ISTITUTO DI ANALISI DEI SISTEMI ED INFORMATICA
"Antonio Ruberti"
CONSIGLIO NAZIONALE DELLE RICERCHE

A. Frangioni, C. Gentile

EXPERIMENTS WITH A HYBRID INTERIOR POINT/COMBINATORIAL APPROACH FOR NETWORK FLOW PROBLEMS

R. 640 Aprile 2006

Antonio Frangioni – Dipartimento di Informatica, Largo B. Pontecorvo 3, 56125 Pisa (Italy). Email: `frangio@di.unipi.it`.

Claudio Gentile – Istituto di Analisi dei Sistemi ed Informatica del CNR, Viale Manzoni 30 - 00185 Roma, Italy. Email: `gentile@iasi.cnr.it`; this author has been partially supported by the UE Marie Curie Research Training Network no. 504438 ADONET.

ISSN: 1128–3378

Collana dei Rapporti dell'Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti", CNR

viale Manzoni 30, 00185 ROMA, Italy

tel. ++39-06-77161
fax ++39-06-7716461
email: iasi@iasi.rm.cnr.it
URL: http://www.iasi.rm.cnr.it

Abstract

Interior Point (IP) algorithms for Min Cost Flow (MCF) problems have been shown to be competitive with combinatorial approaches, at least on some problem classes and for very large instances. This is in part due to availability of specialized crossover routines for MCF; these allow early termination of the IP approach, sparing it with the final iterations where the KKT systems become more difficult to solve. As the crossover procedures are nothing but combinatorial approaches to MCF that are only allowed to perform few iterations, the IP algorithm can be seen as a complex "multiple crash start" routine for the combinatorial ones. We report our experiments about allowing one primal-dual combinatorial algorithm to MCF to perform as many iterations as required to solve the problem after being warm-started by an IP approach. Our results show that the efficiency of the combined approach critically depends on the accurate selection of a set of parameters among very many possible ones, for which designing accurate guidelines appears not to be an easy task; however, they also show that the combined approach can be competitive with the original combinatorial algorithm, at least on some "difficult" instances.

Key words: Min Cost Flow problems, Interior Point algorithms, Relaxation Method, Crossover

1. Introduction

The Min Cost Flow (MCF) problem is the following Linear Program (LP)

$$\min \{ cx \ : \ Ex = b \ , \ 0 \leq x \leq u \} \ , \tag{1}$$

where E is the node-arc incidence matrix of a network $G = (N, A)$, c is the vector of arc costs, u is the vector of arc upper capacities, b is the vector of node deficits, and x is the vector of flows. This problem has a huge set of applications, either in itself or—more often—as a submodel of more complex and demanding problems. This is testified by the enormous amount of research that has been invested in defining efficient solution algorithms for MCF, either by specializing LP algorithms, such as the simplex method, to the network case, or by developing ad-hoc approaches [1].

Interior Point (IP) methods for Linear Programming have grown a well-established reputation as efficient algorithms for large-scale problems. Specialized IP algorithms for MCF, using iterative approaches for the solution of the "core" KKT systems which represent the critical computational task, have been shown [17, 16, 15] to be competitive with "combinatorial" approaches, like primal-dual [3] cost-scaling [10] algorithms or simplex methods [12, 1], albeit not on all problem classes and especially for very large instances. In [16], for instance, a dual IP approach was consistently outperforming the primal-dual combinatorial code `RELAXT-3` [4] on four problems classes on 13, was competitive for graphs of large size on other two classes, but it was slower (sometimes by a high margin) on the seven remaining classes. In [15], a more advanced primal-dual IP approach is shown to be competitive with the primal-dual cost-scaling combinatorial code `CS` [10] for very large graphs on two problem classes on seven, while being constantly outperformed on the other five. In both papers, simplex-based methods like `NETFLO` [12] and `CPLEX NETOPT 4.07` were shown to be, in general, less competitive with the IP methods, being outperformed almost always at least as the size of the network increases.

Closely examining the results (e.g., Table 3 in [16]) reveals that an important contribution to the overall efficiency of the IP approach is provided by the *specialized crossover* routines that can be implemented for MCF algorithms. These attempt to construct an optimal (non necessarily basic) solution out of the information provided by the IP approach; if successful, the whole algorithm is stopped. This actually happens relatively early in the optimization process, sparing the IP approach with several of the final iterations and therefore substantially improving its performances, since:

- IP approaches typically perform "few" iterations, so avoiding a handful of iterations already provides a consistent relative decrease of the running times;
- the "core" KKT system becomes more and more ill-conditioned, and therefore difficult to solve by iterative approaches, as the IP algorithm nears an optimal solution;
- the KKT system can be approximately solved during the IP algorithm, thereby making iterative methods an attractive option, but the required accuracy has to be increased as the IP algorithm nears an optimal solution.

