Joint Resource Allocation and Routing in Wireless Networks via Convex Approximation Techniques

Evangelia Matskani

Advisor: Professor Nicholas D. Sidiropoulos

PhD Thesis

DEPARTMENT OF ELECTRONIC & COMPUTER ENGINEERING
TECHNICAL UNIVERSITY OF CRETE

June 2012

To my family

Acknowledgments

I would like to express my deep and sincere gratitude to my advisor Professor Nicholas Sidiropoulos for his attentive guidance, endless patience and warm encouragement through the past years. I appreciate his effort to introduce me to the world of research, as well as his contributions of time, ideas and funding to make my Ph.D experience productive and stimulating. I am thankful for the valuable knowledge and insights I have acquired through his instructions. He is a great teacher, but I also admire him for his way of thinking and vigor in life. It has been an honor to be one of his Ph.D students.

I would like to convey my sincere thanks to Professor Zhi-Quan (Tom) Luo and Professor Leandros Tassiulas, with whom I have co-authored several publications, for contributing to the research and sharing their insights and perspectives. I would also like to thank my Ph.D committee members, Professors Athanasios Liavas, Michael Paterakis, Aggelos Bletsas and Polychronis Koutsakis for serving on my committee, and for teaching me directly and indirectly through my years at the Technical University of Crete. I also owe special thanks to my colleague and friend Eleftherios Karipidis, for our collaboration and his kind help, in many different ways, through my early years in Technical University of Crete. Finally, I would like to thank my colleagues and friends Vangelis Papalexakis, Eythimios Tsakonas, Alexis Balatsoukas, and Georgina, for sharing their ideas, and generally, for their concern and moral support.

This research is co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund.

Abstract

The major part of this thesis concerns the joint back-pressure routing and power control in the context of cross-layer wireless networking. Throughput-optimal multi-hop wireless network operation entails a key physical-layer optimization problem: maximizing a weighted sum of link rates, with weights given by the differential queue backlogs. This emerges in joint back-pressure routing and power control, which is central in cross-layer wireless networking. We begin by showing in chapter 1 that the core problem is not only non-convex, but also NP-hard. This is a negative result, which however comes with a positive flip side: drawing from related developments in the digital subscriber line (DSL) literature, we propose effective ways to approximate it. Exploiting quasi-periodicity of the power allocation in stable setups due to the push-pull nature of the solution, we derive two custom algorithms that offer excellent throughput performance at reasonable, worst-case polynomial complexity. Judicious simulations illustrate the merits of the proposed algorithms.

Back-pressure power control (BPPC) is amenable to successive convex approximation strategies that deliver manifold improvements in end-to-end throughput relative to the prior art in wireless networking, as our findings in chapter 1 suggest. A drawback is that existing implementations are centralized, whereas practical power control has to be distributed across the network. The work presented in chapter 2 fills this gap by developing distributed implementations of approximations of the BPPC problem. Two approximation approaches to the NP-hard underlying problem are utilized, while feedback requirements and consensus-on-termination issues are also addressed, in order to come up with fully decentralized protocols that allow tight approximation of the BPPC objective in all interference regimes. The first distributed implementation proposed is based on the successive convex approximation approach, and the Alternating Direction Method of Multipliers is utilized towards distributed implementation of the core step of the approach, which is the convex lower-bounding approximation of BPPC, at any operating point.

Judicious simulations verify that the distributed algorithm derived matches the performance of its centralized counterpart. The iteratively re-weighted Minimum Mean Square Error approach to weighted sum-rate maximization for the MIMO interference channel, is also exploited in our context. A second distributed implementation is then possible, providing an one-shot approximate solution to the BPPC problem. Further exploiting quasi-periodicity of the solution arising in stable setups, due to the push-pull evolution of the network in our context, and upon elaboration on distributed warm start, we derive custom adaptive versions of the distributed algorithms, that enjoy low average complexity with no performance loss. Extensive simulation experiments reveal the potentials of the proposed algorithms, and allow comparison of the two approximation approaches to the BPPC problem.

The joint multicast beamforming and admission control problem in the co-channel multicast context is considered in chapter 3. Multicast beamforming was recently proposed as a means of exploiting the broadcast nature of the wireless medium to boost spectral efficiency and meet Quality of Service (QoS) requirements. Infeasibility is a key issue in this context, due to power or mutual interference limitations. We therefore consider the joint multicast beamforming and admission control problem for one or more co-channel multicast groups, with the objective of maximizing the number of subscribers served and minimizing the power required to serve them. The problem is NP-hard even for an isolated multicast group and no admission control; but drawing upon our earlier work for the multiuser SDMA downlink, we develop an efficient approximation algorithm that yields good solutions at affordable worst-case complexity. For the special case of an isolated multicast, Lozano proposed a particularly simple adaptive algorithm for implementation in UMTS-LTE. We identify strengths and drawbacks of Lozano's algorithm, and propose two simple but worthwhile improvements. All algorithms are carefully tested on publicly available indoor/outdoor measured channel data.

Contents

List of Figures

Chapter 1

Convex Approximation Algorithms for Back-Pressure Power Control

Throughput-optimal multi-hop wireless network operation entails a key physical-layer optimization problem: maximizing a weighted sum of link rates, with weights given by the differential queue backlogs. This emerges in joint back-pressure routing and power control, which is central in cross-layer wireless networking. We begin by showing that the core problem is not only non-convex, but also NP-hard. This is a negative result, which however comes with a positive flip side: drawing from related developments in the digital subscriber line (DSL) literature, we propose effective ways to approximate it. Exploiting quasi-periodicity of the power allocation in stable setups due to the push-pull nature of the solution, we derive two custom algorithms that offer excellent throughput performance at reasonable, worst-case polynomial complexity. Judicious simulations illustrate the merits of the proposed algorithms.

1.1 Introduction

Since its inception in the early '90's [54], back-pressure routing has won much acclaim as a (surprisingly) simple and effective adaptive routing solution that is optimal in terms of throughput. It was subsequently generalized in many ways (see [14] for a recent tutorial overview) and currently forms the backbone of emerging approaches aiming to optimize other performance objectives [42], [52].

Back-pressure policies owe their popularity to their natural agility and robustness (e.g., with

respect to link failures and traffic dynamics). Back-pressure implementations for wired unicast networks are lightweight in terms of computation and signaling overhead, but the situation can be very different in wireless networks, due to fading, shadowing, and mutual interference. Along with these challenges come new opportunities, however: for example, it is possible to employ power control to obtain a more favorable 'topology' from the viewpoint of maximizing throughput.

There is a lot of recent activity on optimization based approaches for network control and management, in particular for wireless networks. Advances in wireless technology over the last several years resulted in systems with much increased computational power at each radio node, while at the same time the channel sensing and measurement capabilities increased significantly as well. As a result, fairly sophisticated approaches for real-time network control become feasible. On the other hand, several recent theoretical advances in optimization-based network control increased our understanding of the theoretical performance limits that may be pursued in these systems [8, 14, 20, 24, 41, 42, 52, 53, 56].

Here we consider the joint routing and power control problem for maximal end-to-end throughput in a wireless multihop network. From the work of Tassiulas *et al* [14, 41, 53–56], it is known that the following policy is optimal in the sense of enabling maximal stable throughput: choose link powers to maximize a *differential backlog* - weighted sum of link capacities; then route at each node using back-pressure. We call this policy *back-pressure power control* (BPPC). Our purpose here is to investigate the structure and properties of the BPPC problem, and come up with a suitable algorithm to solve it.

A conference version of part of this work appears in *Proc. IEEE ICASSP 2011* [32]. The conference version [32] presents the basic ideas, batch algorithms, and illustrative simulation results. In this work we add custom algorithms, proofs and derivations, comprehensive simulations, and a fleshed-out discussion of results and insights.

1.2 System Model

Consider a wireless multi-hop network comprising N nodes. The topology of the network is represented by the directed graph $(\mathcal{N}, \mathcal{L})$, where $\mathcal{N} := \{1, \ldots, N\}$ and $\mathcal{L} := \{1, \ldots, L\}$ denote the set of nodes and the set of links, respectively. Each link $\ell \in \mathcal{L}$ corresponds to an ordered

pair (i,j), where $i,j \in \mathcal{N}$ and $i \neq j$. Let $\mathrm{Tx}(\ell)$ and $\mathrm{Rx}(\ell)$ denote the transmitter and the receiver of link ℓ, i.e., when $\ell = (i,j)$, then $\mathrm{Tx}(\ell) = i$ and $\mathrm{Rx}(\ell) = j$. Data can be transmitted from any node to any other node, and each node may split its incoming traffic into multiple outgoing links. Let p_ℓ denote the power transmitted on link ℓ, and $G_{\ell k}$ the aggregate path loss between the transmitter of link ℓ and the receiver of link k. Then, the Signal to Interference plus Noise Ratio (SINR) experienced by the receiver of link ℓ is

$$\gamma_\ell = \frac{G_{\ell\ell} p_\ell}{\frac{1}{G}\sum_{\substack{k=1 \\ k\neq\ell}}^{L} G_{k\ell} p_k + V_\ell}, \tag{1.1}$$

where V_ℓ is the background noise power, and G models spreading / beamforming gain, if any. The transmission rate on link ℓ is upper bounded by the maximum achievable rate c_ℓ, given by[1]

$$\begin{aligned} c_\ell &= \log(1+\gamma_\ell) \overset{(1.1)}{=} \\ &= \log\left(\sum_{k=1}^{L} G_{k\ell} p_k + V_\ell\right) - \log\left(\sum_{\substack{k=1 \\ k\neq\ell}}^{L} G_{k\ell} p_k + V_\ell\right), \end{aligned} \tag{1.2}$$

where G has been absorbed in the coupling factors $G_{k\ell},\ k \neq \ell$. Due to (1.1), c_ℓ is a function of all the transmitted powers in the network.

1.3 Maximum Throughput

We assume that the system is slotted in unit time slots, indexed by t. Let us begin by considering a single flow in the network: traffic stems from node 1 (the source) and traverses the network to reach node N (the destination). Let $W_i(t)$ denote the queue length of node i at the end of slot t. The *differential backlog* [54] of link $\ell = (i,j)$, at the end of slot t is defined as

$$D_\ell(t) := \begin{cases} \max\{0, W_i(t) - W_j(t)\}, & j \neq N \\ W_i(t), & j = N. \end{cases} \tag{1.3}$$

Traffic flows through the links during each slot, based on the link capacities resulting from the power allocation at the beginning of the slot. The powers for slot $t+1$ are to be determined by solving the following optimization problem [14, 15, 21, 41, 53–56]

$$\mathsf{BPPC}: \qquad \max_{\{p_\ell\}_{\ell\in\mathcal{L}}} \sum_{\ell=1}^{L} D_\ell(t) c_\ell \tag{1.4}$$

[1] The usual SINR gap parameter Γ can be introduced in the Shannon capacity formula to account for modulation loss, coding, etc. We skip this for brevity.

$$\text{subject to} \quad 0 \leq \sum_{\ell:\mathrm{Tx}(\ell)=i} p_\ell \leq P_i, \quad i \in \mathcal{N}. \tag{1.5}$$

The objective function in (1.4) is a weighted sum of the capacities of all network links, where the differential backlogs serve as weighting factors. The optimization problem (1.4)–(1.5) aims to maximize the throughput of the network by favoring the links whose receiver is less congested than the transmitter. Note that, from (1.3), links destined to more congested nodes will have low differential backlogs, and are thus down-weighted in (1.4). Inequalities (1.5) upper bound the total transmission power of each node. Per-link power constraints can be used instead of (or together with) per-node power constraints, without changing the nature of the problem and the solutions proposed in the sequel.

Extension of the above throughput-optimal policy to the case of multiple flows turns out being surprisingly simple [54]. The only difference in the case of multiple flows is that each node maintains a separate queue for each flow, and each link computes the maximum differential backlog across all flows traversing the link. The BPPC problem is solved using these maximum differential backlogs as weights on link capacities, and each link then carries a flow that achieves the link's maximum differential backlog (winner-takes-all). This policy is throughput-optimal for multiple flows [54]. A more detailed discussion of the case of multiple flows (commodities) in our particular context can be found in the Appendix (see also [15]). It follows that our results are directly applicable to the case of multiple flows; we continue with a single flow for simplicity of exposition.

Unfortunately the objective function in (1.4) is nonconvex in the optimization variables p_l. This can be appreciated by rewriting the problem using (1.2)

$$\max_{\{p_\ell\}_{\ell\in\mathcal{L}}} \sum_{\ell=1}^{L} D_\ell(t) \left(\log\left(\sum_{k=1}^{L} G_{k\ell} p_k + V_\ell \right) - \log\left(\sum_{\substack{k=1 \\ k\neq\ell}}^{L} G_{k\ell} p_k + V_\ell \right) \right) \tag{1.6}$$

$$\text{subject to} \quad 0 \leq \sum_{\ell:\mathrm{Tx}(\ell)=i} p_\ell \leq P_i, \quad i \in \mathcal{N}. \tag{1.7}$$

The logarithm of an affine expression is concave, but the difference of concave functions is, in general, neither convex nor concave. This raises the question of whether or not the BPPC problem can be efficiently solved. We have the following claim, whose proof can be found in the Appendix.

Claim 1. *Back-pressure power control is NP-hard.*

Back-pressure power control *looks like* the multiuser sum-rate maximization problem for the interference channel, but there is an important difference between the two: the $G_{k,\ell}$ parameters are subject to certain restrictions for the former, versus completely free for the latter. This means that the NP-hardness proof in [29] cannot be directly invoked. Another difference is that back-pressure power control may be subject to per-node (instead of per-link) power constraints, which couple the transmission power across multiple links. The key to a simple and clear proof is *backlog reduction*: realizing that there is freedom to choose the backlogs in such a way that we peel off these complicating factors to reveal the multiuser sum-rate maximization problem as a special case. Details can be found in the Appendix.

While formal proof of NP-hardness of BPPC was missing, and our work closes this gap, earlier work had already recognized that BPPC is a non-convex problem that is likely difficult to solve. It is in part for this reason that [15] proposed a greedy on-line optimization approach. In the following, we take a different, *disciplined convex approximation* approach.

1.4 Convex Approximation

We develop a convex approximation algorithm by borrowing an idea originally developed in the dynamic spectrum management for digital subscriber lines literature [45, 46]. The main point is to lower-bound the individual link rates using a concave function. In particular, we will use the following bound [45, 46]

$$\alpha \log(z) + \beta \leq \log(1+z) \text{ for } \begin{cases} \alpha = \frac{z_o}{1+z_o}, \\ \beta = \log(1+z_o) - \frac{z_o}{1+z_o}\log(z_o) \end{cases} \tag{1.8}$$

which is tight at z_o. Notice that as $z_o \to \infty$, the bound becomes $\log(z) \leq \log(1+z)$.

Applying (1.8) to

$$\max_{\{p_\ell\}_{\ell\in\mathcal{L}}} \sum_{\ell=1}^{L} D_\ell(t) \log\,(1+\gamma_\ell) \tag{1.9}$$

$$\text{subject to} \quad 0 \leq \sum_{\ell:\mathrm{Tx}(\ell)=i} p_\ell \leq P_i, \quad i \in \mathcal{N}. \tag{1.10}$$

results in the approximation[2]

$$\max_{\{p_\ell\}_{\ell\in\mathcal{L}}} \sum_{\ell=1}^{L} D_\ell(t)(\alpha_\ell(t)\log(\gamma_\ell)+\beta_\ell(t)) \tag{1.11}$$

$$\text{subject to} \quad 0 \le \sum_{\ell:\mathrm{Tx}(\ell)=i} p_\ell \le P_i, \quad i\in\mathcal{N}. \tag{1.12}$$

The maximization problem (1.11)-(1.12) is still nonconvex, since the objective is not concave in the variables p_ℓ. However, notice that

$$\log(\gamma_\ell) \overset{(1.1)}{=} \log(G_{\ell\ell}p_\ell) - \log\left(\sum_{\substack{k=1\\k\neq\ell}}^{L} G_{k\ell}p_k + V_\ell\right). \tag{1.13}$$

Introducing a logarithmic change of variables

$$\tilde{p}_\ell := \log(p_\ell), \tag{1.14}$$

yields

$$\log(\gamma_\ell) = \tilde{G}_{\ell\ell} + \tilde{p}_\ell - \log\left(\sum_{\substack{k=1\\k\neq\ell}}^{L} e^{\tilde{G}_{k\ell}+\tilde{p}_k} + e^{\tilde{V}_\ell}\right), \tag{1.15}$$

where $\tilde{G}_{k_\ell} := \log(G_{k_\ell})$ and $\tilde{V}_\ell := \log(V_\ell)$ for brevity. The logarithm of a sum of exponentials is convex, the minus sign reverts the curvature, and addition with an affine expression preserves curvature; hence, (1.15) is a concave function of $\{\tilde{p}_\ell\}_{\ell\in\mathcal{L}}$.

