Discrete Newton Algorithms for Budgeted Network Problems

S. Thomas McCormick[1], Gianpaolo Oriolo[2], and Britta Peis[3]

[1] University of British Columbia, Sauder School of Business, Canada
tom.mccormick@sauder.ubc.ca
[2] DICII Università di Roma Tor Vergata
oriolo@disp.uniroma2.it
[3] RWTH University of Aachen
britta.uni.mails@gmail.com

Abstract. Discrete Newton Algorithm has a good track record for solving various network-type problems. We extend this record towards some problems where budget may be used to reduce the arc capacities: (1) Network Interdiction; (2) Reverse Network Interdiction; (3) Max Robust Flow; and (4) Parametric Scheduling. In cases (1), (2) we get more combinatorial approximation algorithms and faster running times than previously known, in (3) we show hardness and provide an approximation algorithm, in (4) we get new results for what was an open problem.

1 Introduction

Classic max flow has a directed graph $G = (V, E)$ with source s and sink t in V, and non-negative capacities c_e on E. For simplicity, we introduce a return arc $e^* = (t, s)$ of infinite capacity. Then the classical problem to send as much flow as possible from s to t in the capacitated network (G, c) can be formulated as the linear program to $\max x_{ts}$ s.t. (1) $\sum_{(uv)\in E} x_{uv} - \sum_{(vu)\in E} x_{vu} = 0$ for all $v \in V$ ("conservation"), and (2) $0 \le x_e \le c_e$ for each $e \in E$ ("boundedness"). The *value* of a flow is $\mathrm{val}(x) \equiv x_{ts}$. Arc subset S is an s–t *cut* if $S = \delta^+(U) := \{(u, v) \in E \mid u \in U, v \notin U\}$ for some $U \subseteq V \setminus \{t\}$ with $s \in U$; the *capacity* of cut S is $\mathrm{cap}(S) \equiv c(S) \equiv \sum_{e\in S} c_e$. The famous Max Flow/Min Cut Theorem of Ford and Fulkerson [8] says that $\max_{x \text{ a flow}} \mathrm{val}(x) = \min_{S \text{ a cut}} \mathrm{cap}(S)$.

There are many papers that consider extensions of this with parametric capacities, see e.g. [9, 11–14, 17–20]. Now the capacity of $e \in E$ is a non-negative function $c_e(\lambda)$ of the scalar parameter λ. Define $\mathrm{cap}^*(\lambda)$ to be the joint max flow/min cut value for each value of λ. In all of our applications, $c_e(\lambda)$ is a piecewise linear concave function of λ with at most two pieces, and so we assume this property; we sometimes call the curve $\mathrm{cap}^*(\lambda)$ the λ-*profile*. This implies that $\mathrm{cap}^*(\lambda)$ is also piecewise linear concave. There are various problems that one might want to solve in such parametric networks: (a) Find the entire curve $\mathrm{cap}^*(\lambda)$ [9, 11]; (b) Find a minimum value of λ such that there is a flow saturating all arcs at s [13]; (c) Find a cut S minimizing a ratio $a(S)/b(S)$ for vectors

$a, b \in \mathbb{R}^E$ via the trick of Lagrange-ifying the denominator to $\min_S a(S) - \lambda b(S)$ [14, 17–20]. A common thread to these results is that various versions of the Discrete Newton Algorithm (see Section 2) solve all these problems. Note that (a) only makes sense when we know that the λ-profile has a polynomial size, which typically happens only with GGT-type (see [9]) networks. In cases where the λ-profile can have exponential size, Discrete Newton often can still solve problems in polynomial time.

We add a new problem in parametric networks and show that it is also solvable via Discrete Newton: We are given a target B, and we want to find a breakpoint of the $\text{cap}^*(\lambda)$ curve such that the slopes of the two adjacent segments of the curve bracket B. We call this version the *Slope Problem*, and we call the version of Discrete Newton that we use solve the Slope Problem *Newton-B*. We show below that it enjoys the same (strongly) polynomial iteration bounds as the other forms of Discrete Newton. We also show how to extend Newton-B to problems with (a fixed number of) multiple parameters; this is the first extension of such methods to multiple parameters that we are aware of.

We show that the Slope Problem is particularly suitable for network problems where some budget w.r.t. *removal cost* r_e on arc e may be used to reduce (or increase) the arc capacities (possibly fractionally), and we show that Newton-B is quite powerful for solving those problems. Namely we get combinatorial algorithms that produce (pseudo-)approximate solutions to the following well-known hard budgeted network problems:

1. The *Network Inhibition Problem*, also called *Network Interdiction Problem* (*NI*, see [2, 4, 7, 16, 22]), which asks for the best way to use a budget B to delete the arc capacities of a given network so as to minimize the value of a maximum flow that can be sent through the resulting network. Knowing an optimal, or near-optimal, inhibition strategy is useful, for example, to inhibit the flow of dangerous material through a network, or to detect vulnerabilities in a network. Notice that NI allows fractional deletion of arc capacities and also flow to be re-routed after capacity deletion. Applying Newton-B in this case requires proving a conjugate duality result, which allows us to get a combinatorial way to implement the pseudo-approximation framework of [4] (Section 3).
2. A *Reverse Network Inhibition Problem* (*RNI*), which asks for the best way to use a budget B to build up capacities for a network so as to maximize the value of the maximum flow that can be sent through the resulting network. RNI has a perhaps surprising connection to the *Quickest Flow Problem* in flows over time, see [5]. Here we can use bounds on (ordinary) Newton from [20] to get better run times than [5] (Section 4).
3. A *Maximum (non-reroutable) Robust Flow Problem*, which asks for a flow *on paths* that guarantees the maximum amount of flow to reach the sink no matter how an interdictor uses a budget of B to delete arc capacities (possibly fractionally, as for NI). This problem is related to NI, but asking for a flow on paths models the setting where the chosen flow cannot be rerouted after the interdictor destroys some of the arc capacities. A version of MRF

where the interdictor is to remove Γ arcs, for some given Γ (so fractional deletion is forbidden) and $r_e = 1$ on each arc $e$was studied in [2, 3, 7]. Here we use insights from Newton-B applied to NI and some side remarks to generalize and combinatorialize the pseudo-approximation algorithm proposed in [3] (Section 5).

4. A two-parameter version of a scheduling problem of Chen [6] (see also [13]). Suppose that the parameters are λ and μ. Then arcs with tail s have capacities which decrease in λ and μ, and we want to minimize $\lambda + \mu$ such that there exists a flow saturating all arcs with tail s. Here we can characterize the facets of the "efficient frontier" between feasible and infeasible values of (λ, μ), and use Newton-B to compute an optimal solution; it was not previously known how to solve this problem (Section 6).

Finally, we extend our results to more general classes of problems, and to multiple parameters (Section 7).

2 The Newton-B Algorithm

Our aim is to find a flow-based algorithm for solving the Slope Problem for given slope B. We assume that we are given, or can compute, initial values λ_L and λ_R such that the breakpoint whose slopes bracket B is at $\lambda^B \in [\lambda_L, \lambda_R]$. Define $D = \lambda_R - \lambda_L$ as a measure of the size of the data. We also assume that the given parametric capacities are "discrete" enough that if we can refine λ_L and λ_R into an interval of width $1/mD$ that still contains λ^B, then we can use, e.g., continued fractions to round to the exact breakpoint λ^B.

A simple way to get a combinatorial algorithm would be to use binary search. Define $G(\lambda)$ to be the max flow network with capacities $c_e(\lambda)$. When we generate a new value $\bar{\lambda}$ to try, we would compute a max flow in $G(\bar{\lambda})$ of value $\text{cap}^*(\bar{\lambda})$, yielding the point $(\bar{\lambda}, \text{cap}^*(\bar{\lambda}))$ of the λ-profile. Define $\text{sl}^-(\bar{\lambda})$ ($\text{sl}^+(\bar{\lambda})$) to be the slope of the segment of the λ-profile to the left (right) of $\bar{\lambda}$. Since the λ-profile is concave, we have $\text{sl}^-(\bar{\lambda}) \geq \text{sl}^+(\bar{\lambda})$, with equality when $\bar{\lambda}$ is not a breakpoint.

