar{h}} + t \left\| \left[\sum_{h \in \mathcal{K}} \boldsymbol{x}^h - \boldsymbol{u} \right]_+ \right\|_2^2 : \boldsymbol{E}\boldsymbol{x}^{\bar{h}} = \boldsymbol{b}^{\bar{h}} \,,\ \boldsymbol{0} \le \boldsymbol{x}^{\bar{h}} \le \boldsymbol{u}^{\bar{h}} \right\}$$

where all the commodities but $\bar{h}$ are fixed, and then iterating $\bar{h}$ over $\mathcal{K}$ in a round-robin way: in this case, each subproblem is a Nonlinear separable (MCF), where the cost of arc (i, j) is linear if it is *nonsaturated* $\left(\sum_{h \in \mathcal{K}} x_{ij}^h < u_{ij}\right)$ and quadratic otherwise, for which efficient algorithms are available, such as the specialized ε-relaxation method of [DMZ95]. The method can also be parallelized in a

block Jacobi fashion, using either simple or complex coordination steps [ZaA95], as an application of the Parallel Variable Distribution paradigm [FM94]. Another possibility is the Nonlinear *Simplicial Decomposition* method [VHo77], where the feasible region of $(AL_{\bar{y},t})$ is approximated by the convex combination of some of its vertices, a restricted (quadratic) subproblem is solved giving a tentative optimal solution $\bar{\boldsymbol{x}}$, and k independent Linear (MCF)s using the gradient (i.e. a linearization) of the objective function in $\bar{\boldsymbol{x}}$ as costs are solved to obtain a new vertex: with this implementation, the Augmented Lagrangean method is a variant of the Bundle approach, where the subproblem $(\Delta_{\bar{y},t})$ on the *full* polyhedron $\bar{X}$ (Cf. §I.2.2) is solved to optimality rather than performing a Serious Step whenever possible (possibly indicating a potential advantage of the Bundle method).

A different approach to the "smoothing" of φ() [DPZ91] [PZ92] [PZ94] uses the linear-quadratic (univariate) *Smooth Penalty Function*

$$\sigma_\varepsilon(x) = \begin{cases} 0 & x \le 0 \\ x^2/2\varepsilon & 0 \le x \le \varepsilon \\ x - \varepsilon/2 & \varepsilon \le x \end{cases}$$

so that the "master" problem

$$(M_{\varepsilon,t}) \qquad \begin{cases} \min \sum_{h \in \mathcal{K}} \boldsymbol{c}^h \boldsymbol{x}^h + t\left(\sum_{(i,j) \in \mathcal{A}} \sigma_\varepsilon\left(\sum_{h \in \mathcal{K}} x_{ij}^h - u_{ij} \right) \right) \\ \quad \boldsymbol{E}\boldsymbol{x}^h = \boldsymbol{b}^h \,,\ \boldsymbol{0} \le \boldsymbol{x}^h \le \boldsymbol{u}^h \qquad \forall\ h \in \mathcal{K} \end{cases}$$

has a differentiable objective function: once again, the problem is *not* separable into k single-commodity (MCF)s, requiring e.g. a Simplicial Decomposition method for its solution. Such an implementation of the *Smooth Penalty Function* method is similar to that of the Bundle and AL approaches, but the "restricted" subproblem is *not* a (QP), and therefore calls for more general approaches: for instance, a Truncated Newton Method is used in [PZ92]. A possible advantage of the SPF method over both the AL and the Bundle approaches is that $(M_{\varepsilon,t})$ has only a parameter ε instead of the m-vector $\boldsymbol{y}$: however, it is usually beneficial to use modified Augmented Lagrangeans with vector of multipliers $\boldsymbol{t}$, so that it may conversely be better to have a different ε_i for each constraint.

Once a "basic" technique (ACCPM, AL, Bundle, CG, DW, SPF ...) is chosen, the actual performances depend on many details at the implementation level, such as how the "master problem" and the "subproblems" are solved: hence, it is difficult to assess the relative merits of the different ideas, also because extensive computational comparisons of the different implementations are *not* available. In our opinion, a comparison of "uniform" implementations of all the above methods would be a valuable piece of work.

