Università di Pisa
Dipartimento di Informatica

Technical Report: TR-04-23

Experiments with "extended crossover" approaches for network flow problems

Antonio Frangioni Claudio Gentile

December 22, 2004

Address: via F. Buonarroti 2, 56127 Pisa, Italy. Tel: +39 050 2212700 Fax: +39 050 2212726

Experiments with “extended crossover” approaches for network flow problems

Antonio Frangioni *　　Claudio Gentile †

December 22, 2004

Abstract

Interior Point algorithms for Min-Cost Flow problems have been shown to be competitive with more established “combinatorial” approaches, at least on some problem classes and for very large instances. A fundamental contribution to the efficiency of these approaches is provided by the specialized crossover routines that can be implemented for this problem; they allow early termination of the IP approach, sparing it with the “nasty” iterations where the core linear systems become very difficult to solve by iterative algorithms. Because crossover procedures are nothing but one step of a combinatorial approach to MCF, we propose to extend this concept to that of *extended crossover*: use the initial part of an Interior Point algorithm to MCF to warm-start a “combinatorial” algorithm. We report our experiments along this line using, as “combinatorial companion”, a primal-dual algorithm for MCF that has a natural way for exploiting the information provided by the 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 two original approaches in isolation, at least on some “difficult” instances.

Keywords: *Min Cost Flow problems, Interior Point algorithms, Relaxation Method, Crossover.*

1 Introduction

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

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

*Università di Pisa, Dipartimento di Informatica, Largo B. Pontecorvo 1, 56127 Pisa – Italy, e-mail: `frangio@di.unipi.it`

†Istituto di Analisi dei Sistemi ed Informatica “Antonio Ruberti” del C.N.R., Viale Manzoni 30, 00185 Rome – Italy, e-mail: `gentile@iasi.cnr.it`

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. A detailed description of the IP algorithms and their underlying theory can be found in the extensive literature on the subject and in many recent linear programming textbooks, e.g., [14, 15]. In these methods, at each iteration, one or more linear systems of the form

$$(E\Theta E^T)\Delta y = d \tag{2}$$

have to be solved, where Θ and d are respectively an $m \times m$ diagonal matrix ($m = |A|$) with positive entries and a vector of $\mathbb{R}^n$ ($n = |N|$) which depend on the current iteration and on the specific IP variant chosen. General-purpose LP solvers typically use direct methods, such as the Cholesky factorization; however, for structured LPs – like MCF – iterative approaches, such as Preconditioned Conjugate Gradient (PCG) methods, have shown to be competitive [5, 13, 12]. Indeed, appropriate preconditioners that "closely approximate" $M = E\Theta E^T$ can be found by exploiting the structure of the problem [13, 12, 6].

For MCF, approaches carefully implementing the above ideas have been shown [13] to be competitive with "combinatorial" approaches, like primal-dual [3] cost-scaling [8] algorithms or simplex methods [10], albeit on some cases and for very large instances. Closely examining the results reveals that a fundamental contribution to the overall efficiency of the approach is provided by the *specialized crossover* routines that can be implemented for MCF algorithms. These algorithms attempt to construct an optimal (basic) solution out of the information provided by the Interior Point approach; if successful, the whole algorithm is stopped. Indeed, this always happens relatively early in the optimization process, sparing the IP approach with several of the final iterations; this is particularly important because:

- the linear systems (2) become more and more ill-conditioned, and therefore difficult to solve by iterative approaches, as the algorithm nears an optimal solution;
- the linear systems (2) 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 algorithm nears an optimal solution.

Thus, early termination is crucial for IP approaches to MCF. Because crossover procedures are basically simplex-like approaches [11, 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". This suggested us to experiment with the following generalization of the above idea, that we call *extended crossover*: combine an Interior Point method and a combinatorial 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 each of the original approaches in isolation. Clearly, this also depends on the "combinatorial companion" employed; since simplex methods have already been used in standard crossover procedures, we focus our attention on the other class of efficient combinatorial algorithms for MCF, namely, primal-dual approaches. We show by extensive experiments that 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 combined approach can be competitive with the two original approaches, especially on some classes of "difficult" instances.