With respect to the optimization variables $\{\tilde{p}_\ell\}_{\ell\in\mathcal{L}}$, the problem of interest finally becomes

$$\max_{\{\tilde{p}_\ell\}_{\ell\in\mathcal{L}}} \sum_{\ell=1}^{L} D_\ell(t)\left(\alpha_\ell(t)\left(\tilde{G}_{\ell\ell}+\tilde{p}_\ell-\log\left(\sum_{\substack{k=1\\k\neq\ell}}^{L} e^{\tilde{G}_{k\ell}+\tilde{p}_k}+e^{\tilde{V}_\ell}\right)\right)+\beta_\ell(t)\right) \tag{1.16}$$

$$\text{subject to} \quad \log\left(\sum_{\ell:\mathrm{Tx}(\ell)=i} e^{\tilde{p}_\ell}\right) \le \tilde{P}_i := \log(P_i), \quad i\in\mathcal{N}. \tag{1.17}$$

The objective function (1.16) is concave, since it comprises a sum of linear and concave functions of $\{\tilde{p}_\ell\}_{\ell\in\mathcal{L}}$, cf. (1.15). The weights $D_\ell(t)$ are nonnegative, cf. (1.3), and so are the constants $\alpha_\ell(t)$ and $\beta_\ell(t)$, cf. (1.8). Moreover, inequalities (1.17) are convex, since, as noted before, the logarithm of a sum of exponentials is a convex function. Hence, optimization problem

[2]Since we maximize a lower bound, the solution obtained this way will yield a value for the original objective that is at least as high as the maximal lower bound.

(1.16)–(1.17) is *convex*, since the maximum of a concave function is sought over a convex feasible set. The logarithmic change of variables (1.14) is reminiscent of Geometric Programming (GP); indeed, (1.16)–(1.17) is an *Extended GP* problem [9].

In the high-SINR regime ($\gamma_\ell \gg 1$) the bound $\log(\gamma_\ell) \leq \log(1+\gamma_\ell)$ is tight: $\log(\gamma_\ell) \approx \log(1+\gamma_\ell)$. This is an often-made (and debated) approximation; in our present context, it has been adopted in [15] to simplify the problem and enable derivation of best-response type algorithms. The issue with the high-SINR approximation is that we do not know beforehand whether it holds well at the optimum - unless worst-case interference is so low that the problem is easy to begin with.

The high-SINR approximation corresponds to using $\alpha_\ell(t) = 1$ and $\beta_\ell(t) = 0$, $\forall \ell \in \mathcal{L}$ in (1.16)–(1.17), yielding the following approximation algorithm.

Algorithm 1. Batch high-SINR*: For each time slot t, calculate the differential backlogs and solve* (1.16)–(1.17) *with* $\alpha_\ell(t) = 1$, $\beta_\ell(t) = 0$, $\forall \ell \in \mathcal{L}$, *to obtain* $\mathbf{p}(t) = [p_1(t)...p_L(t)]^T$.

Solving the optimization problem in (1.16)–(1.17) maximizes a lower bound on the achievable differential backlog - weighted sum rate of all links. After obtaining $[p_1(t)...p_L(t)]^T$, the individual link rate bounds can be tightened by tuning the parameters $\alpha_\ell(t)$ and $\beta_\ell(t)$, $\forall \ell \in \mathcal{L}$ so that the bounds coincide with the link rates at $[p_1(t)...p_L(t)]^T$. This suggests the following successive approximation algorithm.

Algorithm 2. Batch Successive Approximation (Batch S.A.)*:*

1. *Initialization: For each time slot t, calculate the differential backlogs, reset iteration counter $s = 0$, and set $\alpha_\ell(t,s) = 1$ and $\beta_\ell(t,s) = 0$, $\forall \ell \in \mathcal{L}$.*
2. **repeat:**
3. *Maximization step: Solve* (1.16)–(1.17) *to obtain* $\mathbf{p}(t,s) = [p_1(t,s)...p_L(t,s)]^T$
4. *Tightening step: Pick $\alpha_\ell(t,s)$, $\beta_\ell(t,s)$ according to* (1.8) *for $z_o = \gamma_\ell(\mathbf{p}(t,s))$, see* (1.1), $\forall \ell \in \mathcal{L}$
5. $s = s + 1$
6. **until** *convergence of the objective value (within ϵ- accuracy).*

The sequence of iterates produces a monotonically increasing objective. This is a corollary of the *majorization principle*, e.g., cf. [6]. The idea of using the bound in (1.8) to obtain a convex lower approximation, then tightening the bound to refine the approximation is the essence of the SCALE algorithm in [45, 46], originally proposed for spectrum balancing in cross-talk limited digital subscriber line (DSL) systems. Interestingly, the same problem structure emerges in our present multi-hop cross-layer networking context, where the objective is to choose link powers and flows for maximal stable end-to-end throughput.

In [46], it is shown that the SCALE algorithm converges to a KKT point of the original non-convex power control problem. This also holds in our context, i.e., for each t, $\mathbf{p}(t,s)$ indexed by s converges to a KKT point of the original NP-hard BPPC problem. The argument in [46] carries over verbatim: the main point is that the lower bound approximation is exact at the converged solution.

1.5 Custom Algorithms

Unlike [45, 46], where the spectrum balancing problem is solved once (or 'infrequently'), and the weights used to compute the weighted sum rate objective are fixed design parameters, we have to solve the problem on a per-slot basis with a different set of weights - the differential backlogs, which change dynamically from one scheduling slot to the next, as packets are routed in the network. The objective function changes, and, as a result, the power vector computed in the previous slot may be far from a good solution for the present slot. This calls for custom algorithms that avoid solving the problem from scratch at each slot. Towards this end, we first convert (1.16)–(1.17) into an unconstrained optimization problem using a logarithmic barrier interior point method.

Consider (1.16) under per-link power constraints (per-node power constraints as in (1.17) can be similarly treated), and rewrite as

$$\min_{\{\tilde{p}_\ell\}_{\ell\in\mathcal{L}}} -f(\tilde{p}_1,\ldots,\tilde{p}_L) := -\sum_{\ell=1}^{L} D_\ell(t)\left(\alpha_\ell(t)\left(\tilde{G}_{\ell\ell}+\tilde{p}_\ell-\log\left(\sum_{\substack{k=1\\k\neq\ell}}^{L} e^{\tilde{G}_{k\ell}+\tilde{p}_k}+e^{\tilde{V}_\ell}\right)\right)+\beta_\ell(t)\right) \tag{1.18}$$

$$\text{subject to} \quad \tilde{p}_\ell \le \tilde{P}_\ell\ , \quad \ell\in\mathcal{L}. \tag{1.19}$$

where now the cost function $-f$ is convex. Converting the explicit link-power constraints into

implicit ones, we seek to

$$\min_{\{\tilde{p}_\ell\}_{\ell\in\mathcal{L}}} -f(\tilde{\mathbf{p}}) + \sum_{\ell=1}^{L} I_-(\tilde{p}_\ell - \tilde{P}_\ell) \tag{1.20}$$

where

$$I_-(u) := \begin{cases} 0, & u \le 0 \\ +\infty, & u > 0. \end{cases} \tag{1.21}$$

Interior point methods approximate this using the convex differentiable function

$$\widehat{I}_-(u) = -\frac{1}{\tau}\log(-u), \ for\ \tau > 0, \tag{1.22}$$

which converges to $I_-(\cdot)$ as $\tau \to \infty$. In practice, it suffices to pick τ large enough[3]. This yields the unconstrained convex minimization problem

$$\min_{\{\tilde{p}_\ell\}_{\ell\in\mathcal{L}}} -\tau f(\tilde{\mathbf{p}}) - \sum_{\ell=1}^{L} \log(\tilde{P}_\ell - \tilde{p}_\ell), \ \tau \gg 0, \tag{1.23}$$

i.e.,

$$\min_{\{\tilde{p}_\ell\}_{\ell\in\mathcal{L}}} \tilde{f}(\tilde{\mathbf{p}}) := -\tau \left\{ \sum_{\ell=1}^{L} D_\ell(t) \left(\alpha_\ell(t) \left(\tilde{G}_{\ell\ell} + \tilde{p}_\ell - \log\left(\sum_{\substack{k=1 \\ k\neq\ell}}^{L} e^{\tilde{G}_{k\ell}+\tilde{p}_k} + e^{\tilde{V}_\ell} \right) \right) + \beta_\ell(t) \right) \right\} - \sum_{\ell=1}^{L} \log(\tilde{P}_\ell - \tilde{p}_\ell), \tag{1.24}$$

which can be solved via Newton's method to obtain a solution $\tilde{\mathbf{p}}^*(\tau)$ that approaches the optimal solution $\tilde{\mathbf{p}}^*$ of (1.18)-(1.19) as $\tau \to \infty$.

Recall that the motivation for considering custom solutions to the BPPC problem is that we have to solve it for each scheduling slot, with a different set of weights (the differential backlogs) and possibly time-varying $G_{k\ell}$'s. Even when the physical layer propagation conditions vary slowly with time, the differential backlogs can change swiftly from slot to slot.

Assuming deterministic fixed-rate (or random but bounded) arrivals and fixed physical layer propagation conditions, if all queues remain bounded then the system must exhibit periodic behavior - perhaps with a very long period. This is because there is a (large but still) finite number of system states, hence the system must return to a previously visited state in due time. The same holds for finite-state time-varying physical layer propagation conditions (e.g., of the on-off type) - we only need to augment the state vector to account for those. In practice we

[3] Or iterate with a gradually increasing τ.

typically observe far shorter periods, due to the need to protect links from excessive interference (including no-listen-while-you-talk considerations), which often implies that the best strategy is to schedule "independent" (quasi-) non-interfering subsets of links in subsequent slots. This way the system operates in a multi-stage push-pull fashion, giving rise to periodic or quasi-periodic behavior for stable setups.

Given the above, it is clear that the solution at slot t can be very different from the one at slot $t-1$; thus departing from the classical setting of adaptive algorithms. Not all is lost however: the key is to exploit the aforementioned (quasi-) periodicity. Even though the previously computed solution can be far from the one needed in the present slot, chances are that one of the already encountered solutions for past slots is close to the one for the present slot. This idea is exploited in the following two algorithms.

Algorithm 3. Adaptive high-SINR*:*

Fix $\alpha_\ell(t) = 1$, $\beta_\ell(t) = 0$, $\forall \ell \in \mathcal{L}$

For each time slot $t \geq 1$*:*

1. *Calculate the differential backlogs*

2. *Power initialization:*

   For $t = 1$ *draw random* $\tilde{\mathbf{p}}_o(t)$ *satisfying log-power constraints;*

   else for $t \in [2, W]$ *set:*

$$\tilde{\mathbf{p}}_o(t) = \arg \max_{\tilde{\mathbf{p}} \in \{\tilde{\mathbf{p}}(1), \ldots, \tilde{\mathbf{p}}(t-1)\}} f(\tilde{\mathbf{p}}) \tag{1.25}$$

   where

$$f(\tilde{\mathbf{p}}) := \sum_{\ell=1}^{L} D_\ell(t) \left(\alpha_\ell(t) \left(\tilde{G}_{\ell\ell} + \tilde{p}_\ell - \log \left(\sum_{\substack{k=1 \\ k \neq \ell}}^{L} e^{\tilde{G}_{k\ell} + \tilde{p}_k} + e^{\tilde{V}_\ell} \right) \right) + \beta_\ell(t) \right), \tag{1.26}$$

   else $(t \geq W + 1)$ *set:*

$$\tilde{\mathbf{p}}_o(t) = \arg \max_{\tilde{\mathbf{p}} \in \{\tilde{\mathbf{p}}(t-W), \ldots, \tilde{\mathbf{p}}(t-1)\}} f(\tilde{\mathbf{p}}). \tag{1.27}$$

3. *Starting from* $\tilde{\mathbf{p}}_o(t)$*, solve (1.24) via Newton's method:*

   (a) Reset iteration counter $s = 1$ *and set auxiliary variable* $\widehat{\mathbf{p}}(s) = \tilde{\mathbf{p}}_o(t)$.

   (b) **repeat:**

(c) Compute Gradient vector $\nabla\tilde{f}(\widehat{\mathbf{p}}(s))$ *and Hessian matrix* $\nabla^2\tilde{f}(\widehat{\mathbf{p}}(s))$*, see Appendix.*

(d) Compute Newton direction: $\mathbf{d} = -\left(\nabla^2\tilde{f}(\widehat{\mathbf{p}}(s))\right)^{-1}\nabla\tilde{f}(\widehat{\mathbf{p}}(s))$.

(e) Line search: Choose a step size $\nu = \arg\min_{\mu\in[01]}\tilde{f}(\widehat{\mathbf{p}}(s)+\mu\mathbf{d})$.

(f) $s = s+1$

(g) Update powers: $\widehat{\mathbf{p}}(s) = \widehat{\mathbf{p}}(s-1)+\nu\mathbf{d}$.

(h) **until:** *convergence of the cost* $\tilde{f}(\widehat{\mathbf{p}}(s))$ *(within* ϵ *- accuracy).*

4. Set $\tilde{\mathbf{p}}(t) = \widehat{\mathbf{p}}(s)$.

Notice that (1.25) and (1.27) simply evaluate the current objective function at previously computed solutions (for past slots). The parameter W should be large enough to capture emerging periodic behavior, but using larger than necessary W's is not a problem, as function evaluations are cheap relative to the Newton steps that follow.

Similar to the batch case, it is also possible to begin with a high-SINR approximation and successively refine it by tuning the parameters $\alpha_\ell(t)$ and $\beta_\ell(t)$ to tighten the individual link rate bounds. This yields the following algorithm.

Algorithm 4. Adaptive Successive Approximation (Adaptive S.A.)*:*

Initialize link rate bound parameters: set $\alpha_\ell(t) = 1$, $\beta_\ell(t) = 0$, $\forall\ell\in\mathcal{L}$

For each time slot $t \geq 1$*:*

1. Calculate the differential backlogs

2. Power initialization:

For $t = 1$ *draw random* $\tilde{\mathbf{p}}_o(t)$ *satisfying log-power constraints;*

else for $t\in[2,W]$ *set:*

$$\tilde{\mathbf{p}}_o(t) = \arg\max_{\tilde{\mathbf{p}}\in\{\tilde{\mathbf{p}}(1),\ldots,\tilde{\mathbf{p}}(t-1)\}} f(\tilde{\mathbf{p}}) \tag{1.28}$$

where

$$f(\tilde{\mathbf{p}}) := \sum_{\ell=1}^{L} D_\ell(t)\left(\alpha_\ell\left(\tilde{G}_{\ell\ell}+\tilde{p}_\ell-\log\left(\sum_{\substack{k=1\\k\neq\ell}}^{L} e^{\tilde{G}_{k\ell}+\tilde{p}_k}+e^{\tilde{V}_\ell}\right)\right)+\beta_\ell\right), \tag{1.29}$$

with α_ℓ, β_ℓ, $\forall\ell\in\mathcal{L}$, *updated at* $\gamma_\ell(\tilde{\boldsymbol{p}})$ *according to* (1.8), $\forall\ \tilde{\boldsymbol{p}}\in\{\tilde{\mathbf{p}}(1),\ldots,\tilde{\mathbf{p}}(t-1)\}$,

else $(t\geq W+1)$ *set:*

$$\tilde{\mathbf{p}}_o(t) = \arg\max_{\tilde{\mathbf{p}}\in\{\tilde{\mathbf{p}}(t-W),\ldots,\tilde{\mathbf{p}}(t-1)\}} f(\tilde{\mathbf{p}}). \tag{1.30}$$

3. *Outer initialization: Reset outer iteration counter* $s_1 = 1$, *set* $\tilde{\mathbf{p}}_1(t, s_1) = \tilde{\mathbf{p}}_o(t)$, *and pick* $\alpha_\ell(t, s_1)$, $\beta_\ell(t, s_1)$ *according to* (1.8) *for* $z_o = \gamma_\ell(\tilde{\mathbf{p}}_1(t, s_1))$, *see* (1.1), $\forall \ell \in \mathcal{L}$.

4. **repeat:**

5. *Starting from* $\tilde{\mathbf{p}}_o(t)$, *solve (1.24) via Newton's method:*

   (a) *Inner initialization: Reset inner iteration counter* $s_2 = 1$ *and set auxiliary variable* $\tilde{\mathbf{p}}_2(s_2) = \tilde{\mathbf{p}}_1(t, s_1)$.