The Cost Decomposition idea is also behind the recent ε-approximate algorithms, where constraints are dualized by means of an exponential penalty (or a logarithmic barrier) function, discussed in Sec. II.1.4.6; furthermore, these methods give rise to - more or less obvious - opportunities for parallelization, that will be discussed in §II.3.1.

II.1.4.3 Resource Decomposition methods

The basic idea of Resource Decomposition methods stems from a simple observation: if an optimal

solution $\bar{\boldsymbol{x}}$ of (MMCF) would be known, optimal dual variables could be retrieved by allocating *a priori* the capacity (resource) between the commodities, i.e. setting $u_{ij}^h = \bar{x}_{ij}^h$. More formally, the Resource Allocation problem

$$\text{(RD)} \qquad \min\{\ \psi(\boldsymbol{r}) : \Sigma_{h \in \mathcal{K}} \boldsymbol{r}^h \le \boldsymbol{u}\ ,\ \boldsymbol{r} \ge \boldsymbol{0}\ \}$$

with the (polyhedral and convex) function $\psi() : \mathcal{R}^{m \times k} \to \mathcal{R} \cup \{ +\infty \}$

$$\psi(\boldsymbol{r}) = \Sigma_{h \in \mathcal{K}} \min\{\ \boldsymbol{c}^h \boldsymbol{x}^h : \boldsymbol{E}\boldsymbol{x}^h = \boldsymbol{b}^h\ ,\ \boldsymbol{0} \le \boldsymbol{x}^h \le \boldsymbol{r}^h\ \}$$

is equivalent to (MMCF) since any optimal primal solution $\bar{\boldsymbol{x}}$ of (MMCF) also solves (RD) and vice-versa: (RD) is a constrained NonDifferentiable Optimization problem, and a subgradient of $\psi()$ in $\boldsymbol{r}$ is

$$g(\boldsymbol{r})_{ij}^h = \begin{cases} 0 & \text{if } x(\boldsymbol{r})_{ij}^h < r_{ij}^h \\ -\bar{c}_{ij}^h & \text{if } x(\boldsymbol{r})_{ij}^h = r_{ij}^h \end{cases}$$

where $\boldsymbol{x}(\boldsymbol{r})^h$ and $\bar{\boldsymbol{c}}(\boldsymbol{r})^h$ are respectively an optimal solution and *reduced costs* of the h-th (MCF), so that it can be solved by any suitable NDO algorithm, such as a subgradient method [Gr72] [KS77]. (RD) is similar to (DMMCF), but differs in a number of significant details:

- it has mk variables, while (DMMCF) has only m variables;
- its feasible region is more "complicated" than just the nonnegative orthant;
- the function $\psi()$ have value $+\infty$ if the capacities $\boldsymbol{r}$ make one of the (MCF)s unfeasible: furthermore, finding an initial feasible allocation $\bar{\boldsymbol{r}}$, such that $\psi(\bar{\boldsymbol{r}}) < +\infty$, is (in principle) a nontrivial task.

Nevertheless, computational experiences [AHK79] seem to show that $\psi()$ is an "easier" function than $\varphi()$, at least for the purpose of rapidly obtaining lower bounds with a subgradient algorithm: this can perhaps justify the continuing interest in this approach. For instance, an "inexact" Bundle method has been recently proposed [DGM93] where it is not necessary to compute $\psi()$ exactly, but no computational results are still available for that approach; alternatively, in [Ko94] some variants of (RD) are solved by means of the *Alternating Direction Method Of Multipliers*. ADMOM [PR55] [DR56] [GM75] [KM95] is a general method for the solution of problems in the form

$$\min\{\ f'(\boldsymbol{x}') + f''(\boldsymbol{x}'') : \boldsymbol{A}'\boldsymbol{x}' + \boldsymbol{b} = \boldsymbol{A}''\boldsymbol{x}''\ ,\ \boldsymbol{x}' \in \mathcal{R}^p\ ,\ \boldsymbol{x}'' \in \mathcal{R}^q\ \}$$

where $f'()$ and $f''()$ are both proper convex extended-valued functions: the main idea under ADMOM is to form the Augmented Lagrangean of the problem