Recall that there may be multiple min cuts in $G(\bar{\lambda})$. For $e \in E$ define its *forward slope* $c_e^+(\bar{\lambda})$ (*backward slope* $c_e^-(\bar{\lambda})$) at $\bar{\lambda}$ as its local slope to the right (left) of $\bar{\lambda}$. If S is a min cut in $G(\bar{\lambda})$, define its forward slope $\text{sl}^+(S)$ (backward slope $\text{sl}^-(S)$) to be $c^+(S)$ ($c^-(S)$). We claim that the right-hand slope of the piecewise linear concave curve of $\text{val}(\lambda)$ at $\bar{\lambda}$ is $\text{sl}^+(\bar{\lambda}) = \min\{\text{sl}^+(S) \mid S \text{ is a min cut in } G(\bar{\lambda})\}$, and the left-hand slope of the piecewise linear concave curve of $\text{val}(\lambda)$ at $\bar{\lambda}$ is $\text{sl}^-(\bar{\lambda}) = \max\{\text{sl}^-(S) \mid S \text{ is a min cut in } G(\bar{\lambda})\}$

To prove this, suppose that S is a min cut in $G(\bar{\lambda})$ and T is not a min cut in $G(\bar{\lambda})$. For sufficiently small $\epsilon > 0$, $\text{cap}(S, \bar{\lambda} + \epsilon) = \text{cap}(S, \bar{\lambda}) + \epsilon \cdot \text{sl}^+(S) < \text{cap}(T, \bar{\lambda} + \epsilon)$. Therefore, for sufficiently small $\epsilon > 0$, the minimum cut value at $\bar{\lambda} + \epsilon$ is taken by a min cut S at $\bar{\lambda}$ with a minimum value of $\text{sl}^+(S)$. The proof for $\text{sl}^-(S)$ is similar. The Appendix shows how to compute $\text{sl}^+(S)$ and $\text{sl}^-(S)$ (and their associated cuts) using at most four max flow computations.

If $B \in [\text{sl}^+(\bar{\lambda}), \text{sl}^-(\bar{\lambda})]$ then $\bar{\lambda}$ is the value of λ we are looking for. Otherwise, if $B < \text{sl}^+(\bar{\lambda})$ ($B > \text{sl}^-(\bar{\lambda})$) then we can replace the right (left) endpoint of

our interval by $\bar{\lambda}$ and iterate. Each iteration takes $O(\text{MF})$ time, where MF is a max flow computation, and the number of iterations is $O(\log nD)$, and so binary search is a weakly polynomial combinatorial algorithm.

In order to beat binary search we adapt an analysis of Discrete Newton Algorithm originally developed by McCormick and Ervolina [14], then extended by Radzik [17–20]. The idea is similar to binary search: both algorithms keep an interval $[\lambda_L, \lambda_R]$ known to contain λ^B (since $\text{sl}^+(\lambda_L) \geq B \geq \text{sl}^-(\lambda_R)$). At the next step binary search chooses $\bar{\lambda} = (\lambda_L + \lambda_R)/2$, but Discrete Newton chooses $\bar{\lambda}$ as the intersection point of the line with slope $\text{sl}^+(\lambda_L)$ at λ_L, and the line with slope $\text{sl}^-(\lambda_R)$ at λ_R.

In [19, Section 3.5] Radzik observes that his analyses of Discrete Newton also apply to the case where we are instead seeking to maximize the concave function, i.e., to find a breakpoint whose local slopes bracket 0. It is easy to see that the same reasoning applies to the slightly more general case here where we are looking for a breakpoint whose local slopes bracket B. Since each $c_e(\lambda)$ is piecewise linear with at most two pieces, we can replace e with at most two parallel copies, each carrying one of the pieces of $c_e(\lambda)$, and so we can assume that all capacities are affine functions of λ. Define C to be the largest constant term in any affine function, and R to be the largest coefficient of λ in any affine function (so that $D = O(CR)$).

Theorem 1 ([19, 20]). *The number of iterations of Discrete Newton is (a) never worse than binary search; (b) $O(\frac{\log(mCR)}{1+\log\log(mCR)-\log\log(MR)})$, which is sometimes faster than binary search; (c) $O(m^2 \log^2 n)$, which is strongly polynomial.*

Thus Newton-B is combinatorial, is never worse than binary search, is sometimes faster than binary search, and enjoys a strongly polynomial bound that binary search can never achieve.

3 Network Interdiction

It is well-known that an s–t cut in G of minimum capacity corresponds to an optimal integral solution of the dual of the max flow problem, namely, $\min \sum_{e \in E} c_e y_e$ s.t. $d_u - d_v + y_{uv} \geq 0$ for all $(u, v) \in E$, and $d_t - d_s + y_{ts} \geq 1$, d free and $y \geq 0$. Note that we can require d to be integral without loss of optimality. There exist various fast algorithms that compute a max flow and a min cut in (G, c).

Network interdiction as MIP. It is easy to see that an optimal strategy of the inhibitor will destroy capacities on a single s–t cut S (that depends on B). Thus, if we introduce variables z_e on the arcs to indicate what fraction of arc e we want to destroy, as well as the budget constraint $\sum_{e \in E} r_e z_e \leq B$, we can formulate NI as the following mixed integer linear program (introduced by Wood [22]) which we call NI-MIP: $\min \sum_{e \in E} c_e y_e$ s.t. $d_u - d_v + y_{uv} + z_{uv} \geq 0$ for all $(u, v) \in E$, and $d_t - d_s + y_{ts} + z_{ts} \geq 1$, $\sum_{e \in E} r_e z_e \leq B$, d free and integer, and $y, z \geq 0$. Given budget B, we call the value of NI-MIP the *residual capacity (w.r.t. B)*.

Complexity results and a bicriteria approximation. Phillips[16] showed that NI is strongly $\mathcal{NP}$-complete on general graphs and weakly $\mathcal{NP}$-complete on planar graphs, and gave an FPTAS for planar graphs (see also [22]). Burch et al. [4] give the following bicriteria pseudo-approximation algorithm for NI: they solve the linear relaxation of NI-MIP[1], decompose the optimal fractional solution into a convex combination of two cuts, and show that at least one of these cuts needs a budget of $\hat{B}$ and has residual capacity $\hat{C}$ such that $\frac{\hat{B}}{B} + \epsilon\frac{\hat{C}}{C^*} \leq 1 + \epsilon$, where $\epsilon > 0$ is a given error parameter, and C^* is the minimum residual capacity for any attack of the network with given budget B.

Our contribution. Since NI is a "network-type" problem, it is natural to wonder if there is a more direct, combinatorial way to compute the approximation than the LP-based approach of [4]. We show that the answer is "yes" via the convex duality result in the next section. This leads to a strongly polynomial algorithm that directly finds the two cuts that give the bicriteria pseudo-approximation.

3.1 Conjugate Duality

The dual LPs for max flow/min cut are:

$$\begin{array}{rll} \max x_{ts} & & \\ \text{s.t. } \sum_k x_{ki} - \sum_j x_{ij} = 0 & \forall\ i & \\ c_e \geq x_e \geq 0 & \forall\ e; & \end{array} \qquad \begin{array}{rll} \min \sum_e c_e y_e & & \\ \text{s.t. } d_u - d_v + y_{uv} \geq 0 & \forall\ (u,v) \in E & \\ d_t - d_s + y_{ts} \geq 1 & & \\ y_e \geq 0 & \forall\ e. & \end{array}$$

The NI-MIP inserts new variables z_e into the dual constraint along with the budget constraint $\sum_E r_e z_e \leq B$, and demands that d is integer. If we relax this integrality constraint, associate primal variable λ with the budget constraint, and then take the primal, we get the dual LPs:

$$\begin{array}{rll} \max x_{ts} - B\lambda & & \\ \text{s.t. } \sum_k x_{ki} - \sum_j x_{ij} = 0 & \forall\ i & \\ c_e \geq x_e \geq 0 & \forall\ e & \\ x_e - r_e\lambda \leq 0 & \forall\ e; & \end{array} \qquad \begin{array}{rll} \min \sum_e c_e y_e & & \\ \text{s.t. } d_u - d_v + y_{uv} + z_{uv} \geq 0 & \forall\ (u,v) \in E & \\ d_t - d_s + y_{ts} \geq 1 & & \\ \sum_E r_e z_e \leq B & & \\ y_e \geq 0 & \forall\ e. & \end{array}$$

Let us get some geometrical insight to this dual LP. Fix some s–t cut S. Then S induces dual variables d and y feasible to the dual. We can then raise the z_e for arcs in S to reduce the objective value. Clearly the best we can do for S is to choose its arcs greedily in decreasing order of $\rho_e \equiv \frac{c_e}{r_e}$, arc e's *bang for the buck*. Thus the curve we get for the residual capacity of S as we vary B is the piecewise linear convex curve in Figure 1.