The structure of the paper is the following: in Section 2 (several variants of) Interior Point algorithms are 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), rewritten as

$$\min \{ \, cx \;:\; Ex = b, \; x + s = u, \; x, s \geq 0 \, \} \tag{3}$$

where $x \in \mathbb{R}^m$ and $s \in \mathbb{R}^m$ are respectively the primal variables and the slacks of the box constraints, and its dual

$$\max \{ \, yb - wu \;:\; yE + z - w = c, \; z, w \geq 0 \, \} \;, \tag{4}$$

where $y \in \mathbb{R}^n$, $z \in \mathbb{R}^m$ and $w \in \mathbb{R}^m$ are respectively the dual variables of the structural constraints $Ex = b$, the dual slacks and the dual variables of the box constraints $x \leq u$.

2.1 Variants of Interior Point algorithms

A number of different IP algorithms can be constructed, which all start from the "slackened" version of the *KKT optimality conditions* of the above pair of dual problems:

$$Ex = b \tag{5}$$

$$yE + z - w = c \tag{6}$$

$$x + s = u \tag{7}$$

$$XZe = \mu e \tag{8}$$

$$SWe = \mu e \tag{9}$$

$$(x, s, z, w) \geq 0$$

Here, $\mu \geq 0$ is a parameter, e is the vector of 1's of proper dimension, and each uppercase letter corresponds to the diagonal matrix having as diagonal elements the entries of the corresponding lowercase vector. For $\mu = 0$, the above system characterizes all the optimal solutions of (3) and (4). For $\mu > 0$, the unique solution of the system (5) – (9) lies on the *central path*, a continuous trajectory which, as μ tends to 0, converges to an optimal solution of (3) and (4) (more precisely, it converges to the analytic centers of the primal and dual optimal faces).

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 system (5) – (9). These algorithms can be divided into *primal*, *dual* or *primal-dual* according to how exactly the Newton step is done.

Primal methods In the primal method, a primal (not necessarily feasible) point (x, s) is kept as the center of the Newton iteration, and dual variables (y, z, w) corresponding to the direction $(\Delta x, \Delta s)$ are derived from proper linearizations of the equations of the system. More precisely, from (8) one obtains

$$z = \mu[X + \Delta X]^{-1}e,$$

which is then linearized with a first-order Taylor expansion as

$$z = \mu[X^{-1}e - X^{-2}\Delta x].$$

A similar formula for w can be obtained by linearizing (9), and, by using these relations in the remaining equations, one obtains explicit formulae for y, Δx and Δs.

Dual methods In the dual method, the dual point (y, z, w) is kept as the center of the Newton iteration, and primal variables (x, s) corresponding to the direction $(\Delta y, \Delta z, \Delta w)$ are derived from proper linearizations of the equations of the system. From the first-order Taylor expansion of (8) one obtains

$$x = \mu[Z^{-1}e - Z^{-2}\Delta z],$$

from which explicit formulae for s and $(\Delta y, \Delta z, \Delta w)$ can be derived.

Primal-dual methods In the primal-dual method, both a primal point (x, s) and a dual point (y, z, w) are kept as the center of the Newton iteration, and a primal-dual Newton direction $(\Delta x, \Delta s, \Delta y, \Delta z, \Delta w)$ is sought for. This is obtained by rewriting (8) – (9) as

$$(X + \Delta X)(Z + \Delta Z)e = \mu e$$

$$(S + \Delta S)(W + \Delta W)e = \mu e$$

and then linearizing the above nonlinear system by dropping the second-order terms $\Delta X \Delta Z$ and $\Delta S \Delta W$.

Once that a Newton direction – in the primal and/or dual space – has been obtained, an appropriate stepsize is selected which moves the current point "closer" to the central path for the current value of μ, then μ is reduced by multiplying it for a factor $0 < \rho < 1$ and the whole process is repeated. With appropriate choices of the stepsize, a sequence of primal and/or dual points converging to an optimal solution is constructed.