Because crossover procedures are typically simplex-like approaches restricted to only a few (possibly one) iterations [13, 2], one may restate the above observations as: in the MCF case, the "continuous" IP approach is useful to quickly providing an extremely good starting point to a simplex algorithm, which then exploits the combinatorial structure of the problem to quickly

"finish it off". In other words, the efficiency of IP approaches is due to their capacity of quickly identifying information which the simplex algorithm can use to reach an optimal solution without a combinatorial explosion of pivot moves.

While for general LPs the simplex method is the only possible efficient combinatorial companion, for MCF other alternatives exist; indeed, as previously recalled, simplex-like approaches are typically outperformed by primal-dual combinatorial ones. This suggested us to experiment with the following generalization of the above idea: combine an IP method and a combinatorial (non-simplex) algorithm for MCF problems, using the former to produce warm-starting information for the latter.

Thus, the aim of our study is to verify whether there are choices of the many forms of IP algorithm, and their many algorithmic parameters, such that the combined process is more effective than the original combinatorial approach. We show by extensive experiments that, at least for one primal-dual algorithm, the results critically depend on the accurate selection of some parameters among very many possible choices, for which designing accurate guidelines appears not to be an easy task. However, the results also show that the hybrid approach can be better than the original one, especially on some classes of "difficult" instances.

The structure of the paper is the following: in Section 2 the basic ingredients of IP algorithms are briefly recalled, and the relevant algorithmic issues are discussed, focusing in particular on the MCF case. In Section 3, primal-dual approaches to MCF are rapidly sketched, and the way in which information can be exchanged between an IP approach and a primal-dual one is discussed. In Section 4 the results of a computational experience, aimed at assessing the effectiveness of the hybrid method, are presented, and conclusions are drawn.

2. Interior Point algorithms

Interior Point algorithms for MCF can be described by considering (1) together with its dual

$$\max \{\ yb - wu\ :\ yE + z - w = c\ ,\ z, w \geq 0\ \} \quad , \tag{2}$$

where y, z and w are respectively the dual variables of the flow conservation constraints $Ex = b$, the dual slacks and the dual variables of the box constraints $x \leq u$. Different IP algorithms can be constructed, which all start from a "slackened" version of the *KKT optimality conditions* of the above pair of dual problems: this comprises, other than the usual primal and dual feasibility conditions, the following approximated form of the *complementary slackness conditions*

$$x_{ij} z_{ij} = \mu \qquad (u_{ij} - x_{ij}) w_{ij} = \mu \qquad (i,j) \in A \tag{3}$$

were $\mu \geq 0$ is a parameter. The parametric solution to the system defines the *central path*, a continuous trajectory which, as μ tends to 0, converges to a "central" pair of primal and dual optimal solutions.

Path-following (also known as *barrier*) algorithms attempt to reach close to these optimal solutions by following the central path; this is done by performing a damped version of Newton's iteration applied to the (nonlinear) slackened KKT system for some value of μ, and then reducing μ to attain (fast) global convergence. These algorithms can be divided into *primal*, *dual* or *primal-dual* depending on the fact that only the primal solution, only the dual solution, or both at the same time are modified during the Newton step. Several variants of the above methods have been defined. For instance, in Mehrotra's *predictor-corrector* [14] variant of the primal-dual method, multiple iterations of the Newton approach are performed to compute the

next direction at every IP iteration. Conversely, in *affine* variants of the IP algorithms the formulae for the Newton direction are taken as the limit for $\mu \rightarrow 0$ of the formulae in the path-following case; this makes computations slightly faster (due to the reduction of constant factors, not of the asymptotic complexity) and eliminates μ from the formulae, thereby avoiding the problem of tuning its decrease, possibly at the cost of some numerical stability. A more detailed description of IP algorithms is clearly outside the scope of the present paper; the interested reader can consult the extensive literature on the subject, comprised many recent LP textbooks, e.g., [18, 19].

2.1. Solving the core systems

Remarkably, all the formulae for all the variants of the IP method boil down to a set of linear operations plus one (or more) solution(s) of a "core" linear system of the form

$$(E\Theta E^T)\Delta y = d \quad , \tag{4}$$

where Θ and d are respectively an $m \times m$ diagonal matrix ($m = |A|$) with positive entries and a vector of $\mathbb{R}^n$ ($n = |N|$), both depending on the current iteration and on the specific IP variant chosen. The solution of (4) typically represents by far the main computational burden of the IP algorithms. General-purpose LP solvers typically use direct methods, such as the Cholesky factorization, to solve (4); however, for structured LPs—like MCF—iterative approaches, such as Preconditioned Conjugate Gradient (PCG) methods, have shown to be competitive [5, 17, 16, 15], provided that appropriate preconditioners are employed.