We can now superimpose these curves for all s–t cuts to get the picture in Figure 2. The minimum of all these curves gives the optimum NI value for

[1] the NI-MIP-formulation in [4] is slightly different from ours, but their approach applies as well to our NI-MIP

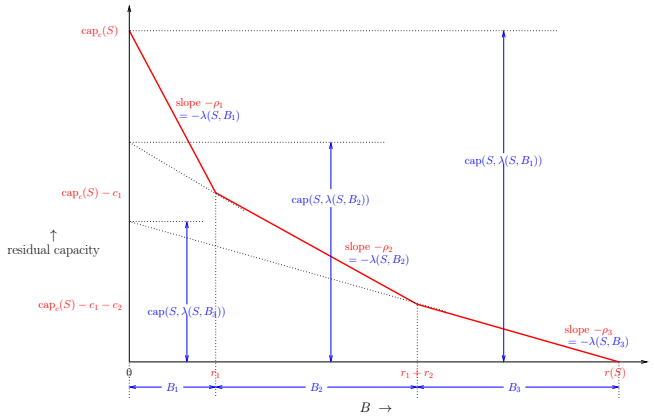

Fig. 1. The heavy red interdiction curve for fixed cut S.

each value of B. Unfortunately, the minimum of convex curves is not in general convex. However, by relaxing d to allow fractional values allows us to take convex combinations of cuts, and so we would get the piecewise linear convex hull of the minimum indicated by the dotted line in Figure 2. We call this curve *the B-profile*, and we denote the value of the B-profile at B (i.e., the optimal value of the above dual LP) by resid$^*(B)$. Every breakpoint of the B-profile is associated with a cut.

On the primal side, the new constraint $x_e - \lambda r_e \leq 0$ is essentially a second upper bound $x_e \leq \lambda r_e$. Putting the two upper bounds together gives $x_e \leq \min(c_e, \lambda r_e) \equiv c_e(\lambda)$, and so we get a network with upper bounds parametrized by λ. Denote the max flow value in the network with bounds $c_e(\lambda)$ by cap$^*(\lambda)$. Then cap$^*(\lambda)$ is a piecewise linear concave curve which is the λ-profile for this parametric flow problem. Note that the optimal value of the above primal is cap$^*(\lambda) - \lambda B$.

We can get some geometric intuition for the λ-profile as well: Fix an s–t cut S. Then S's capacity as a function of λ is a piecewise linear concave curve. Since the max flow value is the minimum over all cuts of such curves, the curve cap$^*(\lambda)$ is again piecewise linear concave as in Figure 3. Every linear piece of the λ-profile is associated with a cut.

Recall from convex duality theory (see Rockafellar [21]) that there is a conjugate (Fenchel) duality between convex functions. In particular, if f is a piecewise linear convex function, its conjugate dual $f^\bullet$ is also piecewise linear convex, and the breakpoints of f correspond 1-1 with the segments of $f^\bullet$, and the segments of f correspond 1-1 with the breakpoints of $f^\bullet$. Convex conjugate duality then implies:

Theorem 2. *The λ-profile is the reverse of the negative of the B-profile, i.e.,* cap$^*(\lambda) = -[\text{resid}^*(-\lambda)]^\bullet$. □

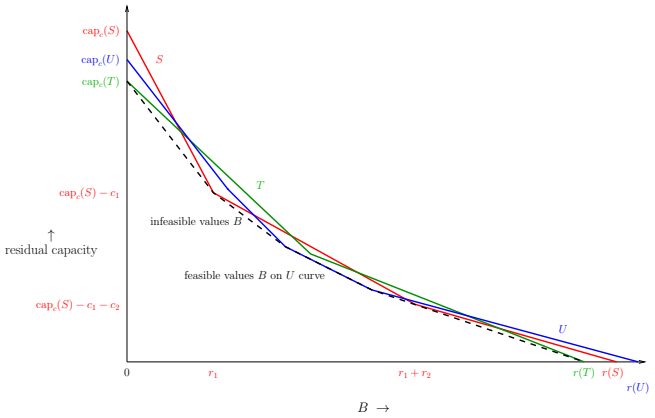

Fig. 2. The dashed black line is the convex hull of the lower envelope, and is the B-profile. Note that the indicated segment of the B-profile is determined by a segment of U's interdiction curve, and so that interval of B values is feasible, whereas other segments that are convex hulls of interdiction curves of different cuts do not contain feasible values of B.

Theorem 2 implies that the breakpoints of the B-profile (corresponding to cuts) are the segments of the λ-profile, and vice-versa. This allows us to translate the Burch et al. pseudo-approximation algorithm framework into our terms: For the given value of B, find the linear segment of the B-profile containing B. This segment runs between two breakpoints with associated cuts S_1 and S_2. Either S_1 or S_2 will have the claimed pseudo-approximation guarantee.

Due to the conjugate duality of Theorem 2, we can equivalently implement the Burch et al. algorithm like this: For the given value of B, find the breakpoint of the λ-profile whose adjacent slopes bracket B. This breakpoint is determined as the intersection of two segments of the λ-profile associated with two cuts S_1 and S_2. These are the same cuts that we need to get the pseudo-approximation. This is an instance of the Slope Problem, and so we can solve it using Newton-B. Then Theorem 1 shows that we have a (strongly) polynomial, combinatorial algorithm for computing a pseudo-approximate solution to NI.

4 Reverse Network Interdiction (RNI)

In RNI a builder is allowed to use a budget of B in order to "buy" arc capacity. In this model, the cost for buying the full capacity c_e on e is r_e. It can be seen that this problem is equivalent to the Quickest Flow problem in flows over time, see e.g. Burkard et al. [5].

A natural algorithm for solving this problem is the *Successive Shortest Paths (SSP)* Algorithm. It would first select the cheapest s–t path and put as much flow as possible on it. Then it would find the next cheapest path and saturate it

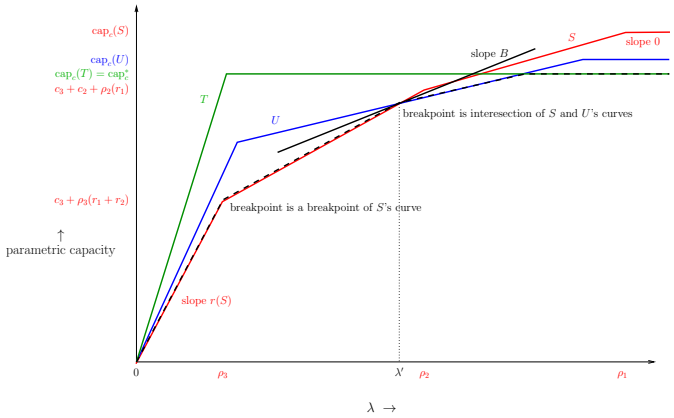

Fig. 3. The λ-profile for cuts S, T, and U is the dashed black line. As indicated, breakpoints of the λ-profile can come either from a breakpoint of a single cut's curve or from the intersection of two cuts' curves. Our desired breakpoint is one where a line of slope B is tangent.

(which might entail reversing some flows on the first path), etc. To solve RNI, we would stop once we have used up the budget B. Notice that SSP traces out a sort of piecewise linear B-profile curve, with increasing B on the x-axis, and the cumulative flow on the y-axis.