Further variants Several variants of the above methods have been defined. For instance, in the *predictor-corrector* variant of the primal-dual method an iterative approach is taken for refining the solution of the nonlinear system by iteratively substituting the obtained values of ΔX, ΔZ, ΔS and ΔW in the neglected quadratic terms and re-solving the modified linear system; this comes at the cost of multiple solutions of the system for each iteration, but usually improves on the number of overall IP iterations. Also, *affine* variants of all the above IP algorithms have been developed, where the formulae for the Newton direction are taken as the limit for $\mu \rightarrow 0$ of the formulae in the path-following case; this simplifies the formulae, making them faster to compute and completely eliminating μ from them, thereby avoiding the problem of tuning its decrease. On the other hand, affine variants tend to be less numerically stable than barrier variants.

2.2 Solving the core sistems

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 (2). The solution of (2) typically represents by far the main computational burden of the IP algorithms; for a structured LP like MCF, developing a specialized approach for the solution of (2) allows to construct more efficient, specialized IP approaches to the problem. Note that, since the form of the core system is independent from the specific variant of IP algorithm used, the same specialized solver for (2) can be used to implement all the variants of IP algorithms.

Preconditioned Conjugated Gradient methods using subgraph-based preconditioners have been shown to provide good performances in the MCF case. The basic idea is to 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; these preconditioners can be further improved by adding them a multiple of the diagonal of the "disregarded" part of the matrix. Possible classes of triangulated graphs are trees [13, 12, 9] or "Brother-connected trees" [6].

These preconditioners usually tend to become more and more efficient as the IP algorithm proceeds, since the interior point 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, (2) 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 (2) 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 (2), with some sequences of IP iterations, especially – but not only – towards the end of the IP algorithm, showing relatively "hard" systems (2) to solve, at least by the PCG approach. 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 (2) by an iterative approach have to be expected, especially towards the end of the IP algorithm; since that step is by far the computational bottleneck of the IP approach, avoiding the "nasty" iterations is crucial for the overall efficiency of current IP algorithms for MCF.

	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

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

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, s) 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, nor even with respect to "simple" constraints like $yE+z-w=c$ or $x+s=u$, only provided that $(x, s, z, w) > 0$. Surprisingly, 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 so robust that their overall performances are only marginally impacted by the choice; alternatively, it may be one of those "little secrets" that practical developers of IP algorithms keep to themselves in order to get an edge on competition. Whatever the reason, it is difficult to get accurate guidance on this subject even in the "standard" context of the solution of a LP by means of a IP algorithm; even more so in our case, where the choice of the crash solution may clearly have a much broader impact 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 and slacks (x, s), the dual solution y and the dual slacks (z, w) can be chosen almost independently, although there are cross-dependencies (e.g., typically (z, w) are chosen after y in order to have $yE + z - w = c$) and there are also cases of combined formulae. We now report on the variants that we implemented and tested.

Dual crash For the dual variables y, the following three variants have been proposed:

D1) $y = 0$;

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

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

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 (z, w).

Primal crash For the primal solution and slacks (x, s), the following five variants have been proposed:

P1) $x_i = \tau u_i$, where $\tau > 0$ if fixed;

P2) $x_i = \tau u_i$ if $\bar{c}_i \geq 0$, $x_i = (1 - \tau)u_i$ otherwise;

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

P4) $x = (1/2)(u + (1/\tau)(E^T\lambda - c))$, where $\lambda = (EE^T)^{-1}((\tau/2)b + E(c - u/\tau))$;

P5) $x_i = u_i/2 + \tau/\bar{c}_i - \kappa_i$ if $\bar{c}_i > 0$, $x_i = u_i/2 + \tau/\bar{c}_i + \kappa_i$ if $\bar{c}_i < 0$, and $x_i = u_i/2$ otherwise, where $\kappa_i = \sqrt{(u_i/2)^2 + (\tau/\bar{c}_i)^2}$.

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 obtain $x + s = u$; this is not necessary but clearly results in a different crash solution, thus the number of alternatives is nearly doubled.

Dual slack crash For the dual slacks (z, w), the following three variants have been proposed:

S1) if $\bar{c}_i \geq 0$ then $w_i = \sigma$ and $z_i = \bar{c}_i + \sigma$, otherwise $z_i = \sigma$ and $w_i = \sigma - \bar{c}_i$, where σ is fixed;

S2) $z_i = \sigma/x_i$, $w_i = \sigma/s_i$;

S3) $z_i = \sigma/u_i + \bar{c}/2 + \sqrt{(\sigma/u_i)^2 + (\bar{c}_i/2)^2}$, $w_i = \sigma z_i/(u_i z_i - \sigma)$.