The most widely used preconditioners are *subgraph-based* ones, which select a large-weight "triangulated" subgraph S of G—the arc weights being the diagonal elements of Θ—so that the restricted matrix $E_S\Theta_S E_S^T$ is very easy to invert; possible classes of triangulated graphs are trees [16, 15, 11] or "Brother-connected trees" [8]. These preconditioners usually tend to become more and more efficient as the IP algorithm proceeds, since the IP solution tend to more and more closely resemble a basic solution; hence, the arc weights Θ tend to "concentrate" on the arcs of the basic tree, which are easily selected by the preconditioner, so that the total weight of the columns not "covered" by the preconditioner tend to become negligible. On the other hand, (4) become more and more ill-conditioned, and therefore difficult to solve, as the IP algorithm proceeds, partly counterbalancing the positive effect of the better preconditioner. Furthermore, while (4) can be only approximately solved during the IP algorithm, thereby making iterative methods an especially attractive option, the required accuracy has to be increased as the IP algorithm nears an optimal solution. The combination of all these effects results in complex fluctuations of the "difficulty" of (4), with some sequences of IP iterations, especially—but not only—towards the end of the IP algorithm, showing relatively "hard" systems to solve. Samples of this behavior can be seen in Table 1, where we report some data about the number of iterations required to solve problems of two different sizes for three different classes of networks; the exact details about the networks are not relevant here, and they will be explained in Section 4. For each network type and size, we report seven rows corresponding to the systems solved at IP iterations 0, 1, $k/4$, $k/2$, $3k/4$, $k-1$ and k, where k is the index of the last iteration and 0 is the system solved for the crash-start (see Section 2.3); this is a significant sample of the systems solved during the IP algorithm. The table clearly shows that significant variations on the difficulty of solving (4) by an iterative approach have to be expected, especially towards the end of the IP algorithm.

Table 1: PCG iterations at various stages of the IP process

	goto		grid		net	
	12.8	12.256	12.8	12.256	12.8	12.256
0	135	379	22	8	15	7
1	40	10	17	8	14	8
$k/4$	65	107	17	11	12	10
$k/2$	42	96	36	13	16	13
$3k/4$	23	67	27	14	11	14
k-1	17	30	10	17	6	18
k	17	30	2	18	6	19

2.2. Crossover

Early termination of IP methods can be obtained by means of "crossover" procedures: as the IP method converges towards an optimal solution, information can be extracted about the set of active (primal and dual) constraints at optimality, thereby being finally able to build an optimal base. For general LPs, crossover procedures have been primarily developed for two different purposes:

- being able to show that the algorithm can be *finitely* stopped attaining an "exact" optimal solution in polynomial time;
- being able to combine IP approaches with traditional simplex methods, in order to exploit the superior reoptimization capabilities of the latter which are especially crucial e.g. in Branch & Bound algorithms.

That is, for the general LP case the crossover procedure is not perceived as an "early termination" rule: the IP algorithm is brought at convergence with very high precision (say, `1e-8` relative) and the crossover is only performed if the additional features of a basic solution (comprised the more accurate precision of, say, `1e-12` relative) are required.

However, for MCF the situation is different: in [16, 15], the IP approach is *always* stopped by the crossover rather than by the standard stopping rules of the IP algorithm. This is also due to the fact that for MCF a "simplex crossover" similar to that of general LP [13, 2] is very cheap: as mentioned in the previous paragraph, the maximum-weight spanning tree of G with arc weights Θ, which is already computed by the preconditioners, towards termination provides, possibly with minimal variations, a primal and dual feasible basis for MCF. Being so inexpensive, the procedure can be repeated at every iteration (of the final sequence), discarding the results if feasibility is not achieved.

Furthermore, for MCF "non simplex" crossover procedures have also been presented; these are strongly related to primal-dual combinatorial algorithms for the problem, and therefore they will be discussed later on.

2.3. Crash start of the IP algorithms

Like every iterative approach, IP algorithms require a starting point ("crash solution") to initiate with. This has to be a primal solution x for a primal method, a dual solution (y, z, w) for a dual method, or both for a primal-dual method. Since the IP algorithms naturally cope with

infeasibilities, these do not need to be primal or dual feasible. Not much is said, in the literature, about how to choose these solutions: (many different variants of) simple formulae are reported as "working", with little or no discussion of the alternatives. This may be due to the fact that IP algorithms—especially primal-dual ones—are generally very robust, so that their overall performances are only marginally impacted by this choice.

However, even if this is true in the "standard" context, crash formulae are set to have a much larger impact in our application, since the IP algorithm is (very) early terminated, and therefore it may not have time enough to "neutralize" a bad choice of the initial solutions. Thus, we resorted to collecting all proposals of crash solutions we could find and testing all possible combinations. In general, the primal solution x, the dual solution y and the dual slacks (z, w) can be chosen almost independently, although there are cross-dependencies: typically (z, w) are chosen after y in order to have $yE + z - w = c$. We now report on the variants that we implemented and tested.

Dual crash. For the y variables, the following four rules have been proposed:

D1) $y = 0$,

D2) $y = (EE^T)^{-1}Ec$,

D3) $y = b\|c\|_\infty / \|b\|_\infty$,

D4) $y = (EE^T)^{-1}(2\tau b + E(c - \tau u))$,

where τ is a fixed parameter. Rule D2 amounts at choosing as y those that give the (orthogonal complement of the) projection of c on the $Ex = 0$ subspace, while D4, known as (the dual part of) "Mehrotra's rule", chooses y as the dual solution of the unconstrained problem $\min \{ cx + (\tau/2)(||x||^2 + ||u - x||^2) : Ex = b \}$; the other two rules are just "quick and dirty" initializations. After having fixed y, the vector $\bar{c} = c - yE$ is the residual of the dual constraints that need to be zeroed if a dual feasible solution is sought for; this is typically used in the formulae for x and (z, w).