A drawback of SSP is that it is known that in general its curve could have an exponential number of segments, and so SSP could take exponential time to reach B. However, notice that we can access points of this curve "from the other direction": We could test some value F of cumulative flow as a required flow from s to t, and then find a min-cost flow w.r.t. costs r_e that achieves this flow value. The min-cost flow optimal value $r(F)$ is how much we would have to pay to buy capacities supporting this flow. If $r(F) = B$ then we are done. If $r(F) > B$ then we need to reduce F, and if $r(F) < B$ then we need to increase F. This suggests a natural binary search algorithm.

As before, we can do better by replacing binary search with Discrete Newton. In this case we are not trying to find a point whose slopes bracket B, but rather just trying to find what the value of the curve is at B under the constraint that we can't easily evaluate the curve for any B value, but we can evaluate the curve at any F value via min-cost flow. The bounds in Theorem 1 certainly apply to ordinary Discrete Newton as well as Newton-B. The bounds (b) and (c) from Theorem 1 are faster than what [5] find.

5 Max Robust Flow (MRF)

In this section, we deal with the *Maximum (non-reroutable) Robust Flow problem* (MRF), which asks for an s–t flow in (G, c) in path-formulation that guar-

antees the maximum amount of flow to reach the sink t no matter how an interdictor uses a budget of B to destroy arc capacities. Recall that a flow in path-formulation is a non-negative assignment y to the set of all s–t paths $\mathcal{P}$ in G that satisfies the capacity constraints on each arc. Let $Y = \{(y_P)_{P\in\mathcal{P}} \mid \sum_{P\in\mathcal{P}:e\in P} y_P \leq c_e \forall e \in E\}$ denote the set of all feasible s–t flows in G[2].

Like in the previous sections, we introduce variables $\mu_e \in [0,1]$ on the arcs to indicate which fraction of e is deleted by the interdictor. Thus, the set of feasible strategies of an interdictor with budget B is $X = \{\mu_e \in [0,1]^E \mid r^T\mu \leq B\}$. Given a flow $y = (y_P)_{P\in\mathcal{P}} \in Y$, the maximal amount the interdictor can delete using a budget of B is $z(y) = \max_{\mu\in X} \sum_{P\in\mathcal{P}} y_P \max_{e\in P} \mu_e$.

We therefore define $Rval(y)$, the *robust value* of y, as $Rval(y) := val(y) - z(y)$, i.e., as the amount of flow that is anyhow guaranteed to reach the sink. Thus, MRF can be written as $\max_{y\in Y} Rval(y)$, i.e., as the problem to choose a feasible flow of maximum robust value.

Note that we allow fractional deletion and different removal costs r_e on the arcs. To the best of our knowledge, so far MRF has not been investigated in either of these settings. Namely, it has been assumed that $r_e = 1$ on each arc e and that $\mu \in X$ has to be integral. In this case: [2] shows that the problem can be solved in polynomial time when $B = 1$; [7] shows that the problem is NP hard when $B = 2$; [3] proposes a pseudo-approximation algorithm for the problem, for any given integer B. Note also that a different model, that is based on arc-based formulation has been introduced by Bertsimas et al. [3].

Our first remark is that, for MRF with $B = 2$ and $r_e = 1$ on each e, imposing integrality to the interdictor is not harmful. Hence (proofs in the Appendix):

Lemma 1. *MRF is NP-hard. Even if $B = 2$ and $r_e = 1$ on every arc $e \in E$.*

We now show that the approach that is taken in [3] to get a pseudo-approximation algorithm can be generalized to the case where we allow fractional deletion and different removal costs. The first remark is that, given a flow $y \in Y$, the value $z(y)$, i.e., the maximal amount of flow the interdictor can delete using a budget of B, is upper bounded by $f(y) = \max_{\mu\in X} \sum_{P\in\mathcal{P}} y_P \min\{1, \sum_{e\in P} \mu_e\}$. However, we can also build on $f(y)$ to provide a lower bound for $z(y)$. Using similar arguments as in [1] we can prove:

[2] In MRF, we ask for a flow in path-formulation (instead of arc-formulation) for the following reason: Given a flow $(x_e)_{e\in E}$ in arc-based formulation, the surviving x' after interdiction is not necessarily a flow anymore, since conservation might be violated. So, what is the value of surviving flow that reaches the sink? In the *adaptable robust flow* model introduced by Bertsimas et al. [3], the maximum surviving flow is defined to be the maximum value of flow in the network with arc-capacities x'_e. Their model allows the flow to be adapted/re-routed after interdiction, at least within the bounds given by the prior chosen flow. It is natural to wonder: if this re-routing is allowed, why not allow re-routing in general? General re-routing would lead back to NI, where it does not matter at all which flow was chosen in the first round. In contrast, if we ask for a flow in path-formulation, i.e., with a precise routing scheme describing which amount of flow should be send along which route, the question for a maximum robust flow/routing scheme makes sense.

Lemma 2. *$\rho f(y) \leq z(y)$, for $\rho = 1 - (1 - \frac{1}{k})^k$ and $k = \max_{P \in \mathcal{P}} |P|$.*

The following theorem shows that any path decomposition of an optimal solution to our Slope Problem (w.r.t. B), returns a pseudo-approximation to MRF. Its proof mainly exploits the fact that $f(y)$ can be computed by a linear program. Note that Theorem 3 is indeed a generalization of Theorem 5 in [3] to the case of different deletion costs and fractional interdiction. We believe that our proof (in the Appendix) fixes a few issues in the proof in [3], in particular for statement (ii).

Theorem 3. *Let $\bar{x}, \bar{\lambda}$ be an optimal solution to the slope problem*

$$(SP) \quad \begin{array}{ll} \max_{x,\lambda \geq 0} & val(x) - B\lambda \\ s.t. & conservation \\ & x_e \leq \min\{u_e, r_e\lambda\} \ \forall e \in E. \end{array}$$

and let $y = \{y_P, P \in \mathcal{P}\}$ be a path-decomposition of $\bar{x}$. Also, let y^* be a flow with maximum robust value, and $g^*(y) := val(y) - f(y)$. Then the followings hold:

(i) $Rval(y) \geq val(\bar{x}) - B\bar{\lambda}$;
(ii) $g(y^*) \leq val(\bar{x}) - B\bar{\lambda}$;
(iii) $Rval(y^*) - Rval(y) \leq \frac{1-\rho}{\rho}(val(y^*) - Rval(y^*))$;

Note [3]: As a consequence of (iii), if y^* is such that $\beta val(y^*) \leq Rval(y^*)$ for some $\beta \in (0, 1]$, then y is a $(1 - \frac{1-\rho}{\rho}\frac{1-\beta}{\beta})$-approximate solution to MRF. In particular, if $\beta \geq \frac{1}{2}$, then y is an α-approximate solution for some $\alpha \geq 0.58197$.

6 Chen-Type Scheduling Problems

The capacity on arc (s, j) is $\max(0, p_j - a_j\lambda - b_j\mu)$, and we want to minimize $\lambda + \mu$ s.t. there being a flow that saturates all arcs with tail s. Let $C \subset V$ be a node subset that does *not* contain s nor t such that there is at least one $j \in V$ with $(s, j) \in E$ and $j \in C$, so that $\delta^-(C)$ is non-empty. Let D be a non-empty subset of the arcs in $\delta^-(C)$ with tail s. Then it is easy to see that any feasible (λ, μ) must satisfy the *cut-subset* constraint $\lambda a(D) + \mu b(D) \geq p(D) - c(\delta^+(C))$ (see the Appendix for further details).

Thus the boundary between feasible and infeasible points (λ, μ) is a piecewise linear convex curve. The optimal value of (λ, μ) is the breakpoint on this curve whose local slopes bracket -1. Thus we can again use Newton-B to solve this problem in (strongly) polynomial time as per Theorem 1.

7 Extensions

)

Multiple parameters: It is natural to consider extensions of NI problems where we have multiple resources that can destroy arc capacity, each with its

own budget. The question would then be how to optimally spend each budget so as to remove as much flow as possible. Since the case with one budget is already NP Hard, the multiple budget case is hard also. But can we extend the Burch et al. pseudo-approximation framework?