$$\lambda_t(\boldsymbol{y}) = \min_{x',x''}\ \{\ f'(\boldsymbol{x}') + f''(\boldsymbol{x}'') + \boldsymbol{y}(\boldsymbol{A}'\boldsymbol{x}' + \boldsymbol{b} - \boldsymbol{A}''\boldsymbol{x}'') + t\left\| \boldsymbol{A}'\boldsymbol{x}' + \boldsymbol{b} - \boldsymbol{A}''\boldsymbol{x}'' \right\|_2^2\ \}$$

and to approach its maximization with a variant of the Method of Multipliers where, rather than calculating $\lambda_t(\boldsymbol{y})$ exactly, the minimization is performed on the $\boldsymbol{x}'$ and on the $\boldsymbol{x}''$ separately. That is, given a current point $(\bar{\boldsymbol{x}}', \bar{\boldsymbol{x}}'', \bar{\boldsymbol{y}})$ the algorithm sets

$$\bar{\boldsymbol{x}}' := \operatorname{argmin}_{x'}\ \{\ f'(\boldsymbol{x}') + (\bar{\boldsymbol{y}}\boldsymbol{A}')\boldsymbol{x}' + t\left\| \boldsymbol{A}'\boldsymbol{x}' + \boldsymbol{b} - \boldsymbol{A}''\bar{\boldsymbol{x}}'' \right\|_2^2\ \}$$

$$\bar{\boldsymbol{x}}'' := \operatorname{argmin}_{x''}\ \{\ f''(\boldsymbol{x}'') + (\bar{\boldsymbol{y}}\boldsymbol{A}'')\boldsymbol{x}'' + t\left\| \boldsymbol{A}'\bar{\boldsymbol{x}}' + \boldsymbol{b} - \boldsymbol{A}''\boldsymbol{x}'' \right\|_2^2\ \}$$

and then updates the dual multipliers with the usual formula

$$\bar{\boldsymbol{y}} := \bar{\boldsymbol{y}} + 2t(\boldsymbol{A}'\bar{\boldsymbol{x}}' + \boldsymbol{b} - \boldsymbol{A}''\bar{\boldsymbol{x}}'');$$

assuming that the two separate subproblems in $\boldsymbol{x}'$ and $\boldsymbol{x}''$ always have a solution, the method can be shown to converge (linearly, if *strong* convexity and a Lipschitz condition also hold) to an optimal solution. ADMOM is typically used in the *Lagrangean Decomposition* framework, where $\boldsymbol{x}'$ and $\boldsymbol{x}''$ are two *copies* of the same set of variables $\boldsymbol{x}$ and the linking constraints are $\boldsymbol{x}' = \boldsymbol{x}''$: obviously, this is sensible if the constraints set of the original problem can be partitioned in two parts such that separate optimization over each part is much easier than over their intersection. In the context of Resource Decomposition, different choices are possible, such as the Resource Proximization Splitting

$$f'(\boldsymbol{x}) = \sum_{h \in \mathcal{K}} \begin{cases} \boldsymbol{c}^h \boldsymbol{x}^h & \text{if } \boldsymbol{E}\boldsymbol{x}^h = \boldsymbol{b}^h,\ \boldsymbol{0} \le \boldsymbol{x}^h \le \boldsymbol{u}^h \\ +\infty & \text{otherwise} \end{cases} \qquad f''(\boldsymbol{r}) = \begin{cases} 0 & \text{if } \sum_{h \in \mathcal{K}} \boldsymbol{r}^h \le \boldsymbol{u} \\ +\infty & \text{otherwise} \end{cases}$$

with constraints $\boldsymbol{x} = \boldsymbol{r}$, or the more general Activity-and-Resource Proximization Splitting

$$f'(\boldsymbol{x}', \boldsymbol{r}') = \sum_{h \in \mathcal{K}} \begin{cases} \boldsymbol{c}^h \boldsymbol{x}'_h & \text{if } \boldsymbol{E}\boldsymbol{x}'_h = \boldsymbol{b}^h,\ \boldsymbol{0} \le \boldsymbol{x}^h \le \min\{\boldsymbol{u}^h, \boldsymbol{r}'_h\} \\ +\infty & \text{otherwise} \end{cases}$$

$$f''(\boldsymbol{x}'', \boldsymbol{r}'') = \begin{cases} 0 & \text{if } \sum_{h \in \mathcal{K}} \boldsymbol{r}''_h = \boldsymbol{u} \\ +\infty & \text{otherwise} \end{cases}$$