Primal crash. For the x variables, the following five rules have been proposed:

P1) $x_{ij} = \tau u_{ij}$,

P2) $x_{ij} = \tau u_{ij}$ if $\bar{c}_{ij} \geq 0$, $x_{ij} = (1 - \tau)u_{ij}$ otherwise,

P3) $x = E^T(EE^T)^{-1}b$,

P4) $x = \frac{1}{2\tau}(\tau u - \bar{c})$,

P5) $x_{ij} = \frac{2\mu u_{ij}}{2\mu + \bar{c}_{ij}u_{ij} + \kappa_{ij}}$, where $\kappa_{ij} = \sqrt{(2\mu)^2 + (\bar{c}_{ij}u_{ij})^2}$ if $\bar{c}_{ij} \leq 0$ and its opposite otherwise,

where $\tau > 0$ is fixed. Rule P3 just amounts at finding the least-norm solution of $Ex = b$, rule P4 is the "primal part" of rule D4, while rule P5 is obtained by asking that x, z and w satisfy the slackened complementary slackness conditions, except $Ex = b$, for the given value of μ; the first two rules are just "quick and dirty" initializations. For all the cases where x is not guaranteed to be bound-feasible ($x \leq u$), each of the above formulae can be modified (as in P2 with $\tau < 1$) in order to make it so, thus nearly doubling the number of alternatives.

Dual slack crash. For (z, w), the following three rules have been proposed:

S1) if $\bar{c}_{ij} \geq 0$ then $w_{ij} = \sigma$ and $z_{ij} = \bar{c}_{ij} + \sigma$, otherwise $z_{ij} = \sigma$ and $w_{ij} = \sigma - \bar{c}_{ij}$,

S2) $z_{ij} = \sigma/x_{ij}$, $w_{ij} = \sigma/(u_{ij} - x_{ij})$,

S3) $z_{ij} = \frac{2\mu+\bar{c}_{ij}u_{ij}+\kappa_{ij}}{2u_{ij}}$, $w_{ij} = \mu z_{ij}/(u_{ij}z_{ij} - \mu)$, where κ_{ij} is as in rule P5,

where σ is fixed. Rule S3 is the dual slack part of rule P5, while the first two rules are just "quick and dirty" initializations; note that not all of the above formulae produce a dual feasible solution.

3. Primal-dual algorithms for MCF

In the following, we briefly describe the characteristics of primal-dual algorithms for MCF that are relevant to our application; for a more detailed description, the reader is referred to [1] and [4].

Any m-vector x such that $0 \leq x_{ij} \leq u_{ij}$ for each $(i,j) \in A$ is a *pseudoflow*; given x, $g_i(x) = b_i - \sum_{(i,j)\in A} x_{ij} + \sum_{(j,i)\in A} x_{ji}$ is the *surplus* of node $i \in N$ w.r.t. x, i.e., the violation of the *flow conservation constraints* in (1). The surplus of a set $S \subseteq N$ w.r.t. x is the sum of the surpluses of the nodes in S, and the total surplus of x is the sum of the positive surpluses of the nodes in G; x is a *flow* if and only if it satisfies all the flow conservation constraints, i.e., its total surplus is zero.

A dual solution y of MCF is also called a vector of *node potentials*; this is the "essential" part of any dual solution, since, given y, the "best possible" z and w are $z = [c - yE]_+$ and $w = [yE - c]_+$, where $[\cdot]_+$ denotes the non-negative part (any other feasible value of z, w corresponds to a larger value of the objective function). Defining the *reduced cost* of arc (i,j) as $\bar{c}_{ij} = c_{ij} - yE^{ij} = c_{ij} - y_i + y_j = z_{ij} - w_{ij}$ (i.e., the residual of the dual constraints, cf. § 2.3), one can develop a different form of "slackening" of the optimality conditions of MCF w.r.t. (3); given a scalar $\mu \geq 0$, a primal-dual pair (x, y) satisfies the μ-complementary slackness conditions (μ-CS for short) if x is a pseudoflow and there holds

$$x_{ij} < u_{ij} \;\Rightarrow\; -\mu \leq \bar{c}_{ij} \qquad 0 < x_{ij} \;\Rightarrow\; \bar{c}_{ij} \leq \mu \qquad (i,j) \in A$$

Primal-dual methods for the solution of MCF problems consider at each iteration a preflow x and a vector of potentials y satisfying μ-CS. This defines a *residual graph* $G_y = (N, A_y)$ which contains only *balanced* arcs, i.e., such that $|\bar{c}_{ij}| \leq \mu$; these are the only arcs where the flow can be changed without violating μ-CS, as all the other arcs are either "active" ($\bar{c}_{ij} > \mu \Rightarrow x_{ij} = 0$) or "inactive" ($\bar{c}_{ij} < -\mu \Rightarrow x_{ij} = u_{ij}$).