   (b) **repeat:**

   (c) *Compute Gradient vector* $\nabla \tilde{f}(\tilde{\mathbf{p}}_2(s_2))$ *and Hessian matrix* $\nabla^2 \tilde{f}(\tilde{\mathbf{p}}_2(s_2))$, *see Appendix.*

   (d) *Compute Newton direction:* $\mathbf{d} = -\left(\nabla^2 \tilde{f}(\tilde{\mathbf{p}}_2(s_2))\right)^{-1} \nabla \tilde{f}(\tilde{\mathbf{p}}_2(s_2))$.

   (e) *Line search: Choose a step size* $\nu = \arg\min_{\mu \in [01]} \tilde{f}(\tilde{\mathbf{p}}_2(s_2) + \mu \mathbf{d})$.

   (f) $s_2 = s_2 + 1$

   (g) *Update powers:* $\tilde{\mathbf{p}}_2(s_2) = \tilde{\mathbf{p}}_2(s_2 - 1) + \nu \mathbf{d}$.

   (h) **until:** *convergence of the cost* $\tilde{f}(\tilde{\mathbf{p}}_2(s_2))$ *(within* ϵ *- accuracy).*

6. $s_1 = s_1 + 1$

7. *Set* $\tilde{\mathbf{p}}_1(t, s_1) = \tilde{\mathbf{p}}_2(s_2)$

8. *Tightening step: pick* $\alpha_\ell(t, s_1)$, $\beta_\ell(t, s_1)$ *according to* (1.8) *for* $z_o = \gamma_\ell(\tilde{\mathbf{p}}_1(t, s_1))$, *see* (1.1), $\forall \ell \in \mathcal{L}$.

9. **until** *convergence of the cost* $\tilde{f}(\tilde{\mathbf{p}}_1(t, s_1))$ *(within* ϵ*- accuracy).*

10. *Set* $\tilde{\mathbf{p}}(t) = \tilde{\mathbf{p}}_1(t, s_1)$.

1.6 Complexity of convex approximation

The worst-case complexity order of the batch algorithms is $O(L^{3.5})$, where L is the number of links (optimization variables) [5, 28]. In dense networks where every node is within range of every other node, $L = O(N^2)$, where N is the number of nodes; thus worst-case complexity is then $O(N^7)$. This is relatively high, but it is important to note that even the solution of a

system of linear equations in the link powers would entail complexity $O(L^3) = O(N^6)$. The successive approximation algorithm typically converges in just a few (3-4 in our experiments) tightening steps, so complexity order is the same as the high-SINR one.

The worst-case complexity of the adaptive approximation algorithms is the same as that of the batch algorithms - mainly due to the matrix inversion in the Newton step which is cubic in L, the remainder being the number of Newton steps needed to converge in the worst case. The constants that are hidden in the big O notation are of course far smaller for the adaptive algorithms, and so is their average complexity - due to their "reuse" of past solutions for warm re-start. This will be illustrated in the simulations section. It is also possible to use quasi-Newton methods such as BFGS to further reduce the average complexity of the adaptive algorithms.

1.7 Baselines

1.7.1 Assessing the quality of approximation

Given that the proposed convex approximation algorithms only find approximate solutions to the original NP-hard problem, we would like to develop means of assessing how far an approximate solution is from an optimal one. Returning to the original objective, we could use the inequality $\log(1+z) \leq \log(1+z_o) + \frac{z-z_o}{1+z_o}$ (follows from $\log(x) \leq x-1$), met with equality at z_o, to upper bound the individual terms of the objective function. This yields an upper bound, but its maximization is difficult, as it involves products of variables.

The classical way to obtain an upper bound is via duality - i.e., by considering the Lagrange dual problem. The dual problem is convex by definition; yet computing the dual function (objective of the dual problem) is also NP-hard. This can be established by showing that it contains (see Appendix) the corresponding computation for single-carrier sum-rate maximization in DSL systems, which is known to be NP-hard [30].

In [47], an algorithm called MAPEL is proposed, based on increasingly accurate approximation of the feasible SINR region to find an optimal solution to the weighted sum-rate maximization problem. When optimal solution is sought, NP-hardness implies that MAPEL's worst-case complexity is exponential. MAPEL can also be used for approximation, however our simulations indicated that it is too complex to be used even as a benchmark in our setting. For this reason,

we will resort to an algorithm of Yu and Lui [60], originally developed for spectrum balancing in DSL systems, which yields an approximate solution of the dual problem - hence an approximate upper bound of our objective. This algorithm is briefly reviewed next.

1.7.2 Iterative Spectrum Balancing Algorithm

Considering per-link power constraints, the dual objective function $g(\boldsymbol{\lambda})$, where $\boldsymbol{\lambda}$ denotes the vector of Lagrange dual variables, is the result of the unconstrained maximization:

$$g(\boldsymbol{\lambda}) = \max_{\{p_\ell\}_{\ell\in\mathcal{L}}} \sum_{\ell=1}^{L} D_\ell(t) \log\left(1 + \frac{G_{\ell\ell}p_\ell}{\frac{1}{G}\sum_{\substack{k=1\\k\neq\ell}}^{L} G_{k\ell}p_k + V_\ell}\right) + \boldsymbol{\lambda}^T(\mathbf{P}-\mathbf{p})$$

$$= \max_{\{p_\ell\}_{\ell\in\mathcal{L}}} \sum_{\ell=1}^{L} D_\ell(t) \log\left(1 + \frac{G_{\ell\ell}p_\ell}{\frac{1}{G}\sum_{\substack{k=1\\k\neq\ell}}^{L} G_{k\ell}p_k + V_\ell}\right) + \sum_{\ell=1}^{L}\lambda_\ell(P_\ell - p_\ell)$$

$$= \max_{\{p_\ell\}_{\ell\in\mathcal{L}}} \sum_{\ell=1}^{L} \left(D_\ell(t) \log\left(1 + \frac{G_{\ell\ell}p_\ell}{\frac{1}{G}\sum_{\substack{k=1\\k\neq\ell}}^{L} G_{k\ell}p_k + V_\ell}\right) - \lambda_\ell p_\ell \right) + \sum_{\ell=1}^{L}\lambda_\ell P_\ell.$$

In [60], an iterative algorithm that alternates between primal (link powers) and dual variables is proposed to obtain an approximate solution to the dual problem. Following straightforward adaptation in our present (single-carrier, weighted sum rate) context, this algorithm can be summarized as follows:

Algorithm 5. Iterative Spectrum Balancing (ISB)*:*

For each time slot t:

1. *Initialize* $\boldsymbol{\lambda} = [\lambda_1 ... \lambda_L]^T$
2. **repeat:**
3. *initialize* $\mathbf{p} = [p_1 ... p_L]^T$
4. **repeat:**
5. *for* $\ell = 1$ *to* L*, set*

$$p_\ell = arg\max_{p_\ell} \sum_{m=1}^{L} \left(D_m(t) \log\left(1 + \frac{G_{mm}p_m}{\frac{1}{G}\sum_{\substack{k=1\\k\neq m}}^{L} G_{km}p_k + V_m}\right) - \lambda_m p_m \right)$$

end

6. **until** $\mathbf{p} = [p_1 ... p_L]^T$ *converges (within* ϵ *- accuracy)*

7. *Update* $\boldsymbol{\lambda} = [\lambda_1 ... \lambda_L]^T$ *using subgradient or ellipsoid method*

8. **until** $\boldsymbol{\lambda} = [\lambda_1 ... \lambda_L]^T$ *converges (within* ϵ *- accuracy).*

In the above, the element-wise power optimization is nonconvex in p_ℓ, but can be accomplished via either line-search, or polynomial rooting (finding the roots of the first derivative and examining the second derivative as well). A sub-gradient iteration is employed for the $\boldsymbol{\lambda}$-update step (7),

$$\boldsymbol{\lambda}^{(m+1)} = \max \ (\boldsymbol{\lambda}^{(m)} - s^{(m)}(\mathbf{P} - \mathbf{p}), \mathbf{0})$$

where m is the iteration number, and the step-size sequence is usually taken as $s^{(m)} = \frac{\beta}{m}$, for some $0 < \beta \leq 1$.

1.7.3 Prior art: Back-Pressure Best Response Algorithm

In addition to a (possibly unattainable, by often tight) upper bound, we will also compare the proposed algorithms to the earlier state-of-art for the BPPC problem - namely [15], where two low-complexity algorithms were derived under the high-SINR assumption. These are the *Best Response* algorithm, and the *Gradient Projection* algorithm. Best Response outperforms Gradient Projection; we therefore only consider the former in the sequel. For every time slot $t = 1, 2, \ldots$, and for each link $\ell = 1, \ldots, L$, the Best Response algorithm computes the differential backlog $D_\ell(t)$ and the *interference price* $\pi_\ell(t) = \frac{D_\ell(t)}{I_\ell(\mathbf{p}(t)) + V_\ell}$, where $I_\ell(\mathbf{p}(t) := \frac{1}{G} \sum_{\substack{k=1 \\ k \neq \ell}}^{L} G_{k\ell} p_k(t)$ is the total interference to link ℓ from other links. The price is subsequently communicated to all links, and link ℓ transmits with power $p_\ell(t) = \min \left(\frac{D_\ell(t-1)}{\sum_{\substack{k=1 \\ k \neq \ell}}^{L} \pi_k(t-1) \frac{1}{G} G_{\ell k}}, \ P_\ell^{max} \right)$.

1.8 Simulation Results

Simulation experiments have been performed in three different scenarios, in order to assess the performance of the various algorithms. In addition to the batch high-SINR / S.A. and their adaptive counterparts developed herein, we also included ISB and the Best Response algorithm as baselines for comparison.

• Scenario 1 - small network with moderate interference: The first scenario considered is a network with $N = 6$ nodes, randomly drawn from a uniform distribution over a 100×100 square. The

resulting topology is shown in Fig. 1.2. The network remains fixed throughout the simulations for scenario 1. The leftmost node is the source, the rightmost is the destination, and the intermediates are relays. Since the destination acts as a sink, and no node transmits to itself, there are $L = 25$ links in the network (note that intermediate nodes are in principle allowed to send packets back to the source). Direct and crosstalk link power losses are taken inversely proportional to d^4, where d is the distance between transmitter and receiver. It is assumed that nodes cannot listen while they talk: if link ℓ terminates at node i and link k departs from i, then $G_{k,\ell} = 1/\text{eps}$, where eps is machine precision. A spreading gain $G = 128$ is assumed and absorbed in the crosstalk factors $G_{k\ell}$, $k \neq \ell$. The background noise power is set to $V_\ell = 10^{-12}$, $\forall \ell$. Per-link power constraints are adopted to be consistent with ISB and Best Response, used here as baselines; $P_\ell^{max} = 5$, $\forall \ell$. We assume deterministic arrivals at the source. Simulating the network under control of each algorithm, we consider various arrival rates in order to narrow down the maximal arrival rate that each algorithm can support. Unless otherwise noted, the network is simulated for 100 packet / control slots. For the batch algorithms, each problem instance is solved using the CVX toolbox [17] for Matlab, on a per-slot basis. The desired accuracy in step 6 of the batch S.A. algorithm was set to $\epsilon = 0.1$.

Simulations verified the expected push-pull 'wave' propagation over the network, and that periodic behavior emerges for stable setups. For arrival rates up to 9 packets per slot (pps), both batch algorithms stabilize the system, and the average throughput equals the incoming traffic at the source. Gradually increasing the arrival rate, we found that 9.7 pps was the maximal value for which the batch high-SINR algorithm managed to stabilize the system. Beyond that, all backlogs grow to infinity. Plots of the evolution of source and relay backlogs, power allocation, and end-to-end throughput for the batch high-SINR algorithm are shown in Fig. 1.3 (left) for arrival rate 9.7 pps, and Fig. 1.3 (right) for arrival rate 9.9 pps. Note that for the stable setup of Fig. 1.3 (left), and after a short transient, all relevant quantities (backlogs, powers, end-to-end throughput) converge to a periodic pattern. In this case, the emerging period is two slots. Simulations also verified that the batch S.A. algorithm is able to stabilize the system at higher input loads than the batch high-SINR algorithm. The maximum arrival rate that batch S.A. can support in this case is 10.4 pps. The queues become unstable at higher arrival rates. This can be easily visualized at 10.8 pps. Plots for the batch S.A. are shown in Fig. 1.4 (left) for arrival rate 10.4 pps, and Fig. 1.4 (right) for arrival rate 10.8 pps. For the stable setup in Fig.

1.4 (left), note again that all relevant quantities quickly settle in to a periodic pattern.

A first check point for the custom adaptive algorithms is to verify that they indeed reproduce the results of their batch counterparts, ideally at far lower complexity. We used a window of $W = 5$ slots for power initialization, and τ in (1.24) was set to 10^6. For the line search in step $3e$ of the adaptive high-SINR ($5e$ of the adaptive S.A., respectively), a grid search of accuracy 10^{-3} was used. The desired accuracy in step 9 of the adaptive S.A. was set to $\epsilon = 0.1$. Simulation results concerning the adaptive high-SINR algorithm are shown in Fig. 1.5, for arrival rates 9.7 and 9.9 pps, passing sanity check (compare to Fig. 1.3). The adaptive S.A. algorithm likewise keeps up with its batch counterpart, as shown in Fig. 1.6 (compare to Fig. 1.4). Notice that the adaptive version of each algorithm yields qualitatively similar results (and in particular attains the same stable throughput) as its batch counterpart, however the respective power allocations do not coincide. This is because the underlying problem does not have a unique solution in general. To see this, consider an interference-limited scenario comprising two intermediate nodes transmitting to the common destination. Assuming symmetric loads and channels, activating either one of the two will yield the same reward.

As an indication of the complexity of the various algorithms, we note that the batch high-SINR one required on average about 8 seconds per slot (problem instance); batch S.A. averaged $3 - 4$ outer iterations for a total of 20 seconds per slot. The adaptive high-SINR algorithm averaged about 4 inner iterations (Newton steps) for a total of 1.5 seconds per slot. Adaptive S.A. averaged $1 - 2$ outer iterations of up to 4 inner Newton steps, for a total of about 1.5 seconds per slot, when initialized at the best (for the present slot) of the W past solutions for the previous slots. The difference is more significant (order of magnitude) in the case of the successive approximation algorithms, and the gap widens quickly with the size of the network. We also note that for time-invariant (or slowly-varying) channels, the rate of power updates (i.e., how often one solves the BPPC problem) only affects delay, not the attainable stable throughput of the algorithm, as shown in [53, 54]. Thus one can take more time between power updates at the expense of increased packet delay, without sacrificing throughput.

Next, we present results for the Best Response algorithm proposed in [15]. We also tried the Gradient Projection algorithm in [15], but it proved consistently inferior[4] to Best Response, so we skip the associated plots. Best Response was initialized with the solution of the Batch

[4]Failed to stabilize the network in cases where Best Response did.

high-SINR algorithm, to give it the best possible warm start under the high-SINR assumption. However, it still takes far longer to reach steady-state (when it can stabilize the system) than previously discussed algorithms, hence we simulate it for 1000 slots in the sequel. Experiments showed that the highest arrival rate that Best Response can handle is 5.7 pps. Plots of the evolution of source and relay backlogs, power allocation, and end-to-end throughput, for the arrival rate 5.7 and 5.8 pps, are shown in Fig. 1.7. Obviously, this algorithm leaves much to be desired in terms of maximum stable throughput, delay (cf. queue backlogs and Little's theorem), and settling time relative to batch and adaptive high-SINR and S.A. On the other hand, Best Response is very cheap in terms of computation. As an indication, its average run-time here was 0.0085 seconds per slot, which is 2 orders of magnitude less than the best-performing algorithms.

Recall that computing the dual function is NP-hard, and ISB is only able to provide an *approximate* upper bound to the maximum attainable objective of the primal problem. Due to approximation, the power allocation computed by ISB need not (and in fact does not, in many cases) yield a higher primal objective than batch/adaptive S.A. Still, it is reassuring to see that the approximate upper bound derived from ISB is indeed higher and not very far from the value of the objective computed from batch/adaptive S.A. Illustrating this gap is the objective of the next two plots. We consider two arrival rates, one well-within the stable region and another on the margin: 8, and 10.4 pps. The network evolves under control of the batch S.A. algorithm, and ISB is used to (approximately) upper bound the attainable differential backlog weighted sum of link rates for each slot. The link power optimization in step 5 of the ISB algorithm is performed using a grid search over $[0\ P_\ell]$ of accuracy 10^{-4}. For each slot t, all elements of $\boldsymbol{\lambda}$ are initially set to 1, and the update of $\boldsymbol{\lambda}$ in step 7 is performed using the sub-gradient method with $\beta = 0.1$. The desired accuracy for the convergence of $\mathbf{p}$ and $\boldsymbol{\lambda}$ was set to $\epsilon = 10^{-2}$. The final objective values attained by ISB and the batch S.A. algorithm are plotted together as functions of t in Fig. 1.8, in separate panels for the two arrival rates considered. Notice that ISB consistently hovers above batch S.A. in both cases considered, which is satisfying.