with constraints $\boldsymbol{x}' = \boldsymbol{x}''$, $\boldsymbol{r}' = \boldsymbol{r}''$: each choice has its advantages and drawbacks, concerning the number of variables, the convergence properties and the actual computational behaviour, but in all cases the computation of the functions can be decomposed per commodity, and hence parallelized. As in the Augmented Lagrangean, the subproblems to be solved are Quadratic (MCF)s, for which efficient codes have been devised. A number of improvements of the basic ADMOM scheme can be devised, such as a *variable penalty term* t and/or a more general matrix-induced norm $\|\cdot\|_H^2$ with a *variable penalty matrix* $\boldsymbol{H}$ in the Augmented Lagrangean term: it is also usually benefical to use *over-relaxation steps*, that is (for the case $\boldsymbol{x}' = \boldsymbol{x}''$)

$$\tilde{\boldsymbol{x}}'' := \operatorname{argmin}_{\boldsymbol{x}''} \left\{ f''(\boldsymbol{x}'') + \bar{\boldsymbol{y}}\boldsymbol{x}'' + t \left\| \omega\bar{\boldsymbol{x}}' + (1 - \omega)\bar{\boldsymbol{x}}'' - \boldsymbol{x}'' \right\|_2^2 \right\}$$

$$\bar{\boldsymbol{y}} := \bar{\boldsymbol{y}} + 2t(\omega\bar{\boldsymbol{x}}' + (1 - \omega)\bar{\boldsymbol{x}}'' - \tilde{\boldsymbol{x}}'')$$

$$\bar{\boldsymbol{x}}'' := \tilde{\boldsymbol{x}}''$$

where $\omega \in (0, 2)$. Only a few comparisons with alternative approaches are known, and the existing results [LMK94] are not very encouraging: intuitively, RD methods should be competitive on tightly coupled problems with few commodities, but it is not clear whether this is really the case.

II.1.4.4 Interior Point methods

Interior Point algorithms have grown a favourable reputation for the solution of very large, sparse Linear Programs: such methods require, at the generic i-th iteration, the computationally expensive solution of a linear system on $\boldsymbol{A}\Theta_i^{-2}\boldsymbol{A}^T$, where $\boldsymbol{A}$ is the constraint matrix of the (LP) and $\Theta_i > 0$ is a diagonal matrix, but, on the other hand, the number of iterations needed to reach convergence is usually very small, achieving a good overall complexity. In its node-arc formulation, (MMCF) is a large (LP) with no more than three nonzeroes per columns, i.e. apparently a good candidate for IP methods: however, direct application of the "standard" techniques requires forming the $(nk + m)$ matrix $\boldsymbol{A}\Theta_i^{-2}\boldsymbol{A}^T$, an expensive task in both memory and time due to the fill-in effect. To circumvent

these difficulties, specialized IP methods have been proposed [CG90] [KKR94] [AS95] [CN96] where, in the spirit of Basis Partitioning methods for the simplex (Cf. Sec. II.1.4.1), the computationally intensive operations are specialized in order to exploit the structural properties of the problem at hand: in this case, one has

$$\boldsymbol{A}\Theta_i^{-1} = \begin{bmatrix} \boldsymbol{B}' \\ \boldsymbol{D}' \end{bmatrix}\Theta_i^{-1} = \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{D} \end{bmatrix} \quad \text{where} \quad \boldsymbol{B}' = \begin{bmatrix} \boldsymbol{E} & & \boldsymbol{0} \\ & \ddots & \\ \boldsymbol{0} & & \boldsymbol{E} \end{bmatrix} \quad \text{so that} \quad \boldsymbol{A}\Theta_i^{-2}\boldsymbol{A}^T = \begin{bmatrix} \boldsymbol{B}\boldsymbol{B}^T & \boldsymbol{B}\boldsymbol{D}^T \\ \boldsymbol{D}^T\boldsymbol{B} & \boldsymbol{D}^T\boldsymbol{D} \end{bmatrix}$$