Then, the approaches alternate between a primal and a dual phase. In the primal phase, max-flow-type computations are used to send flow on the arcs of G_y from nodes with positive surplus to nodes with negative surplus. If a flow x' is found, that is, the surplus of all nodes is driven to zero, then x' is a μ-optimal solution for MCF; if μ is not small enough already, it is reduced to some value $\mu' < \mu$. This causes some previously balanced arcs to become either inactive or active, i.e., their flow to be set to either the lower or the upper bound, in order to satisfy μ'-CS; in turn, this creates some new "imbalanced" nodes (with nonzero surplus), and the process is iterated.

If, instead, it is not possible to bring the surplus of all nodes to zero, then the max-flow-type computation identifies a new pseudoflow x' and a subset S of nodes whose surplus (w.r.t. x') is larger than the residual capacity (w.r.t. x') of the arcs in the cut $(S, N \setminus S)$. The set S identifies an ascent direction in the dual space, and triggers a dual phase where, by simultaneously increasing the potentials of all nodes $i \in S$ by a proper quantity τ, and leaving all other potentials

unchanged, a solution y' with better value of the dual objective function is attained; if the cut is not saturated, the flow of the balanced arcs in the cut is adjusted (set to either the lower or the upper bound) in order to satisfy μ-CS. A proper choice of the stepsize τ creates new "balanced" arcs, thus allowing to return to the primal phase.

Algorithms that work with a variable μ are said of the *cost-scaling* type, while μ is kept fixed to zero in unscaled techniques; furthermore, different methods (augmenting paths or push-relabel computations) can be used for the primal phase.

3.1. Crossover with the `RelaxIV` solver

The `RelaxIV` solver implements an unscaled primal-dual algorithm [3] which use augmenting paths techniques—using all arcs with zero reduced cost—to push flow from nodes with positive surplus to nodes with negative surplus. The code implements checks for early termination of the primal phase (the max-flow computation), in order to avoid performing flow operations as soon as a dual ascent direction is found, that is, when it becomes clear that no feasible flow exists that satisfies 0-CS with the current vector of potentials y.

For our experiments, we used a `C++` version of the `RelaxIV` solver, developed within the `MCFClass` project [6]. The solver, which is a fairly straightforward port of the original `FORTRAN` code, has shown to be quite effective in several applications (e.g., [7]). A distinctive feature of our `C++` version is the implementation of full reoptimization capabilities, i.e., the ability of re-using the information associated with the obtained optimal solution of an instance to speed-up, often very significantly [9], the solution of a new instance that is obtained by "few" changes of the original one. This is possible due to the fact that the `RelaxIV` solver can start from any pair (x, y) satisfying 0-CS, making it an ideal candidate for implementing a hybrid IP/combinatorial approach.

In fact, the alternative crossover procedure for IP algorithms proposed in [16] is nothing but a primal step of a primal-dual algorithm; the residual graph G_y (corresponding to the current IP dual solution y) is constructed, and a feasible flow is sought for in that graph; if it is found, then it is an optimal solution for the MCF problem, and the IP algorithm can be terminated. Since seeking for a feasible flow only requires a very fast max-flow computation, this can be done at every IP iteration (of the final sequence), simply discarding the result in case no feasible solution is found; this is reported [16] to be more successful than the simplex-based crossover in several instances. Hence, once again the—successful—crossover procedure is nothing but one step of an existing—efficient—combinatorial approach to MCF; in particular, a primal-dual algorithm where no "dual" step is allowed. This clearly suggests to extend the approach by simply leaving the primal-dual method free to perform dual steps, too, if necessary.

Thus, we have modified `RelaxIV` in such a way that it accepts an externally provided primal-dual pair (x, y) for constructing its starting solution. The potentials y can be directly used since, as previously remarked, the best possible corresponding z and w can be easily obtained; actually, this is not even required by the code. By contrast, the primal solution x may have to be modified in order to satisfy 0-CS; however, this is very easy. First, if x is not a pseudoflow it can be made so by setting $x_{ij} = \max \{ 0 , \min \{ u_{ij} , x_{ij} \} \}$. Then, 0-CS can be easily attained by setting $x_{ij} = u_{ij}$ if $\bar{c}_{ij} < 0$, $x_{ij} = 0$ if $\bar{c}_{ij} > 0$, and leaving all other values unchanged. Because the x variables provided by an early stopped IP method, especially a dual one, may not be very significant, it is also possible to use $x = 0$ (eventually adjusted according to the reduced cost) instead. This can be seen as an alternative initialization phase with respect to the one originally provided by the code, based on simple one-node dual moves, that is consequently skipped.

A useful characteristic of `RelaxIV` in this context is that it is virtually a "zero parameter" code; once the initialization is done, in whatever way, the algorithm requires no other special setting. This contrasts with scaling-type approaches, where at the very least the possibly critical decision about the initial value of the scaling parameter μ has to be taken, adding at least another degree of freedom to the system.