We next consider a tracking experiment to illustrate the ability of the adaptive S.A. algorithm to follow changes in the operational environment. Packets are now injected not only at the original source node (node 1), but also at an intermediate relay (node 3). There is still one destination, and all packets are treated the same way - there is only one buffer per node. For the first 50 slots, traffic is injected at 3 pps through node 1, and at 6 pps through node 3. For

the next 50 slots the injection pattern reverses: 6 pps through node 1, and 3 pps through node 3. Fig. 1.9 shows simulation results for the adaptive S.A. algorithm, which evidently responds swiftly to the change in the traffic pattern. The Best Response algorithm cannot stabilize the network for this pair of rates; but even at lower, sustainable rates, its response to the change of the injection pattern was very slow and hard to discern. We skip associated plots for brevity.

• Scenario 2 - small network, more interference: We next consider a scenario that is more interference-limited (less power-limited) than before: using $G = 8$ instead of 128. ISB takes disproportional time to converge in this scenario, thus we drop it from consideration. Due to stronger interference, the high-SINR algorithms can now stabilize the system for arrival rates only up to 2.4 pps. Fig. 1.10 plots results for adaptive high-SINR for arrival rate 2.4 pps (left) and 2.6 pps (right). The S.A. algorithms proved much better, supporting *three times higher* stable throughput (7.5 pps). It turned out that, in order to do so, they both converged to the same solution: keeping only the direct link from source to destination, always on and at maximum power. Simulation results are shown in Fig. 1.11 for arrival rate 7.5 pps (left), and 7.9 pps (right). In fact, arrival rates up to 7.8 pps are sustainable, but because of the long transient of the source backlog, we depict, for ease of visualization, the results for the stable rate of 7.5 pps. The complexity advantage of the adaptive algorithms relative to the batch ones remains similar to scenario 1. Best Response managed to stabilize the system for rates up to 2.1 units per slot. The respective plots are shown in Fig. 1.12 for the 2.1 pps (left) and 2.3 pps (right).

• Scenario 3 - larger network, moderate interference: A larger network comprising $N = 12$ nodes ($L = 121$ links) is considered in our third scenario. The setup is otherwise identical to scenario 1. Simulation showed that the high-SINR algorithms managed to stabilize the network for arrival rates up to 12.6 pps. The adaptive S.A., on the other hand, managed to stabilize the system for up to 15.7 pps. Unlike adaptive S.A., the batch S.A. algorithm was too slow to include in this comparison. The maximum rate that Best Response could support was 4.4 pps. We skip associated plots for brevity, and instead gather all results concerning attainable rates in Table 1.1.

Table 1.1: Attainable stable arrival rates in packets per slot.

Scen.	Batch high-SINR	Adapt. high-SINR	Batch S.A.	Adapt. S.A.	Best Response
Scen. 1	9.7	9.7	10.4	10.4	5.7
Scen. 2	2.4	2.4	7.5	7.5	2.1
Scen. 3	12.6	12.6	N/A	15.7	4.4

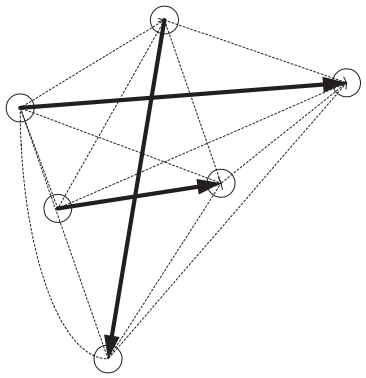

Figure 1.1: Illustration of network construction in proof of Claim 1. Thick lines indicate links with nonzero differential backlog.

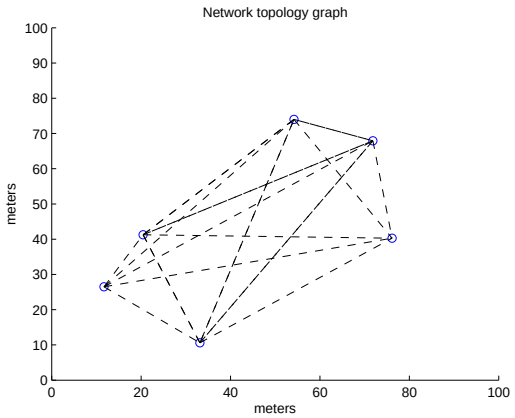

Figure 1.2: Network topology

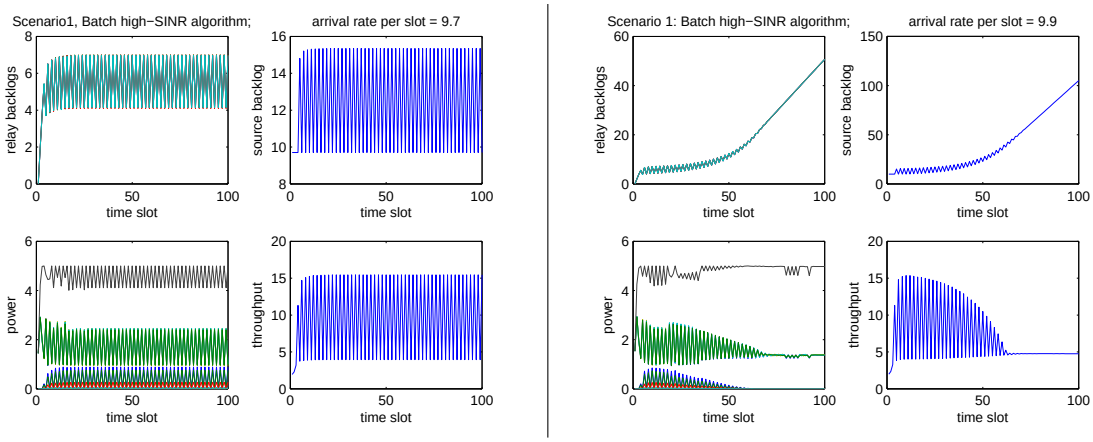

Figure 1.3: Scenario 1: Batch high-SINR, arrival rate = 9.7 (left) and 9.9 (right).

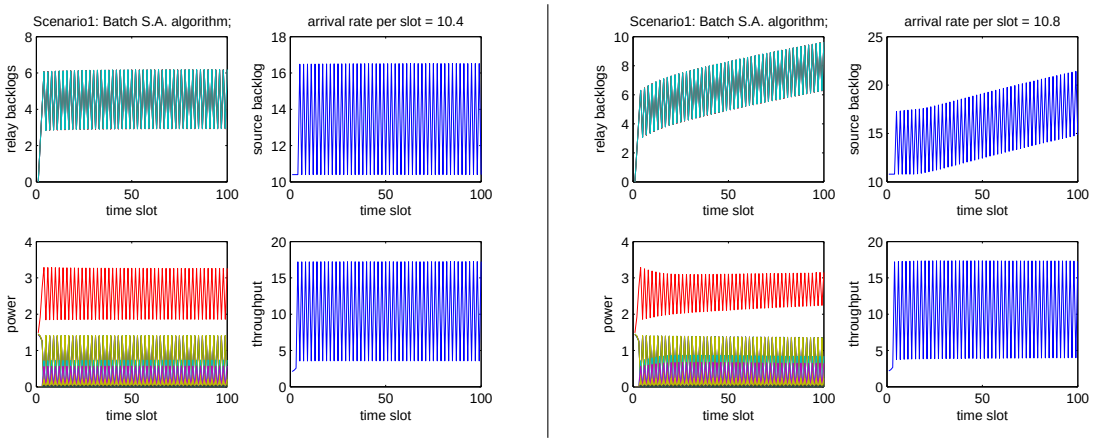

Figure 1.4: Scenario 1: Batch Successive Approximation, arrival rate = 10.4 (left) and 10.8 (right).

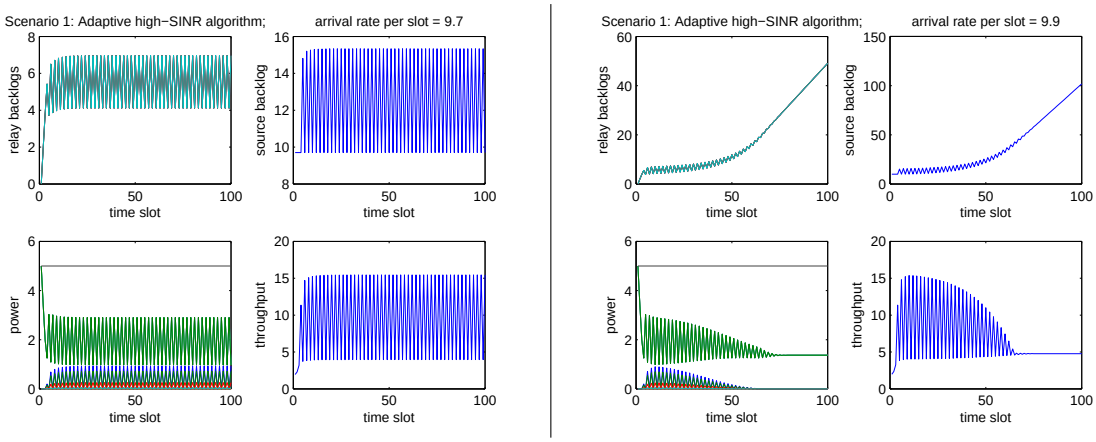

Figure 1.5: Scenario 1: Adaptive high-SINR, arrival rate = 9.7 (left) and 9.9 (right).

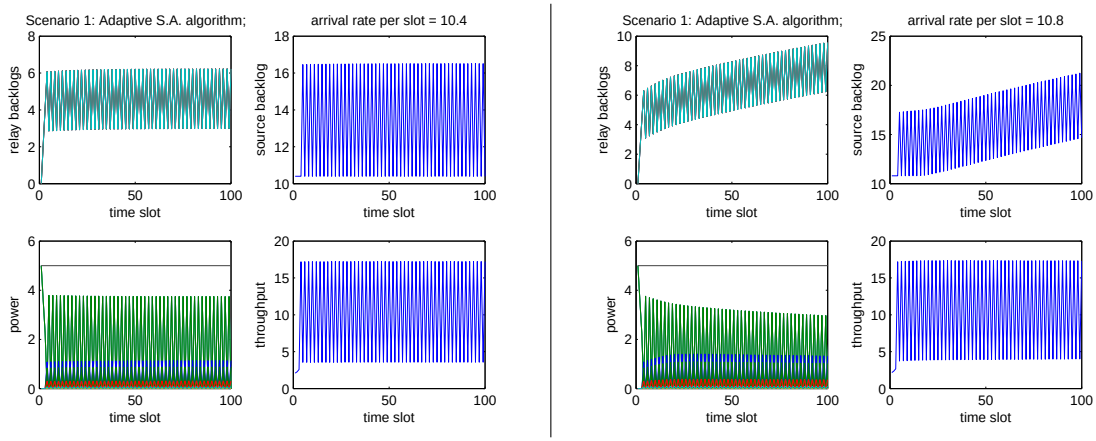

Figure 1.6: Scenario 1: Adaptive Successive Approximation, arrival rate = 10.4 (left) and 10.8 (right).

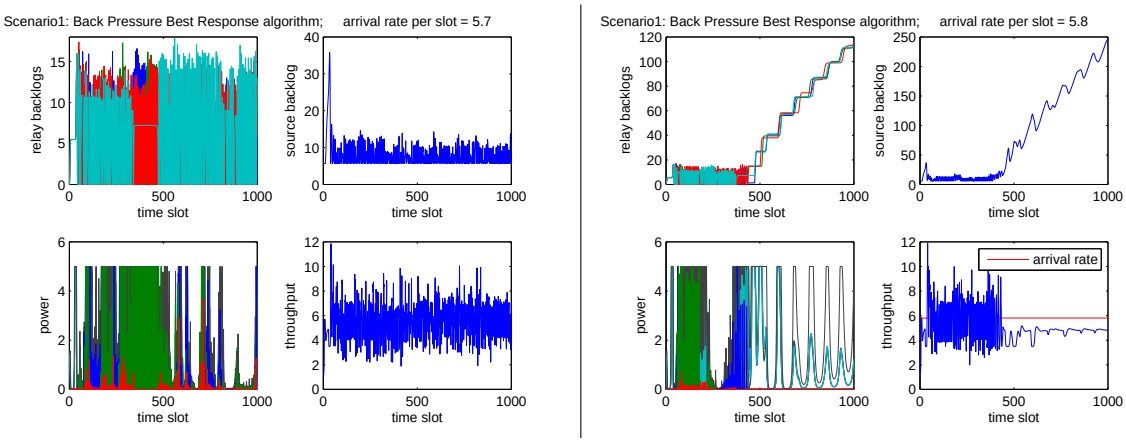

Figure 1.7: Scenario 1: Back Pressure Best Response algorithm, arrival rate = 5.7 (left) and 5.8 (right).

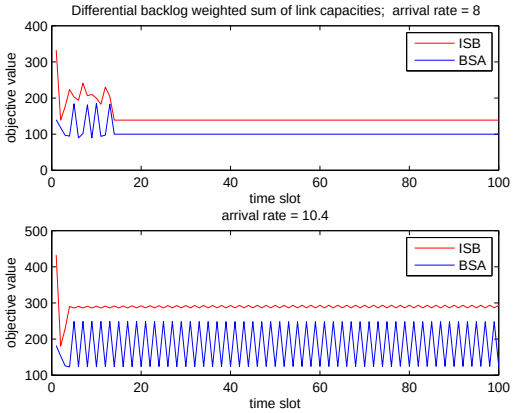

Figure 1.8: Scenario 1: ISB vs Batch S.A.: comparison between objective values attained.

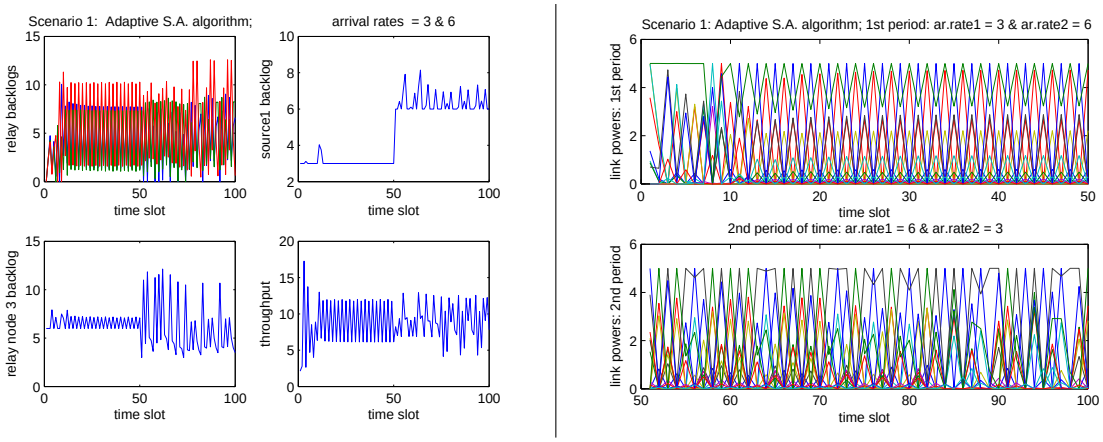

Figure 1.9: Scenario 1: Switching injection pattern: Adaptive S.A., backlogs / throughput(left), link powers (right).