Burch et al. [4] already remark that this is possible by doing the natural thing: now each resource would have its own vector of z_e variables in the relaxed dual LP, and its own knapsack budget constraint in the dual LP. When we primalize, we would then get multiple primal parameters, one for each resource. A conjugate duality result like Theorem 2 would still hold. The subproblem that we would then want to solve looks like this: Given a vector of budgets $(B_1, B_2, \ldots, B_K)$ for the k resources, find a breakpoint of the parametric surface such that the local slopes at the breakpoint bracket B_i along coordinate i, $i = 1, \ldots, k$.

We can solve this via a recursive application of Discrete Newton. For simplicity, consider the case with $k = 2$. For a fixed value of λ_2, we can use Discrete Newton to find a point $\lambda_1^{B_1}$ such that the local slope in the first coordinate direction bracket B_1. Consider the local slopes at this point in the second coordinate direction. If they happen to bracket B_2, then we are done. If they are bigger than B_2 then we know that we have to move λ_2 to the right; if they are smaller than B_2 then we know that we have to move λ_2 to the left. This gives us enough information that we can now implement an outer Discrete Newton to search for an optimal value of λ_2. The resulting running time will be $O((\#\text{ Newton iterations})^2 \cdot \text{MF})$. This easily generalizes to k parameters to get:

Theorem 4. *We can solve a problem with k parameters in* $O((\#\text{ Newton iterations})^k \cdot \text{MF})$ *time.*

This remains a (strongly) polynomial algorithm for any fixed number of parameters, but is not polynomial when k is part of the input. This is frustrating as the LP method of [4] works for any k. It is an open question whether there is a combinatorial method for a non-fixed number of parameters.

Other interdiction problems: The Burch et al. framework, and our Discrete Newton method for computing it, are both quite general. Suppose that we take any "primal" problem with capacities, so that the primal LP includes constraints $x_e \leq c_e$. This will lead to dual variables y_e that need to "cover" the primal costs at min cost. We can get an interdiction version by adding in new variables z_e that can substitute for y_e without penalizing the dual objective, but with the z_e variables constrained by a knapsack-style budget constraint. This would then lead back to a version of the primal with parametrized capacities $\min(c_e, \lambda r_e)$.

Since Theorem 2 comes from LP duality plus conjugate duality, it remains true for this more general problem. Therefore we can again use Discrete Newton to find the optimal breakpoint λ^B and the local structures associated with its adjacent segments. We can then plug these structures into the framework of [4], and get the same guarantee as before. We can also get the extension for a fixed number of parameters. As one example of this, consider the Chen-type scheduling problem with more than two parameters. Our methods here apply to such problems as well as long as the number of parameters is fixed thereby solving an open problem posed in [6, 13].

Acknowledgements We thank Martin Skutella for pointing out [4, 5] and helpful discussions, Sebastian Stiller for pointing out [3] and helpful suggestions, Venus Lo for suggesting RNI, Maurice Queyranne and Stefania Garasto for helpful discussions.

References

1. A.A. Ageev, and M.I. Sviridenko (1999) Approximation Algorithms for Maximum Coverage and Max Cut with Given Sizes of Parts. *LNCS: IPCO 7-th* **1610**, 17–30.
2. Y. P. Aneja, K. P. K. Nair, and R. Chandrasekaran (2001). Maximizing residual flow under arc destruction. *Networks*, **34**, 194–198.
3. D. Bertsimas, E. Nasrabadi, and S. Stiller (2013). Robust Network Flows. *Operations Research*, **34 (1)**, 1218–1242.
4. C. Burch, R. Carr, M. Marathe, C. Phillips, and E. Sundberg (2002). A Decomposition-Based Pseudoapproximation Algorithm for Network Flow Inhibition. Chapter in *Network Interdiction and Stochastic Integer Programming*, 51-68.
5. R.E. Burkard, K.Dlaska, and B.Klinz (1993). The Quickest Flow Problem. *Operations Research*, **34:1**, 31–58.
6. Y.L. Chen (1994). Scheduling Jobs to Minimize Total Cost. *European J. of Operational Research*, **74**, 111–119.
7. D. Du, R. Chandrasekaran (2007). The maximum residual flow problem: Np-hardness with two-arc destruction. *Networks*, **50**, 181–182.
8. L. R. Ford, and Fulkerson, D. R. (1962). Flows in Networks. Princeton University Press, Princeton, NJ.
9. G. Gallo, M.D. Grigoriadis, and R.E. Tarjan (1989). A Fast Parametric Maximum-Flow Algorithm and Applications. *SIAM J. Comput.* **18**, 30–55.
10. M. Goemans, and D. Williamson (1994). New 3/4-approximation algorithms for MAX SAT. *SIAM J. Discrete Math.* **7**, 656–666.
11. F. Granot, S.T. McCormick, M.N. Queyranne, and F. Tardella (2012). Monotone parametric min cut revisited: structures and algorithms. *Math. Pro.*, **135**, 337–367.
12. D. Gusfield and C. Martel (1992). A fast algorithm for the generalized parametric minimum cut problem and applications. *Algorithmica*, **7**, 499–519.
13. S.T. McCormick (2000). Fast Algorithms for Parametric Scheduling come from Extensions to Parametric Maximum Flow. *Operations Research*, **47**, 744–756.
14. S.T. McCormick, and T.R. Ervolina (1994). Computing Maximum Mean Cuts. *Discrete Applied Math*, **52**, 53–70.
15. J.-C. Picard, and M. N. Queyranne (1980). On the Structure of All Minimum Cuts in a Network and Applications. *Math. Prog. Study*, **13**, 8–16.
16. C.A. Phillips (1993). The network inhibition problem. In *Proceedings of the 25th Annual ACM Symp on the Theory of Computing*, 288–293.
17. T. Radzik (1992). Minimizing Capacity Violations in a Transshipment Network. *Proceedings of the Third Annual ACM-SIAM SODA*, Orlando 1992, 185–194
18. T. Radzik (1992). Newton's Method for Fractional Combinatorial Optimization. In *Proceedings of 33rd FOCS 1992*, 659–669.
19. T. Radzik (1993). Parametric Flows, Weighted Means of Cuts, and Fractional Combinatorial Optimization. Chapter in "Complexity in Numerical Optimization" (ed. P. Pardalos), World Scientific, 351–386.
20. T. Radzik (1998). Fractional Combinatorial Optimization. Chapter in "Handbook of Combinatorial Optimization" Kluwer Academic, 429–478.
21. R. T. Rockafellar (1984). *Network Flows and Monotropic Optimization*, Wiley Interscience, New York, NY.
22. R.K. Wood (1993). Deterministic network interdiction. *Mathematical and Computer Modelling* **17** 1–18.

Appendix

How to compute local slopes

We now show how to compute $\mathrm{sl}^+(\bar{\lambda})$ and $\mathrm{sl}^-(\bar{\lambda})$ using at most four max flow computations. We specialize to the case arising in NI, but the method is general. Compute a max flow in $G(\bar{\lambda})$, and then compute $\hat{G}(\bar{\lambda})$, the contracted network on the strongly connected components of the residual graph from the Picard-Queyranne representation of all min cuts in $G(\bar{\lambda})$ [15].

To compute $\mathrm{sl}^+(\bar{\lambda})$, for $P \to Q$ an arc of $\hat{G}(\bar{\lambda})$, set $u_{PQ} = r(\{i \to j \in A \mid i \in P,\ j \in Q, \text{ and } \bar{\lambda} r_{ij} < c_{ij}\})$, i.e., the r-value of the set of forward lively arcs that $P \to Q$ represents. For each arc $P \to Q$ of $\hat{G}(\bar{\lambda})$, also construct a reverse arc $Q \to P$ with $u_{QP} = \infty$.