4. Computational Results

In this section, we present the results of a large-scale computational test aimed at assessing the effectiveness of the hybrid IP/combinatorial approach.

4.1. Testing environment

For our tests, we used a "generic" by-the-book IP code that we developed. The code is contained in a `C++` class, `IPClass`, and implements all the variants of IP algorithm alluded to in Section 2, comprised all the crash-start rules. The code is generic in that the base class does not provide any means for solving the core systems, demanding this to derived classes where all the information about the structure of the coefficient matrix is hidden; this allows to easily implement specialized IP algorithms for linear programs with special structure, such as MCF. `IPClass` also provides support for approximate solution of the core systems (4), which is crucial if iterative approaches are to be used. In fact, as shown, e.g., in [16, 11], rather "crude" solutions of (4) can be used to provide improvement directions at the initial iterations of the IP method, provided that the accuracy is properly increased as the optimal solution is approached. This requires a nontrivial exchange of information between the IP solver and the PCG algorithm.

For MCF, we developed a PCG-based solver of the core systems (4) that can use several different subgraph-based preconditioners, as described in [8]. This is used within `IPClass` to obtain a (family of) specialized IP algorithm(s) to MCF. Clearly, no crossover is used in this context, since the obtained (primal and) dual solution are passed to the `RelaxIV` solver, as described in the previous paragraph.

We performed our tests on a PC with an Athlon MP 2400+ and 1Gb RAM, running Linux. The code was compiled using the GNU g++ compiler version 3.3, using standard optimization option "-O2".

4.2. Test instances

For our tests, we selected five well-known random generators of MCF problems: `goto` (GridOnTOrus), `gridgen`, `gridgraph`, `mesh`, and `netgen`. We also implemented a `complete` random generator for complete graphs. Apart from the latter and `netgen`, which produces graphs with random topology, all the other generators produce mesh-type graphs with different characteristics. `goto` and `mesh` generate toroidal mesh graphs where each node is connected to all arcs upon a certain distance (decided in the input parameters); however, while `goto` produces instances with a single source and a single destination "far away" from each other, `mesh` generates a circulation problem (with all-0 node deficits) with negative cost arcs. `gridgraph` generates instances similar to `goto` except on a regular 2-dimensional grid plus random arcs. `gridgen` also constructs grid graphs where arcs are directed in alternate directions in each row and column. For `goto`, `gridgen`, `mesh` and `netgen` we generated several families of instances named `gen`$k.d$, where `gen` is the specific generator, $n \approx 2^k$ is the number of nodes and d is the average density.

For `complete` and `gridgraph` we generated instances named `gen`k, where k has the same meaning as above. In each family, 5 different instances were generated by simply changing the seed of the pseudo-random number generator; in the tables below, all the results have to be intended as the average over the five instances of the same family. Source code for the generators is widely available; however, it can also be requested, together with the parameters for reproducing the instances, to the authors.

4.3. Preliminary experiments

It is clear from the previous discussion that there are very many possible options for implementing the extended crossover idea; these comprise at least:

- which variant of IP algorithm (primal, dual, or primal-dual) is used;
- whether or not an affine variant is used;
- for the primal-dual method, if Mehrotra's predictor-corrector strategy is used, and if so, how many "multiple centrality corrections" are performed;
- how many iterations of the IP method are performed before switching to the combinatorial approach;
- which of the applicable crash-start rules are used, and how their parameters (τ, σ, μ, ...) are chosen;
- whether or not the pseudoflow x is used to warm-start the combinatorial approach together with the node potentials y.

Even restricting some of the above choices by heuristic decision (e.g., always setting $\tau = 0.5$ in the primal crash formulae Px and testing only two different values for σ in the dual slack crash formulae Sx) led us to more than 7000 variants. Furthermore, several other possibilities could have been tested. The selection of the preconditioner for solving (4) has been done according to the guidelines set forth in [8], but those guidelines have been developed for the *complete* solution of MCF via an IP approach rather than for performing only a few IP iterations. However, further increasing the degrees of freedom of the system would have lead to an unmanageable number of alternatives. Thus, we performed our computational tests focusing our attention only on the above parameters.

The computational experiments were performed in two phases. In the preliminary phase, a significant subset of the instances were tested with *all* possible variants of extended crossover, in order to find out the most promising alternatives. The detailed results of this phase cannot clearly be reported here in full; collecting and analyzing the data was a very long and intensive process, whose results can be briefly described as follows:

- the solution of the IP approach often provides useful information to the combinatorial algorithm (leading to a decrease of its running time) very early, possibly as early as the crash start phase;
- the dual and primal-dual method, in their non-affine variants, are typically much better than the primal method at providing warm starts; for the predictor-corrector, the cost of more than one centrality correction is not worth the effect on the quality of the obtained warm start;

- due to the high iteration cost of the IP approach, only very few IP iterations can be performed in order to obtain an overall competitive hybrid approach (although exceptions exist, as discussed below);

- the impact of the crash formulae is indeed significant; this should be expected in view of the above point;

- using the pseudoflow x provides little benefit, but it does no harm, either.