and $\boldsymbol{B}\boldsymbol{B}^T$ is a block-diagonal matrix with k blocks of dimension n. Hence, systems on $\boldsymbol{B}\boldsymbol{B}^T$ can be (concurrently) decomposed in the k much smaller blocks, and this can be exploited in several different ways: the overall system can be solved by means of "structured" Cholesky decomposition [CG90] [VC93] [AS95], with Preconditioned Conjugated Gradient methods [KKR94] or by mixtures of the two, as in [CN96] where a Cholesky decomposition of the k blocks is kept and, to avoid forming $\boldsymbol{B}\boldsymbol{D}^T(\boldsymbol{B}\boldsymbol{B}^T)^{-1}$, the "small" $m \times m$ system is attacked with a PCG where at each step a system on $\boldsymbol{B}\boldsymbol{B}^T$ is solved. These approaches easily extend to structured (LP)s with "complicating" variables [JYZ94], such as the Recourse Linear Program in stochastic programming, and even to (LP)s with both "linking" columns and rows [AS95]; also, specialized IP methods currently attain the best polynomial bound for the *exact* solution of (MMCF), $O(k^{3.5}n^3m^{1/2}\log(nBU))$ where B and U are the maximum demand and capacity respectively [KV86], even though this is a theoretical result only because it requires the fast matrix multiplication algorithms.

A different IP approach is developed in [SM90a] [SM90b] [MS91], where separability is exploited in a Cost Decomposition fashion: the method is based on a *shifted logarithmic barrier function*, i.e. the *relaxed* constraints $\Sigma_{h \in \mathcal{K}} \boldsymbol{x}^h \le \boldsymbol{r}$ (with $\boldsymbol{r} \ge \boldsymbol{u}$) are moved to the objective function with a classical logarithmic barrier term, yielding

$$(\mathrm{BP}_{\boldsymbol{r},t}) \qquad \min\{ f_{\boldsymbol{r},t}(\boldsymbol{x}) : \boldsymbol{E}\boldsymbol{x}^h = \boldsymbol{b}^h , \boldsymbol{0} \le \boldsymbol{x}^h \le \boldsymbol{u}^h \ \forall\, h \in \mathcal{K} \}$$

$$\text{where} \qquad f_{\boldsymbol{r},t}(\boldsymbol{x}) = \Sigma_{h \in \mathcal{K}} \boldsymbol{c}^h\boldsymbol{x}^h - t(\Sigma_{(i,j) \in \mathcal{A}} \ln(r_{ij} - \Sigma_{h \in \mathcal{K}} x_{ij}^h)).$$

The shifted barrier is used to make any initial point $\bar{\boldsymbol{x}}$, satisfying the flow conservation and individual capacity constraints, interior w.r.t some $\boldsymbol{r} > \Sigma_{h \in \mathcal{K}} \bar{\boldsymbol{x}}^h$: then, $\bar{\boldsymbol{x}}$ is used as a starting point for an ad-hoc IP method, that *approximately* solves $(\mathrm{BP}_{\boldsymbol{r},t})$. The method computes a search direction $\boldsymbol{d}$ as the solution of the k (MCF)s

$$\min\{ \nabla f_{\boldsymbol{r},t}(\bar{\boldsymbol{x}})^h \boldsymbol{d}^h : \boldsymbol{E}(\boldsymbol{d}^h + \bar{\boldsymbol{x}}^h) = \boldsymbol{b}^h , \bar{\boldsymbol{l}}^h \le \boldsymbol{d}^h \le \bar{\boldsymbol{u}}^h \}$$

where the bounds $\bar{\boldsymbol{l}}^h$ and $\bar{\boldsymbol{u}}^h$ define a "trust region" around $\bar{\boldsymbol{x}}$, and are chosen to enforce the feasibility; then, $\boldsymbol{d}$ can either be used in a standard line search, yielding an extremely fast "coordination phase", or in the more complex k-dimensional search

$$\min\{ f_{\boldsymbol{r},t}([\bar{\boldsymbol{x}}^1 + \delta^1\boldsymbol{d}^1 \,..\, \bar{\boldsymbol{x}}^k + \delta^k\boldsymbol{d}^k]) : \boldsymbol{0} \le \bar{\boldsymbol{x}}^h + \delta^h\boldsymbol{d}^h \le \boldsymbol{u}^h \ \forall\, h \in \mathcal{K} \}$$