Now we claim that a min cut in $\hat{G}(\bar{\lambda})$ with these capacities will identify an ideal of the poset that $\hat{G}(\bar{\lambda})$ represents that corresponds to a min cut S of $G(\bar{\lambda})$ with $\mathrm{sl}^+(S) = \mathrm{sl}^+(\bar{\lambda})$. First, note that the infinite-capacity reverse arcs ensure that every finite-capacity cut (and hence a min cut) in $\hat{G}(\bar{\lambda})$ is indeed an ideal (it is easy to construct examples where the lack of such reverse arcs would cause the min cut in $\hat{G}(\bar{\lambda})$ to not be an ideal). It is easy to see that if min cut S in $G(\bar{\lambda})$ corresponds to ideal I of $\hat{G}(\bar{\lambda})$ that $\mathrm{sl}^+(S) = \mathrm{cap}(I)$, and so a min cut I in $\hat{G}(\bar{\lambda})$ does indeed pick out a min cut S attaining the value $\mathrm{sl}^+(\bar{\lambda})$.

Similarly, to compute $\mathrm{sl}^-(\bar{\lambda})$, for $P \to Q$ an arc of $\hat{G}(\bar{\lambda})$, set $l_{PQ} = r(\{i \to j \in A \mid i \in P,\ j \in Q, \text{ and } \bar{\lambda} r_{ij} \le c_{ij}\})$, i.e., the r-value of the set of backward lively arcs that $P \to Q$ represents. In this case the upper bounds u_{PQ} are already implicitly ∞, so we do not need to add the reverse arcs.

Notice that $\hat{G}(\bar{\lambda})$ is acyclic, and so we can use known methods to compute a max cut in it. We claim that a max cut in $\hat{G}(\bar{\lambda})$ with these capacities will identify an ideal of the poset that $\hat{G}(\bar{\lambda})$ represents that corresponds to a min cut S of $G(\bar{\lambda})$ with $\mathrm{sl}^-(S) = \mathrm{sl}^-(\bar{\lambda})$. First, note that the max cut computation automatically picks out only ideals in $\hat{G}(\bar{\lambda})$. It is easy to see that if min cut S in $G(\bar{\lambda})$ corresponds to ideal I of $\hat{G}(\bar{\lambda})$ then $\mathrm{sl}^-(S) = \overline{\mathrm{cap}}(I)$, and so a max cut I in $\hat{G}(\bar{\lambda})$ does indeed pick out a min cut S attaining the value $\mathrm{sl}^-(\bar{\lambda})$.

[One technical detail: this doesn't quite work when $\bar{\lambda} = 0$. In this case $u_{ij}(\bar{\lambda}) \equiv 0$ for all $i \to j$, and so the residual graph has no arcs, and so $\hat{G}$ properly has all nodes as its components, and no arcs. Since there are no arcs, the computation of $\mathrm{sl}^+(0)$ won't do anything. Of course, in this case it is easy to compute $\mathrm{sl}^+(0)$, as it is just cap^*_r.]

Notice that the min cut computations in the two versions of $\hat{G}(\bar{\lambda})$ are on networks with special structure: they are acyclic. When $r \equiv 1$, then their capacities (in fact, sums of capacities) are bounded by m, i.e., $U = O(m)$. Thus these max flows could be done via Dinic in $O(mn)$ time, or by Goldberg-Rao in $O(m\Lambda \log n)$ time, which is usually even quicker.

Proofs of Section 5

Proof of Lemma 1 We start with the following lemma:

Lemma 3. *Let μ' be an optimal μ in the definition of $z(y)$. If $r_e = 1$ on each arc e, without loss of generality, we can assume that μ' is almost-integral, i.e. such that at most one component of μ' is fractional.*

Proof. Let μ' be an optimal μ and suppose that there exist arcs g and f such that $\mu'_g, \mu'_f \in (0,1)$. Observe that, for any $\varepsilon > 0$, the paths that "feel" g, i.e., those paths for which μ'_g is maximum, will still feel g if we increase μ'_g by ε (and possibly other new paths will also feel g). To the contrary, the value on the set of paths that "feel" f can only reduce if we decrease μ'_f by ε. Therefore, if we assume, without loss of generality, that with respect to μ', the total y-value on the paths that feel g is larger than the total y-value on the paths that feel f, there exists $\varepsilon > 0$ such that, by increasing μ'_g by ε and decreasing μ'_f by ε, the new μ vector stays feasible and either μ_g or μ_f is integer, and $\sum_{P\in\mathcal{P}} y_P \max_{e\in P} \mu_e \geq \sum_{P\in\mathcal{P}} y_P \max_{e\in P} \mu_e$. Then the statement follows by induction.

It follows from the above lemma that, when B is integer and $r_e = 1$ on each arc e, there is an optimal μ in the definition of $z(y)$ that is integral. Therefore, in this case, imposing integrality, as to forbid fractional deletion, is not harmful. Therefore, when $B = 2$ and $r_e = 1$ on each arc e, MRF is NP-hard.

Proof of Lemma 2

Proof. For arbitrary $\mu \in [0,1]^E$, we define the three functions $G_y(\mu) := \sum_{P\in\mathcal{P}} (\max_{e\in P} \mu_e) y_P$, $F_y(\mu) := \sum_{P\in\mathcal{P}} (1 - \Pi_{e\in P}(1-\mu_e)) y_P$, and $L_y(\mu) := \sum_{P\in\mathcal{P}} \min\{1, \sum_{e\in P} \mu_e\} y_P$. Note that $G_y(\bar{\mu}) = F_y(\bar{\mu}) = L_y(\bar{\mu})$ holds for any almost-integral $\mu \in [0,1]^E$, so whenever $\bar{\mu}$ has at most one fractional component.

Note that inequality $F_y(\mu) := \sum_{P\in\mathcal{P}} (1 - \Pi_{e\in P}(1-\mu_e)) y_P$ follows from $1 - \Pi_{e\in P}(1-\mu_e) \geq (1-(1-\frac{1}{k})^k) \min\{1, \sum_{e\in P} \mu_e\}$, an inequality which has been used for the first time by Goemans and Williamson [10], but also later in, e.g., [1]. For the sake of completeness we derive its proof below: By using arithmetic-geometric mean inequality it follows that $1 - \Pi_{e\in P}(1-\mu_e) \geq 1 - (1 - \frac{w}{|P|})^{|P|}$, where $w = \min\{1, \sum_{e\in P} \mu_e\}$. Since function $g(w) = 1 - (1 - \frac{w}{|P|})^{|P|}$ is concave on $[0,1]$ with $g(0) = 0$ and $g(1) = 1 - (1 - \frac{1}{|P|})^{|P|} \geq \rho$, we finally obtain $g(w) \geq \rho w$, as desired.

Let μ' be the optimal μ in the definition of $f(y)$, i.e. $f(y) = L_y(\mu')$. Using pipage-steps (similar as described in [1]) we can transform μ' into an almost-integral $\bar{\mu}$ such that $F_y(\bar{\mu}) \geq F_y(\mu')$. I These pipage procedure can be sketched as follows: Initially, set $\bar{\mu} = \mu'$. As long as $\bar{\mu}$ contains at least two fractional components e and f, set $\bar{\mu}_e(\epsilon) \leftarrow \bar{\mu}_e + \epsilon$, and $\bar{\mu}_f(\epsilon) \leftarrow \bar{\mu}_f - \epsilon \frac{r_e}{r_f}$. On all remaining arcs g set $\bar{\mu}_g(\epsilon) = \bar{\mu}_g$. Note that $F(\bar{\mu}(\epsilon))$ is convex on interval $[-\min\{\bar{\mu}_g, (1 - \bar{\mu}_f)\frac{r_f}{r_g}\}, \min\{1-\bar{\mu}_g, \bar{\mu}_f \frac{r_f}{r_g}\}]$. It follows that there is an endpoint ϵ^* of this interval such that $F(\bar{\mu}(\epsilon^*)) \geq F(\bar{\mu})$. Set $\bar{\mu} \leftarrow \bar{\mu}(\epsilon^*)$, and iterate. Summarizing, we get the desired result

$$z(y) \geq G_y(\bar{\mu}) = F_y(\bar{\mu}) \geq F_y(\mu') \geq \rho L_y(\mu') = \rho f(y)$$

.