Perhaps the most important result from the preliminary phase, however, was the extreme difficulty in selecting a small set of "best" parameters. Many variants are dominated, often significantly, by other variants for some families of instances, while dominating them on other families. Finally, we resorted to selecting four variants which showed the best compromise between performances on all families; these differ for using the dual or the primal-dual algorithm and using the combination P1 + S1 or P5 + S3 for the primal and dual slack crash formulae, while all performing one iteration of the IP approach, using formula D1 for the dual crash variables[1], and passing the pseudoflow x as a part of the warm-start. We remark that, by our experiments, performing one IP iteration is, at large, better than performing none, i.e., only relying on the crash formulae; thus, the IP machinery indeed helps in obtaining good starting solutions.

The experiments on the full set of instances were finally performed only on these four variants. We stress that this choice consistently underestimates the best performances attainable on some of the families; in particular, for `goto` instances considerably better results can be obtained by allowing the IP algorithm to run for a much larger number of iterations, as described in the next paragraph.

4.4. Final experiments

The results of the final set of experiments is reported in Table 2 for all families except those obtained with the `goto` generator, and in Table 3 for the latters. In all tables, column RIV reports the solution time in seconds of the standard `RelaxIV` solver. In Table 2, columns D-11 and D-53 report the solution time for the hybrid approach using the dual IP approach and crash formulae P1 + S1 and P5 + S3, respectively, and analogously for columns PD-11 and PD-53 for the primal-dual method. In Table 3, column PD reports the solution time for the hybrid approach using the primal-dual IP approach and P5 + S3, since this variant is found to be the most efficient on this family of instances. All other parameters are set as described in the previous paragraph, except for the number of IP iterations for the results of Table 3, that is set to 8 instead of 1 as in all other cases, since this resulted in a very significant improvement of the efficiency of the hybrid approach.

The results in the tables clearly show that the hybrid algorithm may provide significantly better results than `RelaxIV`, with improvements ranging from a few percentage points (e.g., `mesh`15.40) to 50% (`grid`16.8, `net`14.8). On `goto` instances, where IP approaches typically perform well [16, 15] (despite the "difficulty" of the core systems [8]) while `RelaxIV` is slower [9], improvements range from a factor of 3 to an impressive factor of 35, even for relatively small instances. However, the results also show that the improvements are not uniform; on several families, the hybrid approach is at best on par or marginally slower than `RelaxIV`, although it

[1] Rule D2 provides significantly better potentials but requires the solution of one extra system (4) with all arc weights equal, that can be very costly to solve in some cases as shown by the `goto` instances in Table 1.

Table 2: Comparison of the hybrid approach versus `RelaxIV`

problem	D-11	D-53	PD-11	PD-53	RIV
grid8.32	0.024	0.020	0.022	0.024	0.020
grid8.64	0.064	0.070	0.070	0.072	0.070
grid12.8	1.036	0.870	0.874	0.870	0.933
grid12.64	5.644	5.470	5.542	5.594	5.460
grid12.256	16.41	16.46	16.75	16.51	15.09
grid16.8	32,72	31,75	31,67	32.06	73.78
complete2	0.010	0.006	0.008	0.010	0.010
complete4	0.712	0.758	1.022	1.024	0.650
ggraph10	0.430	0.428	0.450	0.448	0.290
ggraph12	5.224	5.234	5.226	5.240	4.214
ggraph14	8.180	8.174	8.188	8.188	8.433
net8.32	0.018	0.020	0.020	0.014	0.010
net12.8	0.718	0.728	0.726	0.720	0.790
net12.64	3.580	3.590	3.596	3.604	2.540
net12.256	11.35	11,38	11.58	11.64	9.56
net14.8	4.724	4.774	4.812	4.740	7.370
net14.64	14.27	14.38	14.60	14.62	13.82
mesh14.8	3.288	3.286	3.322	3.340	2.940
mesh14.40	15.24	14.37	14.45	14.46	11.88
mesh14.64	18,00	17.41	17.56	17.59	15.98
mesh15.40	83.41	83.61	83.69	83.82	88.67
mesh15.64	85.78	88.60	89.12	89.21	96.41
mesh17.10	158.91	158.77	159.10	159.25	130.82

should be remarked again that significantly better results could be obtained by allowing instance-based setting of the parameters. Also, there does not seem to be any obvious relationship between the characteristics of the instances (graph topology, size, density, ...) and the relative efficiency of the hybrid approach versus `RelaxIV`. Hence, further research is required in order to make hybrid IP/combinatorial approaches routinely usable for solution of MCF problems. Yet, the consistently positive results that can be obtained on some instances show that the approach deserves further development.