that results in faster overall convergence [ZaG95]: this basic step is repeated until a prescribed optimality criteria is satisfied. The solution $\tilde{\boldsymbol{x}}$ may not be feasible w.r.t. the original capacities $\boldsymbol{u}$, but it will be such that $\Sigma_{h \in \mathcal{K}} \tilde{\boldsymbol{x}}^h < \boldsymbol{r}$ so that $\boldsymbol{r}$ can be "squeezed" towards $\boldsymbol{u}$ while keeping $\tilde{\boldsymbol{x}}$ in the interior: the process is then repeated until feasibility ($\boldsymbol{r} = \boldsymbol{u}$) is attained. Then, an optimality phase starts where the barrier parameter t is gradually decreased until an optimal point is reached: the same algorithm is

used for the solution of ($BP_{r,t}$), with the only difference that t is decreased after each iteration instead of waiting for an optimal solution to be achieved.

II.1.4.5 Primal-Dual methods

Primal-Dual methods are a well-known class of algorithms for single-commodity (MCF) problems: a particular variant, the Relaxation approach, is among the most effective approaches to (MCF) in practice. A Primal-Dual method solves the general equality-constrained, separable convex problem

(ESSC) $\min_{\boldsymbol{x}} \{ \sum_{a \in \mathcal{A}} c_a(x_a) : \boldsymbol{E}\boldsymbol{x} = \boldsymbol{b} , \boldsymbol{0} \le \boldsymbol{x} \le \boldsymbol{u} \}$

(where now $\mathcal{A}$ is just the set of variables) by relaxing in a Lagrangean fashion the equality constraints

(PR_π) $\psi(\pi) = \min_{\boldsymbol{x}} \{ \sum_{a \in \mathcal{A}} c_a(x_a) + (\pi\boldsymbol{E})\boldsymbol{x} : \boldsymbol{0} \le \boldsymbol{x} \le \boldsymbol{u} \} - \pi\boldsymbol{b}$

and approaching the corresponding Lagrangean Dual

(PD) $\max_{\pi} \{ \psi(\pi) : \pi \in \mathbb{R}^n \}$;

in the (MCF) case, π is usually referred to as the vector of *node potentials*. $\psi()$ is a concave function, and its subdifferential is (Cf. §I.1.2)

$$\partial\psi(\pi) = \{ \boldsymbol{E}\boldsymbol{x} - \boldsymbol{b} : \boldsymbol{x} \in X(\pi) \}$$

where $X(\pi)$ is the set of *all* optimal solutions of (PR_π): if the cost functions $c_{ij}()$ are *strictly* convex, then (PR_π) has *unique* optimal solution $\boldsymbol{x}(\pi)$ and $\psi()$ is a smooth function, since $\partial\psi(\pi) = \{ \boldsymbol{E}\boldsymbol{x}(\pi) - \boldsymbol{b} \} = \{ \nabla\psi(\pi) \}$. Hence, it is possible to solve (PD) with any smooth optimization approach: Relaxation methods [TB91] choose the *Coordinate Ascent* one, i.e. maximize one π_i variable at a time until the optimum is reached. It is easy to see that the (univariate) maximization problem on π_i is at its optimum if and only if the i-th constraint $\boldsymbol{E}_i\boldsymbol{x}(\pi) = \boldsymbol{b}_i$ is satisfied: hence, Relaxation methods satisfy one violated row of the dualized constraints at a time, and properly set the primal variables by adjusting the corresponding Lagrangean variable, that is used as a "knob".

Relaxation methods can be directly applied to (MMCF) with strictly convex costs [NFT90]: in this case, the vector of potentials is $\Pi = [\pi^1 , .. , \pi^k , \boldsymbol{y}]$, where π^h are the node potentials for the h-th commodity while $\boldsymbol{y}$ are the m Lagrangean variables associated to the mutual capacity constraints (turned into equations by adding slacks). If the cost functions are *Bregman Distances*, such as the quadratic function x^2 and the *entropic* function $x\ln(x)$, the strictly related *Row Action Method* [CL81] can also be used: RAM works on the different Lagrangean Dual

$$\max_{\Pi,\lambda,\mu} \{ \min_{\boldsymbol{x}} \{ \sum_{(i,j) \in \mathcal{A}} c_{ij}(x_{ij}) + (\pi\boldsymbol{E})\boldsymbol{x} - \lambda\boldsymbol{x} + \mu\boldsymbol{x} \} - \pi\boldsymbol{b} - \mu\boldsymbol{u} \}$$

where also the bound constraints $\boldsymbol{0} \le \boldsymbol{x} \le \boldsymbol{u}$ have been relaxed, and can obtain the optimal value of the chosen dual variable (π_i , λ_i or μ_i) and the corresponding primal solution $\boldsymbol{x}(\Pi , \lambda , \mu)$ with a closed form expression, whereas Relaxation methods need an iterative procedure.