Clearly, only some of the above formulae produce a dual feasible solution (such that $yE + z - w = c$).

Combined crash formulae Finally, a combined formula has been proposed that constructs x, z and w simultaneously so that it belongs to the "central path" for a given value of μ, except that constraints $Ex = b$ are ignored: this boils down to

$$\begin{array}{l} z_i = (2\mu + \bar{c}u_i + \sqrt{4\mu^2 + (\bar{c}_i u_i)^2})/(2u_i), \\ w_i = (2\mu - \bar{c}u_i + \sqrt{4\mu^2 + (\bar{c}_i u_i)^2})/(2u_i), \\ x_i = \mu/z_i. \end{array} \tag{10}$$

Clearly, choosing the right combination of the above formulae and their parameters (τ in Px, σ in Sx, ...) is by no means trivial.

2.4 Crossover

Early termination of Interior Point methods can be obtained by means of "crossover" procedures. The idea is that 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 the general IP case, 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), since the IP algorithm would naturally produce an infinite solution sequence;
- being able to combine IP approaches with traditional simplex methods in order to exploit the superior reoptimization capabilities of the latter, especially in some crucial applications like Branch & Bound algorithms.

That is, for the general IP case the crossover procedure is not mainly 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.

For the reasons mentioned above, for MCF the situation is quite different. Indeed, the specialized crossover algorithms for MCF [13] are a crucial element for the overall efficiency of the IP approach. Different crossover procedures can be constructed. The easiest one is perhaps to test whether the maximum-weight spanning tree of G with arc weights Θ – typically, already computed by the preconditioners – provide, possibly with minimal variations, a primal and dual feasible basis for MCF. This is analogous to the standard simplex-like crossover procedures for the general IP algorithm [11, 2] except that, due to its negligible cost, the procedure can be repeated at every iteration (of the final sequence) discarding the results if feasibility is not achieved. Alternatively, the current dual solution of the IP algorithm can be used to construct an "admissible subgraph" of G such that every feasible flow on that graph, if any, is an optimal solution to the problem [13]; since seeking for a feasible flow only requires a very fast max-flow computation, this can be done at every iteration (of the final sequence). As it will become clearer in the next section, this latter approach can be seen as one "primal" step of a generic primal-dual algorithm for MCF where no "dual" step is allowed; once again, the – successful – crossover procedures are nothing but one step of an existing – efficient – combinatorial approach to MCF.

All this suggested us to try to extend the concept of crossover to that of *extended crossover*: use the initial part of an (efficient) Interior Point algorithm to MCF to warm-start an (efficient) combinatorial algorithm. Clearly, this requires one way in which the combinatorial approach can exploit the information provided by the "continuous" IP method. Primal-dual algorithms for MCF have a very natural way for doing this, as described in the next section.

3 Primal-dual algorithms for MCF

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

3.1 A generic primal-dual algorithm

Any vector m-vector x such that $0 \leq x_{ij} \leq u_{ij}$ for each $(i, j) \in A$ is a *pseudoflow*; given x, the surplus $g_i(x)$ of node $i \in N$ w.r.t. x is

$$g_i(x) = b_i - \sum_{(i,j)\in A} x_{ij} + \sum_{(j,i)\in A} x_{ji} \ ,$$

i.e., the violation of the *flow conservation constraints* in (1). The surplus of a subset S of nodes 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. Given a scalar $\varepsilon \geq 0$, a primal-dual pair (x, y) satisfies the ε-complementary slackness conditions (ε-CS for short) if x is a pseudoflow and there holds

$$x_{ij} < u_{ij} \quad \Rightarrow \quad -\varepsilon \leq c_{ij} - y_i + y_j \quad \forall (i,j) \in A \ ,$$

$$0 < x_{ij} \quad \Rightarrow \quad c_{ij} - y_i + y_j \leq \varepsilon \quad \forall (i,j) \in A$$

where $c^y_{ij} = c_{ij} - y_i + y_j$ is the *reduced cost* of the arc (i, j). It is well-known that 0-CS are necessary and sufficient conditions for optimality of a primal-dual pair (x, y) where x is feasible; analogously, a primal-dual pair satisfying ε-CS is said ε-optimal.