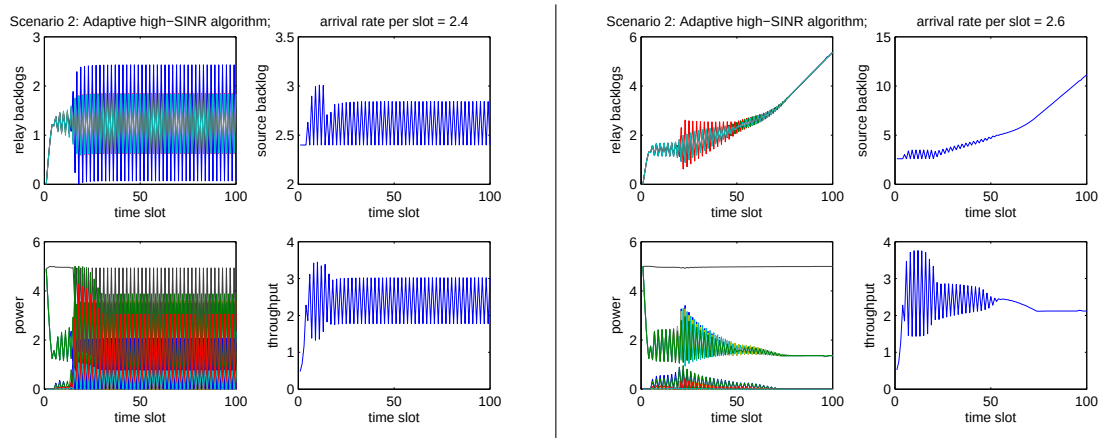

Figure 1.10: Scenario 2: Adaptive high-SINR, arrival rate = 2.4 (left) and 2.6 (right).

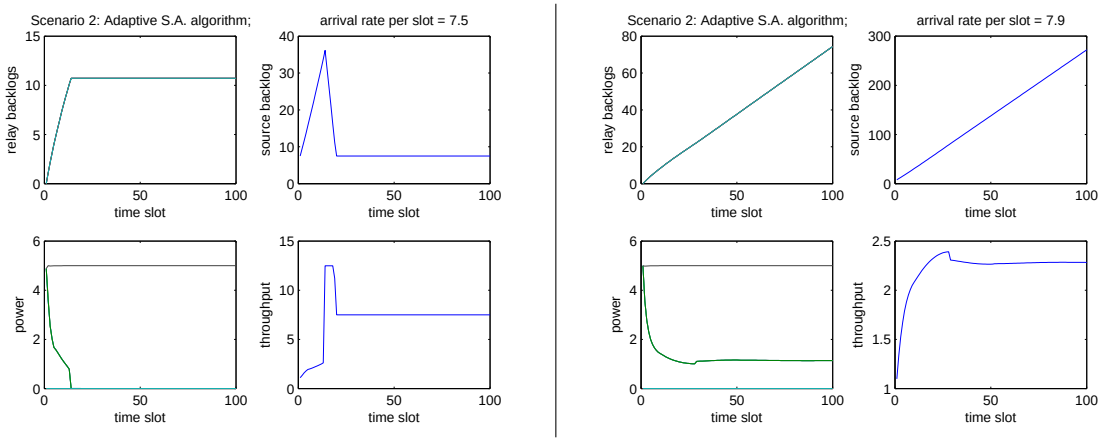

Figure 1.11: Scenario 2: Adaptive Successive Approximation, arrival rate = 7.5 (left) and 7.9 (right).

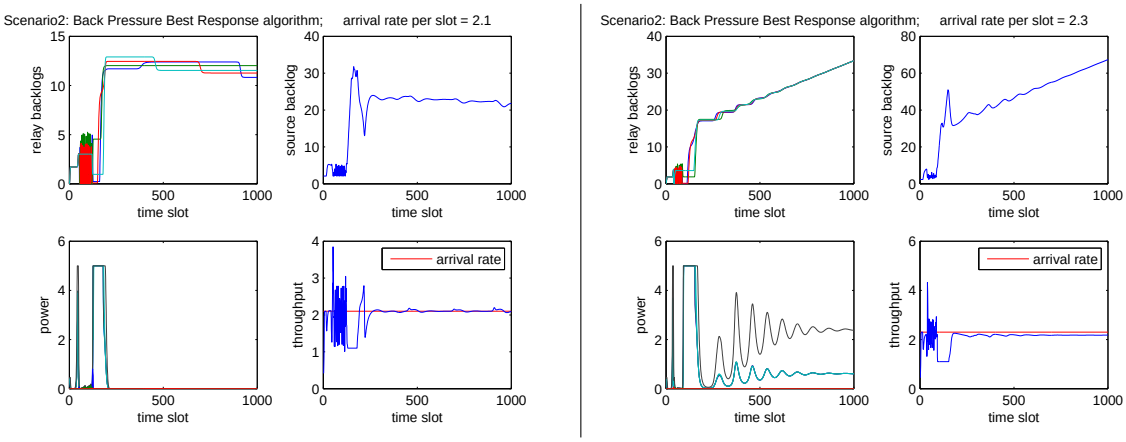

Figure 1.12: Scenario 2: Back Pressure Best Response algorithm, arrival rate = 2.1 (left) and 2.3 (right).

1.9 Conclusions

We have considered the power control problem in wireless multi-hop networks. From the viewpoint of maximizing stable end-to-end throughput, the objective is to maximize a differential backlog-weighted sum of link rates, subject to power constraints. Following physical layer optimization at the beginning of each transmission round, back-pressure routing is used for packet forwarding over the network. This two-step approach is optimal from a throughput perspective, and, for this reason, the back-pressure power control (BPPC) problem is central in wireless multi-hop networks.

BPPC was known to be non-convex [15]; we established that it is in fact NP-hard. This precludes optimal solution at worst-case polynomial complexity, and motivates the pursuit of appropriate approximation algorithms. Drawing from related problems in the DSL literature (in particular the SCALE algorithm of [45, 46]), we proposed two ways to approximate the BPPC problem which far outperform the previous state of art in [15]. Most importantly, recognizing the high computational burden arising from the need to solve BPPC on a per-slot (transmission round) basis, and capitalizing on quasi-periodicity of the power allocation in stable setups due to the push-pull nature of the solution, we proposed two custom adaptive approximation algorithms that offer excellent throughput performance at reasonable computational complexity, which remains worst-case polynomial. In addition to throughput margin, the proposed algorithms feature shorter backlogs / queueing delays, and faster transient response.

An interesting research direction is to consider means of implementing the proposed algorithms - adaptive S.A. in particular - in a distributed fashion. Recent progress in distributing the Newton step using Gaussian belief propagation [12], and related work in distributed network utility maximization [19] may be useful towards this end.

Another interesting direction is to consider time-varying channels. If the channel variation is far slower than the rate of power re-optimization, and the channels can be tracked at the central scheduler, then our methods remain operational. If the channels change (perhaps abruptly) only at certain points in time, but otherwise remain fixed between such changes and known to the central scheduler, and the BPPC problem is *exactly* solved before each such 'dwell', then throughput optimality still holds [53]. This scenario is plausible when several different networks are time-division multiplexed (similar to multiplexing ALOHA protocols, for example). We have

shown that exact solution of BPPC is NP-hard; but we have also shown that our approximate solutions deliver substantial throughput improvement relative to the prior art in [15]. Still, much work is needed to figure out good policies for general time-varying scenarios, including the interplay with distributed implementation.

1.10 Appendix

Extension to multiple flows: In the case of multiple flows, each node is assumed to maintain a separate queue for each flow (this is easy to implement and it does not require additional storage). Let there be F flows (i.e., F destinations) in the network. For every scheduling time slot $t \in \{1, 2, \cdots\}$, the following algorithm is executed.

Let $W_{i,f}(t)$ denote the queue length of node i for flow f at the end of slot t. The differential backlog of link $\ell = (i, j)$ for flow f at the end of slot t is defined as

$$D_{\ell,f}(t) := \begin{cases} \max\{0, W_{i,f}(t) - W_{j,f}(t)\}, & j \neq \text{dest}(f) \\ W_{i,f}(t), & j = \text{dest}(f) \end{cases}$$

where $\text{dest}(f)$ is the destination node for flow f. Let

$$\bar{D}_\ell(t) := \max_{f \in \{1,\cdots,F\}} D_{\ell,f}(t)$$

be the maximum differential backlog for link ℓ at the end of slot t, and

$$\bar{f}_\ell(t) := \arg \max_{f \in \{1,\cdots,F\}} D_{\ell,f}(t)$$

be a flow that attains maximum differential backlog for link ℓ at the end of slot t. Then link ℓ is *dedicated* to flow $\bar{f}_\ell(t)$ for the next scheduling period (if $\bar{D}_\ell(t) = 0$, then link ℓ remains idle), and transmission powers are set according to

$$\max_{\{p_\ell\}_{\ell \in \mathcal{L}}} \sum_{\ell=1}^{L} \bar{D}_\ell(t) c_\ell$$

$$\text{subject to} \quad 0 \leq \sum_{\ell : \text{Tx}(\ell) = i} p_\ell \leq P_i, \quad i \in \mathcal{N}.$$

That is, the same type of BPPC problem is solved as in the single-flow case, but this time using $\bar{D}_\ell(t)$'s in place of $D_\ell(t)$'s. The other difference with the case of a single flow is that now a decision has to be made as to which flow(s) will be routed on any given link and how to multiplex them - but it turns out that simple winner-takes-all routing combined with BPPC is in fact optimal from a throughput perspective, as shown in [54].

Proof. (Claim 1) We will show that the problem contains that of determining the size of the maximum independent set in a graph, which is NP-hard. The proof draws heavily from the proof of Theorem 1 in [29], which deals with the multiuser sum-rate maximization problem for

the interference channel. Back-pressure power control looks like the same problem, but there is a catch: the $G_{k,\ell}$ parameters are completely free in the multiuser sum-rate maximization problem in [29], but in back-pressure power control the $G_{k,\ell}$'s are subject to certain restrictions. Consider the case of two links k, ℓ stemming from the same node. Clearly, $G_{k,\ell} = G_{\ell,\ell}$, and $G_{\ell,k} = G_{k,k}$. Likewise, consider two links k, ℓ with common receiving node. Then $G_{k,k} = G_{k,\ell}$, and $G_{\ell,\ell} = G_{\ell,k}$. This seems to suggest that back-pressure power control is a *restriction* of the multiuser sum-rate maximization problem, and restriction of an NP-hard problem is not necessarily NP-hard. Another important difference is that back-pressure power control may be subject to per-node (instead of per-link) power constraints, which couple the transmission power across multiple links.

The key to a simple and clear proof is *backlog reduction*: realizing that there is freedom to choose the backlogs in such a way that we peel off these complicating factors to reveal the multiuser sum-rate maximization problem as a special case.

The *conflict graph* is a familiar concept in network scheduling. Each *directed link* in the network corresponds to a node in the conflict graph. Nodes k, ℓ in the conflict graph are connected by an *undirected edge* if $G_{k,\ell} \neq 0$, or $G_{\ell,k} \neq 0$, or both - i.e., when links k and ℓ can interfere with each other. An *independent set* of nodes (not connected by an edge) in the conflict graph corresponds to a set of network links that can be simultaneously activated without causing interference to one another. Given an undirected connected graph $\Gamma = (V, E)$ with $|V| = L$ nodes, construct an instance of a wireless multi-hop network whose conflict graph is Γ as follows:

- Choose $2L$ network nodes, split them in L pairs, draw a directed link between each pair, and set the differential backlogs of these links to 1. Set the differential backlogs of all remaining links to 0. The L drawn links are the only ones that can be activated in the next slot. This is important because it effectively reduces the network to a set of co-channel links that do not share transmitters or receivers; see Fig. 1.1. Since no transmitter is shared, there is no distinction between per-link or per-node power constraints in so far as this proof is concerned.

- For the links with non-zero differential backlog, set

  - $G_{\ell,\ell} = 1$, $P_\ell = 1$, $G = 1$ (no spreading), and $V_\ell = M$, where $M > L$ is a constant;

– For every edge in E connecting nodes k and ℓ, set $G_{k,\ell} = G_{\ell,k} = ML^2$ (notice that we enforce symmetry, which will be used later in the proof).

Notice that the choice of the coupling factors in the above has implications for other links in the network: For example, strong crosstalk between two links implies that there is a strong direct channel gain between the transmitter of one link and the receiver of the other. If the two transmitters could switch receivers they would set up more favorable links. This possibility is excluded, however, by our selection of differential backlogs: all except the L chosen links have zero differential backlog.

Let $\mathbf{p}^* \in [0,1]^{L\times 1}$ be an optimum solution of the back-pressure power control problem for this network instance, and let r^* be the corresponding optimal value of the (sum-rate) objective. Let I be a maximum independent set of Γ. By activating only the network links corresponding to conflict graph nodes in I, we obtain sum-rate $|I|\log(1+\frac{1}{M})$, hence $r^* \geq |I|\log(1+\frac{1}{M})$. Conversely, consider $\mathbf{p}^*$ and split it in two parts: those elements that are positive, and the rest that are zero. It has been shown in [29] that, under our working assumptions (in particular, $M > L$) the sum-rate objective is convex-U with respect to each component of $\mathbf{p}$, albeit not jointly convex in $\mathbf{p}$ as a whole. Since the maximum of a convex function over a polytope can always be attained at a vertex, and the constraint is $\mathbf{p}^* \in [0,1]^{L\times 1}$, it follows that we may assume, without loss of generality, that $\mathbf{p}^* \in \{0,1\}^{L\times 1}$. This is important, because it implies that if two or more interfering links are simultaneously active, the interference level will be lower bounded by ML^2. Let $A^* := \{\ell \mid \mathbf{p}^*(\ell) = 1\}$, and let J be a maximum independent subset of A^* in Γ. Clearly, $|J| \leq |I|$. We have:

$$r^* \leq |J|\log(1+\frac{1}{M}) + (|A^*| - |J|)\log(1+\frac{1}{M+ML^2}),$$

since $\log(1+\frac{1}{M})$ is an upper bound on rate for each link in J, in the best-case scenario that no link in $A^* - J$ interferes with it; and links outside J must have at least one interferer in J, for otherwise J is not a maximum independent subset of A^* in Γ. We can further bound r^* as follows:

$$r^* \leq |J|\log(1+\frac{1}{M}) + L\log(1+\frac{1}{M+ML^2}).$$

The term $L\log(1+\frac{1}{M+ML^2})$ is decreasing in L, and equal to $\log(1+\frac{1}{2M}) < \log(1+\frac{1}{M})$ for $L = 1$; it follows that $r^* < (|J|+1)\log(1+\frac{1}{M})$, and since $|J| \leq |I|$, we have $r^* < (|I|+1)\log(1+\frac{1}{M})$.

Putting everything together,

$$|I|\log(1+\frac{1}{M}) \leq r^* < (|I|+1)\log(1+\frac{1}{M}).$$

and solving for $|I|$, we obtain

$$|I| \in \left(\frac{r^*}{\log(1+\frac{1}{M})} - 1, \frac{r^*}{\log(1+\frac{1}{M})}\right].$$

This determines the exact (integer) value of I. It follows that if we could efficiently solve the back-pressure power control problem in polynomial time, we would be in position to determine the size of the maximum independent set in an arbitrary graph in polynomial time. ∎

Gradient and Hessian computation: The first and second order derivatives of

$$\tilde{f} = -\tau\left(\sum_{\ell=1}^{L} D_\ell(t)\left(a_\ell(t)\left(\tilde{G}_{\ell\ell} + \tilde{p}_\ell - \log\left(\sum_{\substack{k=1\\k\neq\ell}}^{L} e^{\tilde{G}_{k\ell}+\tilde{p}_k} + e^{\tilde{V}_\ell}\right)\right) + \beta_\ell(t)\right)\right) - \sum_{\ell=1}^{L}\log\left(\tilde{P}_\ell - \tilde{p}_\ell\right)$$

which are needed to compute the gradient and Hessian for the adaptive algorithms, are given by

$$\frac{\partial\tilde{f}}{\partial\tilde{p}_n} = -\tau D_n(t)a_n(t) + \frac{1}{\tilde{P}_n - \tilde{p}_n} + \tau\sum_{\substack{\ell=1\\\ell\neq n}}^{L}\frac{D_\ell(t)a_\ell(t)e^{\tilde{G}_{n\ell}+\tilde{p}_n}}{I_\ell},\ \forall n\in\mathcal{L}.$$

where

$$I_\ell = \sum_{\substack{k=1\\k\neq\ell}}^{L} e^{\tilde{G}_{k\ell}+\tilde{p}_k} + e^{\tilde{V}_\ell}$$

is the total interference to link ℓ from all other links. The second-order partial derivatives are

$$\frac{\partial^2\tilde{f}}{\partial\tilde{p}_n^2} = \frac{1}{(\tilde{P}_n - \tilde{p}_n)^2} + \tau\sum_{\substack{\ell=1\\\ell\neq n}}^{L}\frac{D_\ell(t)a_\ell(t)e^{\tilde{G}_{n\ell}+\tilde{p}_n}(I_\ell - e^{\tilde{G}_{n\ell}+\tilde{p}_n})}{I_\ell^2}$$

and

$$\frac{\partial^2\tilde{f}}{\partial\tilde{p}_n\partial\tilde{p}_m} = -\tau\sum_{\substack{\ell=1\\\ell\neq m,n}}^{L}\frac{D_\ell(t)a_\ell(t)e^{\tilde{G}_{n\ell}+\tilde{p}_n}e^{\tilde{G}_{m\ell}+\tilde{p}_m}}{I_\ell^2},\ \forall m,n\in\mathcal{L}.$$

Chapter 2

Distributed Back-Pressure Power Control for Wireless Multi-hop Networks

A key problem in wireless networking is how to choose a link activation schedule and associated powers in concert with routing decisions to optimize throughput. Back-pressure control policies are optimal in this context, but the underlying power control problem is non-convex. Back-pressure power control (BPPC) was recently shown to be NP-hard, yet amenable to successive convex approximation strategies that deliver manifold improvements in end-to-end throughput relative to the prior art in wireless networking. A drawback is that existing implementations are centralized, whereas practical power control has to be distributed across the network. The work presented here fills this gap by developing distributed implementations of approximations of the BPPC problem. Two approximation approaches to the NP-hard underlying problem are utilized, while feedback requirements and consensus-on-termination issues are also addressed, in order to come up with fully decentralized protocols that allow tight approximation of the BPPC objective in all interference regimes. The first distributed algorithm developed is based on the successive convex approximation approach and the Alternating Direction Method of Multipliers is utilized towards distributed implementation of the core step of the algorithm, which is the convex lower-bounding approximation of BPPC at any operating point. Judicious simulations show that the distributed algorithm matches the performance of its centralized counterpart. The iteratively

re-weighted Minimum Mean Square Error approach to weighted sum-rate maximization for the MIMO interference channel can be also exploited for our purpose. The WMMSE algorithm is specialized in our context and a second distributed implementation is then possible, providing an one-shot approximate solution to the BPPC problem. Further exploiting quasi-periodicity of the solution arising in stable setups, due to the push-pull evolution of the network in our context, and upon elaboration on distributed warm start, we derive custom adaptive versions of the distributed algorithms, that enjoy low average complexity with no performance loss. Extensive simulation experiments illustrate the merits of the proposed algorithms, and allow comparison of the two approximation approaches to the BPPC objective.