These methods can be streamlined to work with "local" data only, that in the (MCF) case corresponds to the set of arcs incident in each node, so that they are well suited for fine-grain parallelization: Relaxation methods admit both synchronous (data-parallel) and asynchronous (control-parallel) implementations [BT89], while RAM can only be implemented in a synchronous fashion [CCZ91] [Ze91] [Ze93].

For the *non strictly* convec case, matters are complicated by the nondifferentiability of $\psi()$, i.e. by the fact that $X(\ \Pi\ )$ is no more necessarily a singleton: for the single-commodity (MCF), however, Primal-Dual methods can still be constructed [Be91], as described in some detail in §II.4.3. Briefly, these algorithms alternate a *primal phase*, where a Max-Flow problem is (implicitly) solved on the *reduced graph*, and a *dual phase* where the (characteristic vector of the) optimal Min-Cut(set) is used as an *elementary* dual ascent direction to change π. Unfortunately, no equivalent primal phase (i.e. no combinatorial Multicommodity Max-Flow algorithm) is known for (MMCF): although Relaxation methods can be generalized to Linear Programs [TB87], and even to the more general class of *Monotropic Programs* [Ro84], the result uses of the *Painted Index Algorithm* (for constructing the equivalent of the *augmenting paths*) that lacks a clear interpretation in terms of Multicommodity Flows. A specialized Relaxation method for block-structured (LP) have been proposed in [Ts91], with the explicit aim of providing an efficient approach to (MMCF): for the block-structured case, the Painted Index Algorithm can be streamlined and implemented by combinatorial operations on properly defined graph structures, but the resulting algorithms (from a private communication by the author) do not obtain particularly encouraging computational results.

II.1.4.6 ε-approximation methods

Starting with the seminal work [LR88] on the All-Pair Unit Demands problem (Cf. §II.1.2), a number of ε-approximation algorithms have been recently developed for Multicommodity Max-Flow problems: other than quickly providing good solutions, they can also be used to prove "approximate" Max-Flow / Min-Cut theorems. For instance, it can be shown [KRA90] [PT93] that, if the minimal capacity-to-separated-demand ratio is at least $O(\log^2 k)$, than a feasible multiflow can be *constructed*, while the required ratio is reduced to $O(\log k)$ when considering multicuts [GVY93] (i.e. Max-Sum rather than Concurrent problems): both these results can be improved if the graph has special properties, such as planarity [KPR93] [TV94]. Much weaker results hold for directed graphs: this can be seen by considering the n-commodity problem on a circle with arcs $(i, i + 1)$ of capacity u and one unit of commodity i to be sent from $i + 1$ to i. Here, the capacity-to-separated-demand of each cut is u, but it must be $u \geq n - 1$ in order for a feasible flow to exist, since each commodity must traverse $n - 1$ arcs to reach its destination: hence, the Max-Flow / Min-Cut gap can be in general as bad as $O(k)$, while an $O(\log^3 k)$ bound can be proved if the *demands* are symmetric [KPT93], i.e. if for any commodity having source i and sink j there exists another one having source j and sink i.

The ε-approximation algorithms are all based on the idea, introduced in [SMa90] for the All-Pair Unit Demands problem, of rerouting a(n entire) commodity at a time from "highly congested" to "lightly congested" arcs: in [LMP91], for instance, this is done by defining arc lengths $l_{ij} = e^{\alpha\lambda(i,j)} / u_{ij}$, where $\lambda(i, j) = \sum_{h \in \mathcal{K}} x_{ij}^h / u_{ij}$ is the *congestion* of arc (i, j), and rerouting some flow from a "long" path to a "short" one. These algorithms admit both deterministic and randomized versions [Go92] with lower (expected) complexity: actual implementations [LPS93] can be practical for finding solutions as long as the required accuracy is not too high, e.g. ε not smaller than 10^{-3}.