4.5. Conclusion

Combining Interior-Point and "combinatorial" approaches for the solution of Linear Programs is a very well-established technique; without crossover, the usefulness of IP algorithms would be severely limited in several contexts. For general LPs, the only combinatorial approach that can be paired with IP algorithms is the simplex method; however, for structured LPs like MCF, other specialized combinatorial companions can be used instead. By combining the strengths of the different algorithms—the fast global convergence of IP methods with the extreme speed of "local" optimization moves of combinatorial approaches—better results could be obtained.

Our results show that this is indeed true in some relevant cases. However, they also show that a delicate tuning of the several possible options (IP algorithm employed, crash start formula and

Table 3: Hybrid approach versus `RelaxIV`: `goto` instances

problem	PD	RIV
goto8.8	0.05	0.15
goto8.16	0.11	0.81
goto8.32	0.22	2.35
goto12.8	12.82	65.32
goto12.64	129.88	4718.60
goto12.256	5782.43	25203.44

parameters, number of iterations, ...) is required, which makes hybrid approaches currently unsuitable for general-purpose, "fire-and-forget" MCF solvers. The need for developing accurate and dependable guidelines about when switching between the IP algorithm and its "combinatorial companion" brings about some issues at the frontier between IP algorithms for MCF [8] and the study of warm-starts for combinatorial algorithms to MCF [9], namely: are there metrics that allow to measure how good, say, a primal-dual pair (x, y) is as a warm start to some combinatorial MCF approach? and, is there some variant of IP algorithm that is particularly well-suited for rapidly producing such good solutions? We believe that further investigation on these issues could bring results of interest in their own right, as well as allowing to implement effective general-purpose hybrid IP/combinatorial MCF solvers.

Acknowledgments

This research has been partly funded by Line 2.4 of CNR/MIUR Project "Simulazione e Ottimizzazione per Reti: Software e Applicazioni (SORSA) – SP7". We are also grateful to Ares Salvadori for his help in performing the computational tests and analyzing the results.

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: theory, algorithms and applications*. Prentice Hall, New Jersey, 1993.

[2] E.D. Andersen and Y. Ye. Combining interior-point and pivoting algorithms for linear programming. *Management Science*, 42(12):1719–1731, 1996.

[3] D.P. Bertsekas and P. Tseng. Relax-IV: A faster version of the Relax code for solving minimum cost flow problems. LIDS-P-2276. November 1994. Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

[4] D.P. Bertsekas and P. Tseng. RELAX: A Computer Code for Minimum Cost Network Flow Problems. *Annals of Operations Research*, 13:127–190, 1988.

[5] J. Castro. A specialized interior-point algorithm for multicommodity network flows. *SIAM Journal on Optimization*, 10:852–877, 2000.

[6] A. Frangioni. `http://www.di.unipi.it/optimize/Software/ MCF.html`.

[7] A. Frangioni and G. Gallo. A bundle type dual-ascent approach to linear multicommodity min cost flow problems. *INFORMS Journal on Computing*, 11(4):370–393, 1999.

[8] A. Frangioni and C. Gentile. New Preconditioners for KKT Systems of Network Flow Problems. *SIAM Journal on Optimization*, 14(3):894–913, 2004.

[9] A. Frangioni and A. Manca. A Computational Study of Cost Reoptimization for Min Cost Flow Problems. *INFORMS Journal on Computing*, 18(1):61–70, 2006.

[10] A.V. Goldberg. An efficient implementation of a scaling minimum- cost flow algorithm. *J. of Algorithms*, 22:1–29, 1997.

[11] J.J. Jùdice, J.M. Patrício, L.F. Portugal, M.G.C. Resende, and G. Veiga. A Study of Preconditioners for Network Interior Point Methods. *Computational Optimization and Applications*, 24(1):5–35, 2003.

[12] J.L. Kennington and R.V. Helgason. *Algorithms for network programming.* John Wiley and Sons, New York, NY, 1980.

[13] N. Megiddo. On finding primal- and dual-optimal bases. *ORSA Journal on Computing*, 3:63–65, 1991.

[14] S. Mehrotra. On the implementation of a (primal-dual) interior point method. *SIAM Journal on Optimization*, 2:575–601, 1992.

[15] L.F. Portugal, M.G.C. Resende, G. Veiga, and J.J. Jùdice. A Truncated Primal-infeasible Dual-feasible Network Interior Point Method. *Networks*, 35:91–108, 2000.

[16] M.G.C. Resende and G. Veiga. An efficient implementation of a network interior point method. In D.S. Johnson and C.C. McGeoch, editors, *Network flows and matching: First DIMACS implementation challenge*, volume 12, pages 299–348. American Mathematical Society, Providence, Rhode Island, 1993.

[17] M.G.C. Resende and G. Veiga. An Implementation of the dual affine scaling algorithm for minumum cost flow on bipartite uncapacited networks. *SIAM Journal on Optimization*, 3/3:516–537, 1993.

[18] T. Roos, T. Terlaky, and J.-Ph. Vial. *Theory and Algorithms for Linear Optimization: An Interior Point Approach.* John Wiley and Sons, Chichester, 1997.

[19] S.J. Wright. *Primal-Dual Interior-Point Methods.* SIAM, Philadelphia, PA, 1997.