2.1 Introduction

Back-pressure routing is well-appreciated for its throughput optimality and conceptual simplicity, since its introduction in the early 90's [53–55]. More recently (e.g., see [14] and references therein) it has attracted considerable interest in the context of cross-layer wireless networking. We consider the back-pressure power control problem for maximal end-to-end throughput in a wireless multi-hop network. At the physical layer, for each scheduling slot, the core back-pressure power control (BPPC) problem amounts to maximizing a differential backlog-weighted sum of link rates. This was recently shown to be NP-hard in [32, 34], which also explored effective successive convex approximation strategies. A drawback is that the solution in [32, 34] is centralized. The contribution here is the distributed implementation of the successive convex approximation approach in [32, 34], using the Alternating Direction Method of Multipliers (ADMoM) [2–4]. Towards this end, the core step is distributed implementation of the convex lower-bounding approximation of BPPC at any operating point.

Distributed approximation of the weighted sum-rate maximization problem has been considered in [9], in the high-SINR (Signal to Interference plus Noise Ratio) regime and using a dual decomposition approach [9, 43, 44]. Instead of employing dual decomposition as in [9], here we use an Alternating Direction Method of Multipliers (ADMoM) approach, in light of its favorable convergence properties [2–4]. Further employing a consensus-on-the-min algorithm [10, 49] to reach agreement among links regarding the termination of iterations, we come up with a fully decentralized and promising, performance-wise, algorithm for the core step of the succes-

sive convex approximation of BPPC. A conference version of this work appears in *Proc. IEEE ICASSP 2012* [33]. A variation of this algorithm is also developed, less dependent on the initial choice of the *penalty* parameter used in ADMoM, based on existing variations of ADMoM in the literature (e.g., see [4] and references therein), enjoying faster convergence in large scale problems.

Algorithms for the distributed implementation of the successive convex approximation of BPPC are then derived, which allow tight approximation of the BPPC objective in the general SINR regime. Feedback requirements and consensus-on-termination issues are addressed, in order to propose fully decentralized protocols. Further exploiting the push-pull nature of the solution and periodic patterns arising in stable setups, as well as managing distributed warm start, we develop custom adaptive algorithms, which reproduce the results of their batch counterparts at far lower complexity. Simulation experiments verify that the distributed implementations proposed here match the performance of their centralized counterparts in [34].

An iteratively re-weighted Minimum Mean Square Error (MMSE) approach to weighted sum rate maximization for the MIMO interference channel has been very recently proposed in [50]. The algorithm in [50] (WMMSE) provides an one-shot approximate solution to the weighted sum rate maximization problem - it cannot be tuned to approximate the problem around different operating points, as needed for successive convex approximation. We show that this approach applies in our cross-layer networking context and we specialize WMMSE algorithm for the BPPC problem. A second distributed protocol, based on the WMMSE algorithm, is then presented for the approximation of BPPC. Although WMMSE has low computational complexity, we propose a simple improvement of the algorithm and derive its corresponding adaptive version, exploiting the nature of the solution resulting in stable setups in our context. Performance evaluation of all distributed algorithms for BPPC and comparisons among them follow through extensive simulation experiments, which illustrate the potentials of the algorithms.

2.2 System Model and Problem Statement

The same system model adopted in chapter 1 is also considered here, but will be described again, for convenience. The wireless multi-hop network comprising N nodes, can be modeled by the directed graph $(\mathcal{N}, \mathcal{L})$, where $\mathcal{N} := \{1, \ldots, N\}$ and $\mathcal{L} := \{1, \ldots, L\}$ denote the set of nodes

and the set of links, respectively. Each link $\ell \in \mathcal{L}$ corresponds to an ordered pair (i, j), where $i, j \in \mathcal{N}$ and $i \neq j$. Denoting by $\mathrm{Tx}(\ell)$ the transmitter, and by $\mathrm{Rx}(\ell)$ the receiver of link ℓ, for each $\ell = (i, j)$, then $\mathrm{Tx}(\ell) = i$ and $\mathrm{Rx}(\ell) = j$. We assume that each node is equipped with a single antenna, thus each transmitter-receiver pair has a single transmit and receive antenna. Furthermore, we assume that any node can transmit data to any other node, and may also split its incoming traffic into multiple outgoing links. Crosstalk factors $G_{k\ell}$ denote the aggregate path loss between the transmitter of link k, $\mathrm{Rx}(k)$, and the receiver of link ℓ, $\mathrm{Rx}(\ell)$. Let p_ℓ denote the power transmitted on link ℓ and V_ℓ the background noise power at the receiver of link ℓ. Then, the Signal to Interference plus Noise Ratio (SINR) attained at the receiver of link ℓ is

$$\gamma_\ell = \frac{G_{\ell\ell} p_\ell}{\frac{1}{G} \sum_{\substack{k=1 \\ k \neq \ell}}^{L} G_{k\ell} p_k + V_\ell}, \tag{2.1}$$

where G accounts for potential speading/beamfroming gain. In the sequel, G is absorbed by the coupling factors $G_{k\ell}$, $k \neq \ell$, for notational convenience. The transmitted rate on link ℓ is upper bounded by the maximum achievable rate c_ℓ

$$c_\ell = \log(1 + \gamma_\ell), \tag{2.2}$$

where the usual SINR gap parameter Γ can be introduced in the Shannon capacity formula to account for modulation loss, coding, etc. We skip this for brevity.

Similar to chapter 1, we assume a unit time slotted system, indexed dy t, and denote by $D_\ell(t)$ the *differential backlog* of link $\ell \in \mathcal{L}$ at the end of slot t, as defined in [34, 54]

$$D_\ell(t) := \begin{cases} \max\{0, W_i(t) - W_j(t)\}, & j \neq N \\ W_i(t), & j = N. \end{cases} \tag{2.3}$$

In case of multiple $\mathcal{F}$ flows, $D_\ell(t)$ is the maximum over all flows traversing link ℓ, i.e., with obvious notation, $D_\ell(t) := \max_{f \in \mathcal{F}} D_\ell^{(f)}(t)$, which is throughput-optimal in this case according to [54]. Then, the BPPC problem under per link power constraints is [14, 15, 21, 34, 41, 53–56]

$$\mathsf{BPPC}: \quad \max_{\{p_\ell\}_{\ell \in \mathcal{L}}} \sum_{\ell=1}^{L} D_\ell(t) \log \left(1 + \frac{G_{\ell\ell} p_\ell}{\sum_{\substack{k=1 \\ k \neq \ell}}^{L} G_{k\ell} p_k + V_\ell} \right) \tag{2.4}$$

$$\text{subject to} \quad 0 \leq p_\ell \leq P_\ell, \quad \ell \in \mathcal{L}, \tag{2.5}$$

where P_ℓ denotes the maximum power budget for each link.

The optimization problem (2.4)–(2.5), as well as its general formulation including per node power constraints, has been proved in [34] to be NP-hard. However, BPPC was also shown in [34] to be amenable to successive convex approximation strategies. The proposed implementations in [34] are centralized, whereas practical power control has to be distributed across the network. Here we address the distributed implementation of the approximation of problem in (2.4)–(2.5) in all interference regimes. We use first the successive convex approximation approach to approximate (2.4)–(2.5) and aim to derive the distributed versions of the centralized algorithms proposed in [34] for the above problem. Then, we utilize the WMMSE algorithm in [50] to approximate problem (2.4)–(2.5), and develop the final distributed algorithm for BPPC based on this second approach. In our developments we consider per link power constraints to be consistent with the centralized successive convex approximation algorithms in [34]. Per node power constraints may be used instead, or added, without changing the nature of the problem and of the solutions proposed in the sequel.

2.3 Distributed Successive Convex Approximation of BPPC

In this section we derive distributed algorithms for BPPC based on the successive convex approximation approach. As a first step, we develop a distributed version of the core step of the successive convex approximation, building upon the Alternating Direction Method of Multipliers (ADMoM). We propose two variations of the decentralized core step algorithm, derived in subsections 2.3.1, and 2.3.2. We then develop the distributed version of the complete successive convex approximation algorithm for BPPC (batch S.A.), as well as the corresponding one in the high-SINR regime, in subsection 2.3.3. Their adaptive versions are derived in subsection 2.3.4. Computational complexity of the algorithms, and feedback requirements for distributed implementation, are discussed in subsection2.3.5.

2.3.1 Distributed Convex Approximation of BPPC

The centralized approximate solution to the BPPC problem (2.4)–(2.5), as proposed in [32,34], is obtained via solving the following convex optimization problem

$$\max_{\{\tilde{p}_\ell\}_{\ell\in\mathcal{L}}} \sum_{\ell=1}^{L} D_\ell(t)\alpha_\ell(t)\left(\tilde{G}_{\ell\ell}+\tilde{p}_\ell-\log\left(\sum_{\substack{k=1\\k\neq\ell}}^{L} e^{\tilde{G}_{k\ell}+\tilde{p}_k}+e^{\tilde{V}_\ell}\right)\right)+\beta_\ell(t) \tag{2.6}$$

$$\text{subject to} \quad \tilde{p}_\ell \leq \tilde{P}_\ell, \quad \ell\in\mathcal{L}, \tag{2.7}$$

resulting from (2.4)–(2.5), upon lower-bounding the link rates at slot t (using an idea from [45, 46]) as $\log(1+\gamma_\ell) \geq \alpha_\ell(t)\log(\gamma_\ell)+\beta_\ell(t)$, and applying a logarithmic transformation of variables so that $\tilde{p}_\ell := \log p_\ell$, $\tilde{P}_\ell := \log(P_\ell)$, $\tilde{G}_{k_\ell} := \log(G_{k_\ell})$, and $\tilde{V}_\ell := \log(V_\ell)$. The successive convex approximation approach suggests solving at each t the problem in (2.6)–(2.7) to obtain a solution $\tilde{\boldsymbol{p}}(t) := \{\tilde{p}_\ell(t)\}_{\ell\in\mathcal{L}}$, then tighten the lower bounds by updating the parameters $\alpha_\ell(t)$ and $\beta_\ell(t)$ at point $\gamma_\ell(\tilde{\boldsymbol{p}}(t))$, for each link ℓ, solve the core problem in (2.6)–(2.7) for the updated $\{\alpha_\ell(t),\beta_\ell(t)\}_{\ell\in\mathcal{L}}$ towards a new $\tilde{\boldsymbol{p}}(t)$, and repeat until convergence, i.e., until the tightening step maps onto itself. This approach is also the essence of the SCALE algorithm in [45, 46].

Our aim here is to solve the core of the successive convex approximation of BPPC in a distributed fashion. Consider an arbitrary instance of problem (2.6)–(2.7) resulting in a time slot t, and for some fixed parameters $\alpha_\ell(t),\beta_\ell(t),D_\ell(t)$, for each link ℓ. We therefore omit the index t from all relevant quantities. Towards distributed implementation, we would ideally like each link ℓ to be able to optimize its own variable $\tilde{p}_\ell$, relying on as low-rate feedback as possible from other links. The core problem could then split into L subproblems that could be solved in parallel. Yet, this is not directly possible, due to the coupling of the power variables in the objective (2.6). As a first step, we may shift this coupling from the objective to the constraints, by introducing auxiliary variables $\{\tilde{i}_{\ell k}\}_{k\neq\ell}$ to represent the interference that link ℓ receives from link $k\neq\ell$, and consensus constraints $\tilde{G}_{k\ell}+\tilde{p}_k=\tilde{i}_{\ell k}, \forall k\neq\ell, \forall\ell\in\mathcal{L}$; e.g., cf. [9, 43, 49]. This yields

$$\min_{\tilde{\boldsymbol{p}},\left\{\tilde{\boldsymbol{i}}_\ell\right\}_{\ell\in\mathcal{L}}} \sum_{\ell=1}^{L} -D_\ell\alpha_\ell\left(\tilde{G}_{\ell\ell}+\tilde{p}_\ell\right)+D_\ell\alpha_\ell\log\left(\sum_{\substack{k=1\\k\neq\ell}}^{L} e^{\tilde{i}_{\ell k}}+e^{\tilde{V}_\ell}\right)-D_\ell\beta_\ell \tag{2.8}$$

$$\text{subject to} \quad \tilde{G}_{k\ell}+\tilde{p}_k=\tilde{i}_{\ell k}, \quad \forall k\neq\ell, \quad \forall\ell\in\mathcal{L}, \tag{2.9}$$

$$\text{subject to} \quad \tilde{p}_\ell\leq\tilde{P}_\ell, \quad \ell\in\mathcal{L}, \tag{2.10}$$

where $\tilde{\boldsymbol{p}}$ denotes the vector of variables $\{\tilde{p}_\ell\}_{\ell\in\mathcal{L}}$, $\tilde{\boldsymbol{i}}_\ell$ the vector of auxiliary variables $\left\{\tilde{i}_{\ell k}\right\}_{k\neq\ell}$ of link ℓ. Note that $\tilde{p}_\ell$ and $\left\{\tilde{i}_{\ell k}\right\}_{k\neq\ell}$ are local variables of link ℓ.

Note that the objective (2.8) is separable and convex with respect to the variables $\tilde{\boldsymbol{p}}$ and $\left\{\tilde{\boldsymbol{i}}_\ell\right\}_{\ell\in\mathcal{L}}$, as consists of an affine function of $\tilde{\boldsymbol{p}}$ and the logarithm of a sum of exponentials with respect to variables $\left\{\tilde{\boldsymbol{i}}_\ell\right\}_{\ell\in\mathcal{L}}$. Instead of using the traditional dual decomposition method, we utilize the Alternating Direction Method of Multipliers (ADMoM), in view of its favorable convergence properties, in combination with the gradient ascent method, in order to solve problem (2.8)–(2.10) in a distributed fashion. For link ℓ, let variables $\{\gamma_{\ell k}\}_{k\neq\ell}$ denote the Lagrange multipliers associated with its local equality constraints $\tilde{G}_{k\ell}+\tilde{p}_k=\tilde{i}_{\ell k}, \forall k\neq\ell$, and λ_ℓ the Lagrange multiplier associated with the power constraint $\tilde{p}_\ell\leq\tilde{P}_\ell$. The augmented Lagrangian with *penalty parameter* ρ, is given by

$$L_\rho\left(\tilde{\boldsymbol{p}},\left\{\tilde{\boldsymbol{i}}_\ell\right\}_{\ell\in\mathcal{L}},\left\{\{\gamma_{\ell k}\}_{k\neq\ell}\right\}_{\ell\in\mathcal{L}},\boldsymbol{\lambda}\right)=\sum_{\ell=1}^{L}-D_\ell\alpha_\ell\left(\tilde{G}_{\ell\ell}+\tilde{p}_\ell\right)+D_\ell\alpha_\ell\log\left(\sum_{\substack{k=1\\k\neq\ell}}^{L}e^{\tilde{i}_{\ell k}}+e^{\tilde{V}_\ell}\right)$$
$$-D_\ell\beta_\ell+\sum_{\ell=1}^{L}\lambda_\ell\left(\tilde{p}_\ell-\tilde{P}_\ell\right)+\sum_{\ell=1}^{L}\sum_{\substack{k=1\\k\neq\ell}}^{L}\gamma_{\ell k}\left(\tilde{G}_{k\ell}+\tilde{p}_k-\tilde{i}_{\ell k}\right)+\frac{\rho}{2}\sum_{\ell=1}^{L}\sum_{\substack{k=1\\k\neq\ell}}^{L}\left(\tilde{G}_{k\ell}+\tilde{p}_k-\tilde{i}_{\ell k}\right)^2. \tag{2.11}$$

Note that $\tilde{p}_\ell$ and $\left\{\tilde{i}_{\ell k}\right\}_{k\neq\ell}$ are *local primal* variables for link ℓ, while, λ_ℓ and $\{\gamma_{\ell k}\}_{k\neq\ell}$ are its *local dual* variables. Note, also, that the penalty quadratic term added according to ADMoM enforces strict convexity with respect to the primal variables. A key step, proceeding next, is to exploit the decomposable structure of the augmented Lagrangian in (2.11).