Primal-dual methods for the solution of MCF problems consider at each iteration a preflow x and a vector of potentials y satisfying $\varepsilon - CS$. Max-flow-type computations are used to send flow on the arcs of the reduced graph (i.e., such that $|c^y_{ij}| \leq \varepsilon$) in order to reduce the total surplus; if a flow is found, ε is reduced and the process iterated. Otherwise, a cut of the graph is identified such that its surplus is larger than its residual capacity; this gives an ascent direction in the dual space that is used to modify y. Algorithms that work with a variable ε are said of the cost-scaling type, while ε is kept fixed to zero in unscaled techniques; furthermore, different methods (augmenting paths or push-relabel computations) can be used for the primal phase.

3.2 The `RelaxIV` solver

The `RelaxIV` solver [4] implements an unscaled primal-dual algorithm 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.

The `RelaxIV` solver can start from any pair (x, y) satisfying 0-CS; this makes it very convenient for implementing the extended crossover idea. In fact, any IP method, stopped after any fixed number of IP iterations, provide a primal-dual pair (x, y) that can be easily used to initialize `RelaxIV`. The dual values y can be directly used as the current vector of potentials, since, as previously shown, 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 also very easy. First, if x

is not bound-feasible ($0 \leq x_{ij} \leq u_{ij} \ \ \forall (i,j) \in A$) it can be made so by setting $x_{ij} = \max\{0, \min\{u_{ij}, x_{ij}\}\}$; then, one can set $x_{ij} = u_{ij}$ if $c^y_{ij} < 0$ and $x_{ij} = 0$ if $c^y_{ij} > 0$, 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.

Thus, we have modified `RelaxIV` in such a way that it accepts externally provided values for the initial vector of potentials and (possibly) pseudoflow; the latter is then internally adjusted to satisfy 0-CS. This can be seen as an alternative initialization phase with respect to the one originally provided by the code, 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 ε 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 extended crossover idea.

4.1 Testing environment

For our tests, we used a "generic" by-the-book Interior Point code that we developed. The code is contained in a `C++` class, `IPClass`, and implements all the variants of IP algorithm described 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 (2), which is crucial if iterative approaches are to be used. In fact, as shown, e.g., in [13], rather "crude" solutions of (2) 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 (2) that can use several different subgraph-based preconditioners, as described in [6]. This is used within `IPClass` to obtain a (family of) specialized IP algorithm(s) to MCF. Clearly, no crossover is used in this context.

As "combinatorial companion" of the IP approach, we used the `C++` version of the `RelaxIV` solver available as a part of the `MCFClass` project

`http://www.di.unipi.it/di/groups/optimize/Software/MCF.html`

The code has been modified, as described in the previous paragraph, in order to accept a (primal and) dual solution as an externally-provided warm-start, thereby skipping all the built-in initializations.

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, 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, ...) 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 DSx) led us to more than 7000 variants. Furthermore, several other possibilities could have been tested. Other combinatorial approaches may have parameters of their own (e.g., the ε parameter for scaling-type primal-dual approaches) to tune according to the starting point. The selection of the preconditioner for solving (2) has been done according to the guidelines set forth in [6], 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 primal-dual, 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 extended crossover 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 dual slack crash formula S1 (together with the primal crash formula

P0) or (10), while all performing one iteration of the IP approach, using formula D0 for the dual crash variables[1], and passing the pseudoflow x as a part of the warm-start. 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 considerably 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, columns RIV and IP report the solution time in seconds of the `RelaxIV` and the IP solver in isolation (for the latter, the final required precision is `1e-8` relative). In Table 2, columns D-1 and D-C report the solution time for the extended crossover variant using the dual IP approach and dual slack crash formula S1 or (10), respectively, and analogously for columns PD-1 and PD-C for the primal-dual method. In Table 3, column PD reports the solution time for the extended crossover variant using the primal-dual IP approach and (10), 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 extended crossover approach.