Proof of Theorem 3

Proof. We first show (i). Let $y = \{y_P, P \in \mathcal{P}\}$ be a path-decomposition of $\bar{x}$ and let μ be an optimal solution in the definition of $f(y)$. We have:

$$Rval(y) \geq g(y) = val(\bar{x}) - \sum_{P\in\mathcal{P}} y_P \min\{1, \sum_{e\in P} \mu_e\} \geq val(\bar{x}) - \sum_{P\in\mathcal{P}} y_P \sum_{e\in P} \mu_e \geq val(\bar{x}) - \sum_{e\in E} \mu_e \sum_{P\in\mathcal{P}:e\in P} y_P =$$

$$= val(\bar{x}) - \sum_{e\in E} \mu_e \bar{x}_e \geq val(\bar{x}) - \sum_{e\in E} \mu_e r_e \bar{\lambda} \geq val(\bar{x}) - B\bar{\lambda}.$$

To show (ii), consider a path-flow $y^* = \{y^*_P, P \in \mathcal{P}\}$ with maximum robust value. If we consider the dual to the linear program defining $f(y^*)$, we obtain:

$$\begin{array}{rll} f(y^*) = \min & B\lambda + \sum_{P\in\mathcal{P}} \beta_P & \\ s.t. & \sum_{P\in\mathcal{P}:e\in P} \alpha_P \leq r_e\lambda & \forall e \in E \\ & \alpha_P + \beta_P \geq y^*_P & \forall P \in \mathcal{P} \\ & (\lambda), \alpha_P, \beta_P \geq 0 & \forall P \in \mathcal{P}. \end{array}$$

Since $g(y^*) = val(y^*) - f(y^*)$, it follows that

$$\begin{array}{rll} g(y^*) = \max & \sum_{P\in\mathcal{P}} y^*_P - B\lambda - \sum_{P\in\mathcal{P}} \beta_P & \\ s.t. & \sum_{P\in\mathcal{P}:e\in P} \alpha_P \leq r_e\lambda & \forall e \in E \\ & \alpha_P + \beta_P \geq y^*_P & \forall P \in \mathcal{P} \\ & \alpha_P, \beta_P \geq 0 & \forall P \in \mathcal{P}. \end{array}$$

Note that we can assume that an optimal solution $(\tilde{\alpha}, \tilde{\beta}, \tilde{\lambda})$ to the latter LP is such that $\tilde{\alpha}_P \leq y^*_P$, for each $P \in \mathcal{P}$. Otherwise, if $\tilde{\alpha}_P > y^*_P$ for some path P, we can reduce the value $\tilde{\alpha}_P$ to y^*_P and, without changing anything else, maintain feasibility and do not decrease the objective value. Therefore, there exists an optimal solution to this LP such that $\tilde{\beta}_P = y^*_P - \tilde{\alpha}_P$, for each $P \in \mathcal{P}$. This implies that:

$$\begin{array}{rll} g(y^*) = \max & \sum_{P\in\mathcal{P}} \alpha_P - B\lambda & \\ s.t. & \sum_{P\in\mathcal{P}:e\in P} \alpha_P \leq r_e\lambda & \forall e \in E \\ & 0 \leq \alpha_P \leq y^*_P & \forall P \in \mathcal{P}. \end{array}$$

Define $\alpha'_e := \sum_{P\in\mathcal{P}:e\in P} \tilde{\alpha}_P$ and $x'_e := \sum_{P\in\mathcal{P}:e\in P} y^*_P$, for each $e \in E$. Note that $(\alpha', x', \tilde{\lambda})$ yields a feasible solution to the following problem

$$\begin{array}{rlll} w(\alpha, x, \lambda) & = \max & val(\alpha) - B\lambda & \\ & s.t. & \text{flow conservation on } \alpha \text{ and on } x & \\ (SP^*) & 0 \leq x_e \leq u_e & & \forall e \in E \\ & \alpha_e \leq r_e\lambda & & \forall e \in E \\ & 0 \leq \alpha_e \leq x_e & & \forall e \in E. \end{array}$$

Moreover, note that the value of the objective function w evaluated at $(\alpha', x', \tilde{\lambda})$ is still $g(y^*)$. We now give a look at Problem (SP^*). Clearly, it has an optimal solution with $x_e := \alpha_e$, for each $e \in E$. Such a solution is indeed a feasible solution to problem (SP), that we re-write in the following.

$$(SP) \quad \begin{array}{ll} \max_{x,\lambda \geq 0} & val(x) - B\lambda \\ s.t. & \text{conservation} \\ & x_e \leq \min\{u_e, r_e\lambda\} \ \forall e \in E. \end{array}$$

Vice-versa, each feasible solution to (SP) can be extended to a feasible solution to (SP^*) by setting $\alpha_e := x_e$, for each $e \in E$. A quick check on the objective functions shows that problem (SP^*) and (SP) are indeed equivalent, and therefore $(\bar{x}, \bar{x}, \bar{\lambda})$ is an optimal solution to (SP^*).

Finally, recall that we have shown that any path-decomposition y of $\bar{x}$, has at least robust value $Rval(y) \geq val(\bar{x}) - B\bar{\lambda}$, and the last term is indeed the optimal value for problem (SP) and (SP^*). Since $(\alpha', x', \tilde{\lambda})$ yields a feasible solution to (SP^*) with value $Rval(y^*)$, it follows that $val(\bar{x}) - B\bar{\lambda} \geq g(y^*)$.

(iii) If we now assume that $y = \{y_P, P \in \mathcal{P}\}$ is a path-decomposition of $\bar{x}$, by combining (i), (ii) and Lemma 2, we have that:

$$Rval(y) \geq val(\bar{x}) - B\bar{\lambda} \geq g(y^*) = val(y^*) - f(y^*) \geq val(y^*) - \frac{1}{\rho} z(y^*) = \frac{1}{\rho} Rval(y^*) - \frac{1-\rho}{\rho} val(y^*).$$

Therefore:

$$0 \leq Rval(y^*) - Rval(y) \leq Rval(y^*) - \frac{1}{\rho} Rval(y^*) + \frac{1-\rho}{\rho} val(y^*) = \frac{1-\rho}{\rho}(val(y^*) - Rval(y^*)).$$

Further details about Chen-type scheduling problems

One of the oldest problems in scheduling is the following: We have n jobs, where job j has *processing requirement* p_j, and *due date* d_j. There are no precedence constraints among the jobs, and jobs can be *preempted*, i.e., the processing of any job can be split into *pieces* that execute at different times and/or on different machines. These n jobs are to be scheduled on m machines. The machines might be *identical*, meaning that they all have the same speed, or *uniform*, meaning that machine i has *speed* σ_i (i.e., if job j spends t time units on machine i, then $\sigma_i t$ of its processing requirement is finished). Given some schedule, the *completion time* C_j of job j is the latest time that any piece of job j finishes. We usually prefer a schedule where every job finishes as early as possible.

In Chen's problems [6], each job j also has a *release date* r_j. No piece of any job can start before its release date, and every piece must finish before its due date, i.e., we must have $C_j \leq d_j$ for all jobs j. For now we assume that the m machines are identical. There is no objective function, we ask only if a feasible schedule exists. Of course, if we can answer this problem in polynomial time, then we can, e.g., use binary search in polynomial time to minimize *makespan* (the largest completion time of any piece of any job).

It is well-known how to solve this problem by constructing a max flow network as follows. Sort the release times and due dates into a single increasing sequence $t_1 < t_2 < \cdots < t_{k+1}$, where $k \leq 2n - 1$. These t_i then define k time *intervals* $I_1, \ldots, I_k$, where $I_i = [t_i, t_{i+1}]$ has size $\Delta_i = t_{i+1} - t_i$. Now the network has a

node for each job j, a node for each interval i, a source s, and a sink t. For each job j, there is an arc $s \to j$ with capacity p_j, and for each interval I_i contained in $[r_j, d_j]$, there is an arc $j \to i$. Finally, each interval I_i has an arc $i \to t$ with capacity $m\Delta_i$, see Figure 4.