Notice in (2.11) that all terms are already decoupled across links, except for the last two terms. Considering first the term associated with the consensus constraints, we can decompose it as

$$\sum_{\ell=1}^{L}\sum_{\substack{k=1\\k\neq\ell}}^{L}\gamma_{\ell k}\left(\tilde{G}_{k\ell}+\tilde{p}_k-\tilde{i}_{\ell k}\right)=\sum_{\ell=1}^{L}\sum_{\substack{k=1\\k\neq\ell}}^{L}\gamma_{\ell k}\tilde{G}_{k\ell}+\sum_{\ell=1}^{L}\sum_{\substack{k=1\\k\neq\ell}}^{L}\gamma_{\ell k}\tilde{p}_k-\sum_{\ell=1}^{L}\sum_{\substack{k=1\\k\neq\ell}}^{L}\gamma_{\ell k}\tilde{i}\ell k.$$

The medium term, which is coupled across links with respect to the power variables, can be equivalently rewritten as

$$\sum_{\ell=1}^{L}\sum_{\substack{k=1\\k\neq\ell}}^{L}\gamma_{\ell k}p_k=\sum_{k=1}^{L}\sum_{\substack{\ell=1\\\ell\neq k}}^{L}\gamma_{\ell k}p_k,$$

which, after swapping variables k and ℓ, is equal to

$$\sum_{\ell=1}^{L}\sum_{\substack{k=1\\k\neq\ell}}^{L}\gamma_{k\ell}\tilde{p}_\ell=\sum_{\ell=1}^{L}\tilde{p}_\ell\sum_{\substack{k=1\\k\neq\ell}}^{L}\gamma_{k\ell}.$$

Finally, the augmented Lagrangian can be rewritten as

$$L_\rho\left(\tilde{\boldsymbol{p}},\left\{\tilde{\boldsymbol{i}}_\ell\right\}_{\ell\in\mathcal{L}},\left\{\{\gamma_{\ell k}\}_{k\neq\ell}\right\}_{\ell\in\mathcal{L}},\boldsymbol{\lambda}\right)=\sum_{\ell=1}^{L}\left(-D_\ell\alpha_\ell\left(\tilde{G}_{\ell\ell}+\tilde{p}_\ell\right)+D_\ell\alpha_\ell\log\left(\sum_{\substack{k=1\\k\neq\ell}}^{L}e^{\tilde{i}_{\ell k}}+e^{\tilde{V}_\ell}\right)\right.$$
$$\left.-D_\ell\beta_\ell+\lambda_\ell\left(\tilde{p}_\ell-\tilde{P}_\ell\right)+\sum_{\substack{k=1\\k\neq\ell}}^{L}\gamma_{k\ell}\tilde{G}_{\ell k}+\tilde{p}_\ell\left(\sum_{\substack{k=1\\k\neq\ell}}^{L}\gamma_{k\ell}\right)-\sum_{\substack{k=1\\k\neq\ell}}^{L}\gamma_{\ell k}\tilde{i}_{\ell k}\right)+\frac{\rho}{2}\sum_{\ell=1}^{L}\sum_{\substack{k=1\\k\neq\ell}}^{L}\left(\tilde{G}_{k\ell}+\tilde{p}_k-\tilde{i}_{\ell k}\right)^2. \tag{2.12}$$

The regularization term in L_ρ is already decomposable with respect to variables $\left\{\tilde{\boldsymbol{i}}_\ell\right\}_{\ell\in\mathcal{L}}$. With respect to $\tilde{\boldsymbol{p}}$ variable, it can also be decoupled across links, according to

$$\sum_{\ell=1}^{L}\sum_{\substack{k=1\\k\neq\ell}}^{L}\left(\tilde{G}_{k\ell}+\tilde{p}_k-\tilde{i}_{\ell k}\right)^2=\sum_{k=1}^{L}\sum_{\substack{\ell=1\\\ell\neq k}}^{L}\left(\tilde{G}_{k\ell}+\tilde{p}_k-\tilde{i}_{\ell k}\right)^2=\sum_{\ell=1}^{L}\sum_{\substack{k=1\\k\neq\ell}}^{L}\left(\tilde{G}_{\ell k}+\tilde{p}_\ell-\tilde{i}_{k\ell}\right)^2,$$

where the first equality holds due to $\sum_{x=1}^{N}\sum_{y=1}^{N}f(\mathbf{x},\mathbf{y})=\sum_{y=1}^{N}\sum_{x=1}^{N}f(\mathbf{x},\mathbf{y})$, and the second one results after swapping variables k and ℓ.

Then, the optimization steps of ADMoM for the update of variables $\tilde{\boldsymbol{p}}$, $\left\{\tilde{\boldsymbol{i}}_\ell\right\}_{\ell\in\mathcal{L}}$, and $\left\{\{\gamma_{\ell k}\}_{k\neq\ell}\right\}_{\ell\in\mathcal{L}}$, can be executed in parallel across all links in the network. Applying also a projected gradient step for the update of dual variable $\boldsymbol{\lambda}$, and denoting by s the iteration index, the iterates to be carried out at each link $\ell\in\mathcal{L}$ boil down to

$$\tilde{p}_\ell(s):=\arg\min_{\tilde{p}_\ell}-D_\ell\alpha_\ell\tilde{p}_\ell+\lambda_\ell(s-1)\tilde{p}_\ell+\tilde{p}_\ell\left(\sum_{\substack{k=1\\k\neq\ell}}^{L}\gamma_{k\ell}(s-1)\right)$$
$$+\frac{\rho}{2}\sum_{\substack{k=1\\k\neq\ell}}^{L}\left(\tilde{G}_{\ell k}+\tilde{p}_\ell-\tilde{i}_{k\ell}(s-1)\right)^2 \tag{2.13}$$

$$\left\{\tilde{i}_{\ell k}\right\}_{k\neq\ell}(s):=\arg\min_{\left\{\tilde{i}_{\ell k}\right\}_{k\neq\ell}}D_\ell\alpha_\ell\log\left(\sum_{\substack{k=1\\k\neq\ell}}^{L}e^{\tilde{i}_{\ell k}}+e^{\tilde{V}_\ell}\right)-\sum_{\substack{k=1\\k\neq\ell}}^{L}\gamma_{\ell k}(s-1)\tilde{i}_{\ell k}$$
$$+\frac{\rho}{2}\sum_{\substack{k=1\\k\neq\ell}}^{L}\left(\left(\tilde{G}_{k\ell}+\tilde{p}_k\right)(s)-\tilde{i}_{\ell k}\right)^2 \tag{2.14}$$

$$\gamma_{\ell k}(s):=\gamma_{\ell k}(s-1)+\rho\left(\left(\tilde{G}_{k\ell}+\tilde{p}_k\right)(s)-\tilde{i}_{\ell k}(s)\right),k\neq\ell \tag{2.15}$$

$$\lambda_\ell(s):=\left[\lambda_\ell(s-1)+\delta_s\left(\tilde{p}_\ell(s)-\tilde{P}_\ell\right)\right]_0^+. \tag{2.16}$$

Steps (2.13)–(2.16) are executed in parallel at all links, as long as certain feedback requirements are satisfied. Step (2.13) for link ℓ involves $\{\gamma_{k\ell}(s-1)\}_{k\neq\ell}$ and $\left\{\tilde{i}_{k\ell}(s-1)\right\}_{k\neq\ell}$, i.e., dual and auxiliary variables of all its interfering links $k\neq\ell$, as computed in the previous iteration $(s-1)$. This information can be communicated via message passing. Then, (2.13) is a convex quadratic in $\tilde{p}_\ell$, whose minimum can be found analytically. Specifically, the resulting closed form update is given by

$$\tilde{p}_\ell(s)=\frac{D_\ell\alpha_\ell-\lambda_\ell(s-1)-\sum_{\substack{k=1\\k\neq\ell}}^{L}\gamma_{k\ell}(s-1)+\rho\sum_{\substack{k=1\\k\neq\ell}}^{L}\left(\tilde{i}_{k\ell}(s-1)-\tilde{G}_{\ell k}\right)}{\rho\,(L-1)},\forall\ell\in\mathcal{L}. \tag{2.17}$$

The unconstrained minimization in (2.14) with respect to $\tilde{\boldsymbol{i}}_\ell$ is carried out at link ℓ, which is assumed to have knowledge of the interference $\left(\tilde{G}_{k\ell}+\tilde{p}_k\right)(s)$ received from link $k\neq\ell$ at iteration s. Depending on the operational setup, this can either be estimated by ℓ, or communicated to ℓ. Then, variables $\tilde{\boldsymbol{i}}_\ell$ are updated by solving (2.14) using e.g., damped Newton's method. Next, step (2.15) is straightforward. Notice that ADMoM requires a step size for (2.15) equal to the parameter ρ, in order for its convergence properties to hold, along with other assumptions; cf. [2–4]. The update in (2.16) is carried out according to the dual ascent method. The associated step-size sequence δ_s can be chosen as $\delta_s=\delta_1/s$, or a sufficiently small constant $\delta_s=\delta$ can be employed - convergence of the iterates to the optimal centralized solution is guaranteed in both cases. We chose a small constant step size in our simulations, because it resulted in faster convergence. The resulting distributed algorithm is summarized as follows:

Algorithm 6. Distributed convex approximation of BPPC (**core step 1**) algorithm *:*
Given D_ℓ, α_ℓ, β_ℓ, $\forall\ell\in\mathcal{L}$, and $s:=$ iteration counter,

- ***Initialization:*** *For $s=0$ set: $\rho>0$, $\delta_0>0$, $\{\lambda_\ell(0)\}_{\ell=1}^{L}>0$, $\{\gamma_{\ell k}(0)\}_{\ell\in\mathcal{L},k\neq\ell}>0$, and $\left\{\tilde{i}_{\ell k}(0)\right\}_{\ell\in\mathcal{L},k\neq\ell}$ random.*
- *$\forall\ell\in\mathcal{L}$: transmit initial $\gamma_{\ell k}(0)$ and $\tilde{i}_{\ell k}(0)$ to link k, $\forall k\neq\ell$.*
- ***Repeat:*** *Set $s:=s+1$*
  1. *$\forall\ell\in\mathcal{L}$: update $\tilde{p}_\ell(s)$ according to* (2.17).
  2. *$\forall\ell\in\mathcal{L}$: update $\left\{\tilde{i}_{\ell k}(s)\right\}_{k\neq\ell}$ by solving* (2.14).
  3. *$\forall\ell\in\mathcal{L}$: update $\{\gamma_{\ell k}(s)\}_{k\neq\ell}$ according to* (2.15).

4. $\forall \ell \in \mathcal{L}$*: update* $\lambda_\ell(s)$ *according to* (2.16)*.*

5. $\forall \ell \in \mathcal{L}$*: transmit* $\gamma_{\ell k}(s)$ *and* $\tilde{i}_{\ell k}(s)$ *to link* k*,* $\forall k \neq \ell$*.*

- ***Until:*** *convergence (within* ϵ*-accuracy);*
  then set: $\tilde{p}_\ell^{opt} := \tilde{p}_\ell(s)$*,* $\tilde{\boldsymbol{i}}_\ell^{opt} := \tilde{\boldsymbol{i}}_\ell(s)$*,* $\lambda_\ell^{opt} := \lambda_\ell(s)$*,* $\left\{\gamma_{\ell k}^{opt}\right\}_{k\neq\ell} := \{\gamma_{\ell k}(s)\}_{k\neq\ell}$*,* $\forall \ell \in \mathcal{L}$*.*

Convergence of the algorithm can be determined based on local computation and communication. Each link ℓ keeps track of a local metric, such as the value of its local Lagrange function, and/or the norm of its residual local equality constraint violation vector $\mathbf{r}_\ell(s)$, with elements $r_{\ell k}(s) := \tilde{G}_{k\ell} + \tilde{p}_k(s) - \tilde{i}_{\ell k}(s)$, $\forall k \neq \ell$, evaluated at s. Each link maintains a binary flag taking the value 1 whenever convergence with respect to its local metric has been achieved, within a given accuracy. Then, a distributed consensus-on-the-min algorithm [10, 49] can be employed among links, so that iterates terminate once all links reach convergence.

2.3.2 Variation with Varying Local Penalty Parameters

Convergence of ADMoM is guaranteed for any positive value of the penalty parameter, cf. [2–4]. However, many variations have been explored in the literature cf. [2, 4] (and references therein). Some of these variants can give superior superior convergence in practice compared to the standard ADMoM. One possibility is to consider more general augmenting terms, allowing for different penalty parameter for each constraint. It is also possible to vary the penalty parameters by iteration according to a scheme, while convergence results still apply under certain assumptions, or under the assumption that the penalty parameters become fixed after a finite number of iterations, cf. [4, 18, 57].

In order to possibly improve convergence of the core step 1 algorithm and to make its performance less dependent on the initial choice of the penalty parameter, we develop here a variation of the algorithm based on the aforementioned ideas. Specifically, we consider a different penalty parameter for each link, used as the step-size for the update step of its dual variables $\{\gamma_{\ell k}\}_{k\neq\ell}$, which can be varied by iteration according to the scheme proposed in [4]. Let therefore ρ_ℓ^s denote the penalty parameter for link ℓ, used at iteration s. The varying scheme in [4] involves the primal and dual residual vectors which are viewed therein as the residuals for the primal and dual feasibility condition for the problem to which ADMoM is applied, respectively. By direct application to problem (2.8)–(2.10), the *primal residual* vector, defined in [4], corresponds to

$\mathbf{r}_\ell(s)$, with elements $\{r_{\ell k}(s)\}_{k\neq\ell} := \left\{\tilde{G}_{k\ell} + \tilde{p}_k(s) - \tilde{i}_{\ell k}(s)\right\}_{k\neq\ell}$, evaluated at s. The *dual residual* vector corresponds to $\mathbf{d}_\ell(s)$, with elements $\{d_{\ell k}(s)\}_{k\neq\ell} := \rho_\ell^s \{i_{\ell k}(s) - i_{\ell k}(s-1)\}_{k\neq\ell}$, evaluated at s. Then, at each iteration s, the penalty parameter ρ_ℓ^{s+1} for the next iteration $s+1$, of each link $\ell \in \mathcal{L}$, is updated according to the following scheme [4, 18, 57]:

$$\rho_\ell^{s+1} := \begin{cases} \tau\rho_\ell^s, & \|\mathbf{r}_\ell(s)\|_2 > \mu\|\mathbf{d}_\ell(s)\|_2 \\ \frac{\rho_\ell^s}{\tau}, & \|\mathbf{d}_\ell(s)\|_2 > \mu\|\mathbf{r}_\ell(s)\|_2 \\ \rho_\ell^s, & otherwise \end{cases} \tag{2.18}$$

where $\mu > 1$ and $\tau > 1$ are parameters. Typical choices might be $\mu = 10$ and $\tau = 2$. According to the convergence properties of ADMoM (cf. [4]) both primal and dual residual norms converge to zero, as the method converges. The above varying mode tends to keep the residual norms within a factor of μ of one another, in order to speed up convergence. To gain some insight, notice that a large value of ρ_ℓ places a large penalty on violations of the consensus constraints $\left\{\tilde{G}_{k\ell} + \tilde{p}_k(s) - \tilde{i}_{\ell k}(s)\right\}_{k\neq\ell}$ of link ℓ and, therefore, tends to reduce the norm of the residual vector $\mathbf{r}_\ell$. On the other hand, a small value of ρ_ℓ implies small value of the dual residual norm $\mathbf{d}_\ell$, as its definition suggests, and looses the penalty for the violation of the consensus constraints of ℓ, allowing this way for larger primal residual norm. Thus, the varying scheme in (2.18) favors in each case the smaller residual norm.