The results in the tables clearly show that the IP algorithm without any form of crossover is not competitive with the combinatorial approach. The extended crossover 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, which are notoriously "difficult" for both IP [6] and combinatorial approaches [7], improvements range from a factor of 3 to a factor of 35. However, the results also show that the improvements are not uniform; on several families, the extended crossover is at best on par or marginally slower than `RelaxIV`, at least if, as in our experiments, only a small set of the many possible variants of extended crossover is allowed. 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 extended crossover versus the combinatorial approach. Hence, further research is required in order to make extended crossover 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.

[1] Formula D1 provides significantly better potentials but requires the solution of one extra system (2) with all arc weights equal, that can be very costly to solve in some cases as shown by the `goto` instances in Table 1.

problem	D-1	D-C	PD-1	PD-C	RIV	IP
grid8.32	0.024	0.020	0.022	0.024	0.020	0.431
grid8.64	0.064	0.070	0.070	0.072	0.070	0.900
grid12.8	1.036	0.870	0.874	0.870	0.933	5,212
grid12.64	5.644	5.470	5.542	5.594	5.460	35.36
grid12.256	16.41	16.46	16.75	16.51	15.09	209.73
grid16.8	32,72	31,75	31,67	32.06	73.78	1058.94
complete2	0.010	0.006	0.008	0.010	0.010	0.46
complete4	0.712	0.758	1.022	1.024	0.650	103.05
ggraph10	0.430	0.428	0.450	0.448	0.290	9.51
ggraph12	5.224	5.234	5.226	5.240	4.214	83.35
ggraph14	8.180	8.174	8.188	8.188	8.433	204.00
net8.32	0.018	0.020	0.020	0.014	0.010	0.31
net12.8	0.718	0.728	0.726	0.720	0.790	3,24
net12.64	3.580	3.590	3.596	3.604	2.540	28,44
net12.256	11.35	11,38	11.58	11.64	9.56	540.44
net14.8	4.724	4.774	4.812	4.740	7.370	207.85
net14.64	14.27	14.38	14.60	14.62	13.82	272.48
mesh14.8	3.288	3.286	3.322	3.340	2.940	36.18
mesh14.40	15.24	14.37	14.45	14.46	11.88	518.03
mesh14.64	18,00	17.41	17.56	17.59	15.98	864.59
mesh15.40	83.41	83.61	83.69	83.82	88.67	1972.77
mesh15.64	85.78	88.60	89.12	89.21	96.41	4098.99
mesh17.10	158.91	158.77	159.10	159.25	130.82	5532.27

Table 2: Comparison of extended crossover versus IP and RelaxIV in isolation

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 Interior Point 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. The extended crossover approach brings further this idea by trying to combine the strengths of the different algorithms: the fast global convergence of IP methods with the extreme speed of "local" optimization moves of combinatorial approaches.

Our results show that extended crossover approaches are quite successful in some relevant cases. However, they require a delicate tuning of the several possible options (IP algorithm employed, crash start formula and parameters, number of iterations, ...), which makes them currently unsuitable for general-purpose, "fire-and-forget" MCF solvers. The need for developing accurate and dependable guidelines about when and how the extended crossover can be successfully used brings about some issues at the frontier between Interior-Point algorithms

problem	PD	RIV	IP
goto8.8	0.05	0.15	0.14
goto8.16	0.11	0.81	0.45
goto8.32	0.22	2.35	1.15
goto12.8	12.82	65.32	67.46
goto12.64	129.88	4718.60	2458.87
goto12.256	5782.43	25203.44	31289.30

Table 3: Extended crossover versus IP and `RelaxIV` in isolation: `goto` instances

for MCF [6] and the study of warm-starts for combinatorial algorithms to MCF [7], 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 MCF solvers based on the extended crossover approach.

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

[7] Frangioni, A. and Manca, A. A Computational Study of Cost Reoptimization for Min Cost Flow Problems. *INFORMS Journal on Computing*, to appear, 2004.

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

[9] 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.

[10] A. Löbel. Solving large-scale real-world minimum-cost flow problems by a network simplex method. Technical report, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Germany, 1996.

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

[12] 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.

[13] 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.

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

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