Building on this, ε-approximation algorithms have been developed for the more general Fractional Packing [PST91] and Resource Sharing problems [GK94a] (Cf. §I.1.2): although different in the details and the analysis, they are all based on an exponential penalty function analogous to the above. In extreme synthesis, given - with the notation of (FP) - an unfeasible point $\bar{\boldsymbol{x}} \in X$, the basic steps

$$\bar{y}_i = \frac{e^{\alpha(\boldsymbol{A}_i \bar{\boldsymbol{x}} / u_i)}}{u_i} \qquad \forall\, i$$

$$\tilde{\boldsymbol{x}} = \operatorname{argmin}_{\boldsymbol{x}} \{ (\bar{\boldsymbol{y}}\boldsymbol{A})\boldsymbol{x} : \boldsymbol{x} \in X \}$$

$$\bar{\boldsymbol{x}} = \omega\bar{\boldsymbol{x}} + (1 - \omega)\tilde{\boldsymbol{x}}$$

are repeated, together with suitable adjustments of the parameters, until an (approximately) feasible solution is obtained: as in Cost Decomposition methods, the gradient of the penalty function "direct" the subproblem solver in the search for a better point. For the block-structured case, the convex combination may be limited to the only blocks that have produced "good" new proposals $\tilde{\boldsymbol{x}}^h$: in the randomized variants, only one randomly chosen block is solved and modified at each iteration. Analogous techniques can be devised for (RS), that can in turn be used for solving the Block Structured convex program of §I.1.1 [GK95a] (hence (MMCF)): as in [MS91], the idea is a two-phase approach where first a feasible point is obtained, and then the objective function is turned into a constraint for the actual optimization. These methods require $\tilde{O}(\varepsilon^{-2} k \log(m))$ iterations, where $\tilde{O}(\cdot)$ means "up to polylog factors", to achieve an accuracy of ε, yielding a $\tilde{O}(\varepsilon^{-2} knm)$ bound [GK95b] for the ε-approximate solution of (MMCF): if ε is not too small, this compares favourably with the $O(k^{2.5} n^3 m^{1/2} \log(\varepsilon^{-1} nBU))$ bound of Interior Point methods [KV86]. Related algorithms also exist [VG96], based on a *logarithmic* barrier function, that achieve an accuracy of ε in $\tilde{O}(\varepsilon^{-2} m)$ iterations, therefore being more effective when $k >> m$: the combination of logarithmic *and* exponential barrier methods gives an almost optimal algorithm in terms of *coordination complexity* [GK94b] (the cost of the coordination phase), possibly indicating that those methods are well-suited for parallelization.

II.1.4.7 Special cases and theoretical results

Due to space limitations, a number of further results about Multicommodity Flows can only be rapidly sketched here: they are just intended to make it apparent that this class of problems have attracted many researchers from many different viewpoints.

It is known from more than 40 years than the special case of $k = 2$ admits polynomial approaches that do not generalize even to $k = 3$ [RW66a] [RW66b] [Sa66] [Hu63] [Hu64] [Hu69]: research on two-commodity flows has continued [Ra80] [Ra94], and results have been obtained about two-commodity flows in planar graphs [KP93] and the relationships with the integrality property [GP92].
Interesting theoretical results can be obtained if the underlying graph G has some special properties [Ev78] [EM78] [NI89] [NI90] [EC95]: for instance, elegant theoretical results [BGJ70] [EJD77] [Se81] [AD91] [CH92] have been obtained that describe the polyhedral properties of *undirected* multiflows and the so-called "Cut Cone", or connections with the theory of matroids. Also, polynomial algorithms [Lo85] [Ka94] have been found for generalizations of the undirected All-Pair problem; minor results have been obtained for the directed case also [Je66a] [JT72].
Finally, sequential [Pa81] [Pa84] [GMY92] and parallel [NK89] approaches have been devised for Multicommodity Flow problems with *equilibrium* constraints: although these problems have a relevant interest in fields like economic planning and transportation systems analysis and design, the required techniques are out of the scope of the present work.