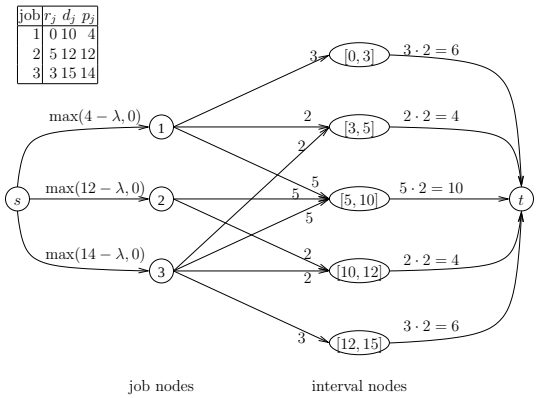

Fig. 4. Example of Chen's network with three jobs and two machines. The given data yields five intervals, which in turn give the capacities indicated on the arcs from the job nodes to the interval nodes (capacity equals interval length), and from the interval nodes to t (capacity equals (interval length)$\cdot$(number of machines = 2)). The processing times give the parametric capacities on the arcs from s to the job nodes. Here the minimum value of λ that allows a flow saturating arcs at s is 5..

Since two pieces of a job are not allowed to execute in parallel with each other, job j can be processed for at most Δ_i time units during interval I_i, and the capacity of $j \to i$ is Δ_i.

Theorem 5. *The max flow value in the network of Figure 4 equals $\sum_j p_j$ (i.e., saturates the source) if and only if there is a feasible schedule.* □

If there is no feasible schedule, Chen [6] suggests that what is often done in practice is to reduce the processing time necessary for some or all jobs by, e.g., outsourcing. This leads Chen to propose a model where the processing time of job j is $\max(0, p_j - \lambda a_j)$, where λ is the amount we pay to reduce p_j. He proposed two models; here we focus on the one where paying λ simultaneously reduces all p_j's and whose objective is to minimize λ subject to having a feasible schedule. In this model the capacity of arc $s \to j$ is $p_j(\lambda) \equiv \max(0, p_j - \lambda a_j)$, which satisfies the monotonicity condition necessary to apply GGT. The objective is to find the minimum value of λ such that there exists a flow (schedule) w.r.t. capacities $p_j(\lambda)$ that saturates all arcs at s.

There was a long-standing open problem here (posed by Chen [6] and [13]) where we extend to considering multiple parameters. Suppose that we have two

parameters, λ and μ, and that the parametric processing time of job j if we pay $\$\lambda$ and $\$\mu$ is $p_j(\lambda,\mu) \equiv \max(0, p_j - a_j\lambda - b_j\mu)$ ([13, Section 8] suggests the narrower case where the jobs with non-zero a_j are disjoint from the jobs with non-zero b_j, but the proposed algorithms below don't require this). Now we want to minimize $\lambda + \mu$ s.t. there exists a feasible flow w.r.t. capacities $p_j(\lambda,\mu)$ that saturates the arcs at s.

Consider the graph with horizontal axis λ, and vertical axis μ. Points (λ,μ) are either feasible (if a feasible flow saturating arcs at s exists w.r.t. $p_j(\lambda,\mu)$), or infeasible. We are assuming that $(0,0)$ is infeasible, and that for sufficiently large (λ,μ) the problem is feasible (i.e., that at least one of a_j or b_j is positive for each j). Let $\mathcal{F} \equiv \{(\lambda',\mu') \geq 0 \mid$ there is a flow saturating s in the network with capacities $\max(0, p_j - a_j\lambda' - b_j\mu')\}$, the set of feasible points, and $\mathcal{I}$ be its complement, the set of infeasible points, see Figure 5.

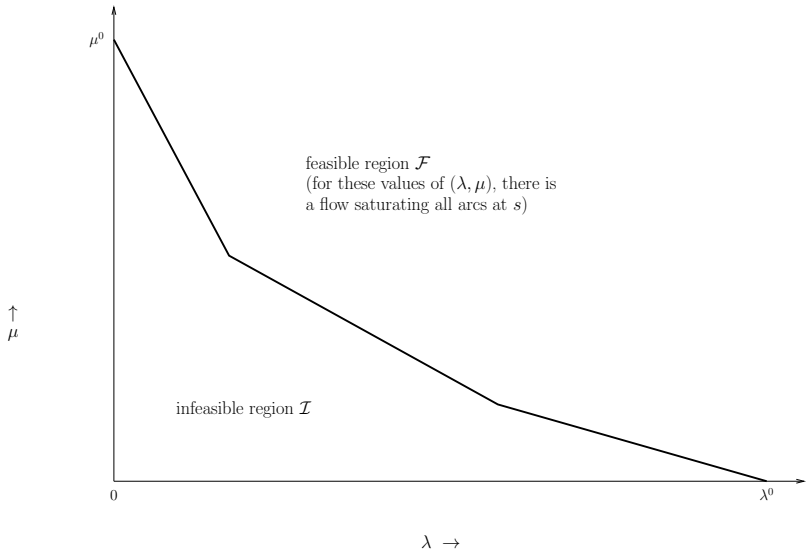

Fig. 5. The heavy black line is the boundary between the set of feasible points $\mathcal{F}$ and infeasible points $\mathcal{I}$ in the (λ,μ) plane.

Let S be some cut of the network with $\delta^-(S)$ including at least one $s \to j$ arc, and let D be any non-empty subset of $\delta^-(S)$. For simplicity abbreviate $\sum_{s\to j\in D} a_j$ with $a(D)$. The total desired flow into S through the arcs of D is $\sum_{s\to j\in D}(p_j - a_j\lambda - b_j\mu)$, and the total capacity going out of S is $c(\delta^+(S))$. If we are to have a feasible flow saturating all arcs at s, then we must make λ and μ large enough that $\sum_{s\to j\in D}(p_j - a_j\lambda - b_j\mu)$ is at most $c(\delta^+(S))$, or

$$\lambda a(D) + \mu b(D) \geq p(D) - c(\delta^+(S)). \tag{1}$$

Call this type of inequality a *cut-subset* constraint.

If we take the union of cut-subset constraints over all possible non-empty $D \subseteq \delta^-(S)$, then we get the full set of constraints coming from cut S. It can be seen that this set of constraints induces a piecewise linear convex boundary between infeasible and potentially feasible points. The constraint for D will be active only in the region where arcs in D still have positive capacities, and arcs not in D have capacity zero (this is true because it is easy to see that constraint (1) for $D+e$ dominates the one for D when (λ, μ) is such that $p_e - a_e\lambda - b_e\mu > 0$, i.e., the parametric capacity of e is positive, as adding e makes (1) tighter). Finally, if we take non-negativity plus the union of all cut-subset constraints over all cuts, and all possible D for each cut, then we get a polyhedron, which clearly contains $\mathcal{F}$.

Lemma 4. *Region $\mathcal{F}$ is the polyhedron defined by all the cut-subset constraints and non-negativity.*

Proof. Take some $(\lambda', \mu') \in \mathcal{I}$ with $(\lambda', \mu') \geq 0$, and compute a max flow. Since $(\lambda', \mu') \in \mathcal{I}$, the max flow does not saturate all arcs at s, and so the min cut S^* must be of the correct type. In particular, there must still be at least one arc $s \to j \in \delta^-(S^*)$ whose capacity is still positive w.r.t. (λ', μ'); define D^* to be the set of such arcs. Then it is easy to see that (λ', μ') violates the cut-subset constraint for S^* and D^*. Since every non-negative point $(\lambda', \mu') \in \mathcal{I}$ is separated from $\mathcal{F}$ by a cut-subset constraint, in fact $\mathcal{F}$ is defined by the cut-subset constraints.

Notice that the proof of Lemma 4 easily generalizes to more than two parameters to show that the feasible region is in general a polyhedron defined by the obvious generalization of cut-subset constraints.

Lemma 4 shows that the boundary between $\mathcal{F}$ and $\mathcal{I}$ is a piecewise linear convex curve as shown in Figure 5. We want to find a point (λ', μ') on this curve such that the local slopes at (λ', μ') bracket -1, an instance of the Slope Problem, and so Newton-B solves it.