Following the derivation of the optimization steps of the core step 1 algorithm for the following augmented Lagrange function with $\{\rho_\ell\}_{\ell\in\mathcal{L}}$ penalty parameters

$$L_{\{\rho_\ell\}_{\ell\in\mathcal{L}}} = \sum_{\ell=1}^{L} \left(-D_\ell\alpha_\ell \left(\tilde{G}_{\ell\ell} + \tilde{p}_\ell\right) + D_\ell\alpha_\ell \log\left(\sum_{\substack{k=1\\k\neq\ell}}^{L} e^{\tilde{i}_{\ell k}} + e^{\tilde{V}_\ell}\right) - D_\ell\beta_\ell + \lambda_\ell\left(\tilde{p}_\ell - \tilde{P}_\ell\right) \right.$$
$$\left. + \sum_{\substack{k=1\\k\neq\ell}}^{L} \gamma_{k\ell}\tilde{G}_{\ell k} + \tilde{p}_\ell\left(\sum_{\substack{k=1\\k\neq\ell}}^{L}\gamma_{k\ell}\right) - \sum_{\substack{k=1\\k\neq\ell}}^{L}\gamma_{\ell k}\tilde{i}_{\ell k}\right) + \sum_{\ell=1}^{L}\frac{\rho_\ell}{2}\sum_{\substack{k=1\\k\neq\ell}}^{L}\left(\tilde{G}_{k\ell} + \tilde{p}_k - \tilde{i}_{\ell k}\right)^2, \tag{2.19}$$

the respective update steps, for each link $\ell \in \mathcal{L}$, become

$$\tilde{p}_\ell(s) := \arg\min_{\tilde{p}_\ell} -D_\ell\alpha_\ell\tilde{p}_\ell + \lambda_\ell(s-1)\tilde{p}_\ell + \tilde{p}_\ell\left(\sum_{\substack{k=1\\k\neq\ell}}^{L}\gamma_{k\ell}\,(s-1)\right)$$

$$+\sum_{\substack{k=1\\k\neq\ell}}^{L}\frac{\rho_k^s}{2}\left(\tilde{G}_{\ell k}+\tilde{p}_\ell-\tilde{i}_{k\ell}(s-1)\right)^2 \tag{2.20}$$

or,

$$\tilde{p}_\ell(s)=\frac{D_\ell\alpha_\ell-\lambda_\ell(s-1)-\sum_{\substack{k=1\\k\neq\ell}}^{L}\gamma_{k\ell}(s-1)+\sum_{\substack{k=1\\k\neq\ell}}^{L}\rho_k^s\left(\tilde{i}_{k\ell}(s-1)-\tilde{G}_{\ell k}\right)}{\sum_{\substack{k=1\\k\neq\ell}}^{L}\rho_k^s} \tag{2.21}$$

$$\left\{\tilde{i}_{\ell k}\right\}_{k\neq\ell}(s):=\arg\min_{\left\{\tilde{i}_{\ell k}\right\}_{k\neq\ell}} D_\ell\alpha_\ell\log\left(\sum_{\substack{k=1\\k\neq\ell}}^{L}e^{\tilde{i}_{\ell k}}+e^{\tilde{V}_\ell}\right)-\sum_{\substack{k=1\\k\neq\ell}}^{L}\gamma_{\ell k}(s-1)\tilde{i}_{\ell k}$$
$$+\frac{\rho_\ell^s}{2}\sum_{\substack{k=1\\k\neq\ell}}^{L}\left(\left(\tilde{G}_{k\ell}+\tilde{p}_k\right)(s)-\tilde{i}_{\ell k}\right)^2 \tag{2.22}$$

$$\gamma_{\ell k}(s):=\gamma_{\ell k}(s-1)+\rho_\ell^s\left(\left(\tilde{G}_{k\ell}+\tilde{p}_k\right)(s)-\tilde{i}_{\ell k}(s)\right),k\neq\ell \tag{2.23}$$

$$\lambda_\ell(s):=\left[\lambda_\ell(s-1)+\delta_s\left(\tilde{p}_\ell(s)-\tilde{P}_\ell\right)\right]_0^+. \tag{2.24}$$

Note that in this case the power update step in (2.21) for link ℓ requires knowledge of the penalty parameters $\{\rho_k^s\}_{k\neq\ell}$, of all its interfering links $k\neq\ell$, updated at iteration $(s-1)$ according to the scheme (2.18), whenever they do change. This information must be communicated to ℓ additionally to the variables $\{\gamma_{k\ell}(s-1)\}_{k\neq\ell}$ and $\left\{\tilde{i}_{k\ell}(s-1)\right\}_{k\neq\ell}$. Consequently, the potential benefit of using varying local penalty parameters in the efficiency and convergence of the algorithm, comes at the expense of increasing the communication overhead. The version of the core step algorithm with varying local penalty parameters is described next.

Algorithm 7. Distributed convex approximation of BPPC (**core step 2**) algorithm*:*

Given D_ℓ, α_ℓ, β_ℓ, $\forall\ell\in\mathcal{L}$, and $s:=$ iteration counter,

- ***Initialization:*** *For $s=0$ set: $\left\{\rho_\ell^1\right\}_{\ell\in\mathcal{L}}>0$, $\delta_0>0$, $\{\lambda_\ell(0)\}_{\ell=1}^L>0$, $\{\gamma_{\ell k}(0)\}_{\ell\in\mathcal{L},k\neq\ell}>0$, and $\left\{\tilde{i}_{\ell k}(0)\right\}_{\ell\in\mathcal{L},k\neq\ell}$ random.*
- *$\forall\ell\in\mathcal{L}$: transmit initial $\gamma_{\ell k}(0)$, $\tilde{i}_{\ell k}(0)$, and ρ_ℓ^1 to link k, $\forall k\neq\ell$.*
- ***Repeat:*** *Set $s:=s+1$*
  1. *$\forall\ell\in\mathcal{L}$: update $\tilde{p}_\ell(s)$ according to* (2.21).
  2. *$\forall\ell\in\mathcal{L}$: update $\left\{\tilde{i}_{\ell k}(s)\right\}_{k\neq\ell}$ by solving* (2.22).

3. $\forall \ell \in \mathcal{L}$*: update* $\{\gamma_{\ell k}(s)\}_{k \neq \ell}$ *according to* (2.23)*.*

4. $\forall \ell \in \mathcal{L}$*: update* $\lambda_\ell(s)$ *according to* (2.24)*.*

5. $\forall \ell \in \mathcal{L}$*: update* ρ_ℓ^{s+1} *according to scheme* (2.18)

6. $\forall \ell \in \mathcal{L}$*: transmit* $\gamma_{\ell k}(s)$*,* $\tilde{i}_{\ell k}(s)$*, and* ρ_ℓ^{s+1} *to link* k*,* $\forall k \neq \ell$*.*

- ***Until:*** *convergence (within* ϵ*-accuracy);*
  then set: $\tilde{p}_\ell^{opt} := \tilde{p}_\ell(s)$*,* $\tilde{\boldsymbol{i}}_\ell^{opt} := \tilde{\boldsymbol{i}}_\ell(s)$*,* $\lambda_\ell^{opt} := \lambda_\ell(s)$*,* $\left\{\gamma_{\ell k}^{opt}\right\}_{k \neq \ell} := \{\gamma_{\ell k}(s)\}_{k \neq \ell}$*,* $\forall \ell \in \mathcal{L}$*.*

Determination of local convergence by each link is based on the aforementioned local convergence metrics, while termination of the iterates among links is accomplished via a consensus on-the-min algorithm, as in the case of core step 1 algorithm.

2.3.3 Distributed Successive Convex Approximation Algorithms for BPPC

In this section we develop the decentralized batch algorithms for the BPPC problem in (2.6)–(2.7), operating in the high-SINR and in the general SINR regime, in correspondence with their centralized counterparts developed in [34]. In the unit time slotted system we consider, the batch algorithms solve the problem (2.6)–(2.7), resulting at each time slot, from scratch. Similar to the centralized algorithms, we make use of the following bound in order to lower-bound the individual link rates [45, 46]:

$$\alpha \log(z) + \beta \leq \log(1+z) \text{ for } \begin{cases} \alpha = \frac{z_o}{1+z_o}, \\ \beta = \log(1+z_o) - \frac{z_o}{1+z_o} \log(z_o) \end{cases} \tag{2.25}$$

which is tight at z_o. The high-SINR approximation corresponds to using $\alpha_\ell(t) = 1$ and $\beta_\ell(t) = 0$, $\forall \ell \in \mathcal{L}$, in (2.6)–(2.7), while we utilize the core-step algorithms developed herein for the distributed implementation of the convex optimization problem. The distributed batch high-SINR algorithm is then

Algorithm 8. Distributed Batch high-SINR for BPPC*:*

For each time slot $t \geq 1$*:*

1. $\forall \ell \in \mathcal{L}$*: Calculate* $D_\ell(t)$ *according to*
$$D_\ell(t) := \begin{cases} \max\{0, W_i(t) - W_j(t)\}, & j \neq N \\ W_i(t), & j = N. \end{cases}$$
2. *Initialization:* $\forall \ell \in \mathcal{L}$*: set* $\alpha_\ell(t) = 1$ *and* $\beta_\ell(t) = 0$*.*
3. *Run the* ***core step*** *algorithm for the given* $D_\ell(t)$*,* $\alpha_\ell(t)$*,* $\beta_\ell(t)$*,* $\forall \ell \in \mathcal{L}$*,*
get $p_\ell^{opt}(t)$ *from* $\tilde{p}_\ell^{opt}$*,* $\forall \ell \in \mathcal{L}$*.*

For the distributed batch S.A. algorithm we successively refine this approximation by tightening the individual link rates bounds, in correspondence with the centralized batch S.A. in [34], yielding the following distributed algorithm

Algorithm 9. Distributed Batch Successive Convex Approximation for BPPC (Batch S.A.)*:*

For each time slot $t \geq 1$*:*

1. $\forall \ell \in \mathcal{L}$*: Calculate* $D_\ell(t)$ *according to*
$$D_\ell(t) := \begin{cases} \max\{0, W_i(t) - W_j(t)\}, & j \neq N \\ W_i(t), & j = N. \end{cases}$$
2. *Initialization* $\forall \ell \in \mathcal{L}$*: Reset iteration counter* $s = 1$*, set* $\alpha_\ell(t,s) = 1$ *and* $\beta_\ell(t,s) = 0$*.*
3. ***repeat:***
4. *If* $s = 1$*: Run the* ***core step*** *algorithm for the given* $D_\ell(t)$*,* $\alpha_\ell(t,s)$*, and* $\beta_\ell(t,s)$*,* $\forall \ell \in \mathcal{L}$*.*
Set: $\boldsymbol{\lambda}_o := \boldsymbol{\lambda}^{opt}$*,* $\left\{\{\gamma_{\ell k,o}\}_{k\neq\ell}\right\}_{\ell\in\mathcal{L}} := \left\{\left\{\gamma_{\ell k}^{opt}\right\}_{k\neq\ell}\right\}_{\ell\in\mathcal{L}}$*,* $\left\{\tilde{\boldsymbol{i}}_{\ell,o}\right\}_{\ell\in\mathcal{L}} := \left\{\tilde{\boldsymbol{i}}_\ell^{opt}\right\}_{\ell\in\mathcal{L}}$*.*

Else for $s > 1$ *: Run the* ***core step*** *algorithm for given*
$\left\{D_\ell(t), \alpha_\ell(t,s), \beta_\ell(t,s), \lambda_{\ell,o}, \{\gamma_{\ell k,o}\}_{k\neq\ell}, \tilde{\boldsymbol{i}}_{\ell,o}\right\}_{\ell\in\mathcal{L}}$*.*
Update: $\boldsymbol{\lambda}_o := \boldsymbol{\lambda}^{opt}$*,* $\left\{\{\gamma_{\ell k,o}\}_{k\neq\ell}\right\}_{\ell\in\mathcal{L}} := \left\{\left\{\gamma_{\ell k}^{opt}\right\}_{k\neq\ell}\right\}_{\ell\in\mathcal{L}}$*,* $\left\{\tilde{\boldsymbol{i}}_{\ell,o}\right\}_{\ell\in\mathcal{L}} := \left\{\tilde{\boldsymbol{i}}_\ell^{opt}\right\}_{\ell\in\mathcal{L}}$*.*
5. $\forall \ell \in \mathcal{L}$*: Obtain solution* $p_\ell(t,s)$ *from* $\tilde{p}_\ell^{opt}$*,*
use solution $\tilde{\boldsymbol{i}}_\ell^{opt}$ *to calculate total received interference:* $I_\ell(t,s) = \sum_{\substack{k=1 \\ k\neq\ell}}^{L} e^{\tilde{i}_{\ell k}^{opt}}$*.*
6. *Tightening step* $\forall \ell \in \mathcal{L}$*: Pick* $\alpha_\ell(t,s+1)$ *and* $\beta_\ell(t,s+1)$ *according to* (2.25)*,*
for $z_o = \log\left(1 + \frac{G_{\ell\ell} p_\ell(t,s)}{I_\ell(t,s) + V_\ell}\right)$*.*

7. $s = s + 1$

8. ***until*** $|\alpha_\ell(t,s) - \alpha_\ell(t,s-1)| \leq \epsilon$ *and* $|\beta_\ell(t,s) - \beta_\ell(t,s-1)| \leq \epsilon$, $\forall \ell \in \mathcal{L}$.

9. $\forall \ell \in \mathcal{L}$*: set* $p_\ell(t)^{opt} := p_\ell(t,s)$.

For the termination criterion in step 8 of the above algorithm each link ℓ keeps track of a local metric, such as the variation of its local augmented Lagrange function, or alternatively, of its parameters $\alpha_\ell(t), \beta_\ell(t)$. Note that at convergence of the successive convex approximation approach any further tightening step maps the parameters $\alpha_\ell(t), \beta_\ell(t)$ onto themselves, and thus, results in the same operating point for each link. The outer iterations of the S.A. algorithm (successive convex approximations) must terminate once all links reach convergence (within ϵ-accuracy) with respect to their local metric. Towards this end, a consensus-on-the-min algorithm [9,4] can be employed among links, similarly as in the core step algorithm. The distributed core step algorithm employed in the above algorithms can be either the version with the constant global ρ (core step 1), or the one using varying local penalty parameters $\{\rho_\ell\}_{\ell \in \mathcal{L}}$ (core step 2).

Note also that, as in the case of the centralized approximation algorithms, links destined to more congested nodes have zero differential backlog from (2.3) and are down-weighted in (2.4). Therefore, the above distributed algorithms account only for the *scheduled* links, i.e., links with positive differential backlog. In practice, problem (2.8)–(2.10) corresponds to links $\ell \in \mathcal{L}'$, where $\mathcal{L}' := \{\ell | D_\ell(t) > 0\}$. Also, links with resulting zero operating point in step 6 of batch S.A. (estimated by each link via its auxiliary variables), i.e., links which attain SINR below machine precision, stop iterating and remain silent. This is enforced for technical reasons, on the one hand, because, otherwise, these links cause inaccuracies and the Newton step in the minimization (2.14), or (2.22), can not be computed (singular Hessian matrices). On the other hand, it is also reasonable. To see this, consider some links, which after the high-SINR convex approximation, where the individual link rate bounds are loose, are not operational, i.e., their attained SINRs are below machine precision. Then, these links are precluded from being operational at the subsequent operating point, which corresponds to tighter lower bounds of all link rates, and to increased interference level (general SINR regime). Whenever not operational links result after every convex approximation they can be set idle.

2.3.4 Distributed Adaptive Algorithms for BPPC

The motivation for considering adaptive solutions to the BPPC problem in [34] is that it has to be solved on a per-slot basis. This is due to the fact that the differential backlogs can change dynamically from each scheduling slot to the next, as packets are routed in the network, even if the physical layer propagation conditions are assumed fixed. Assuming deterministic fixed-rate (or random but bounded) arrivals and fixed physical layer propagation conditions, in addition to no-listen-while -you-talk considerations, we expect, as explained in [34], the system to operate in a multi-stage push-pull fashion, exhibiting this way a periodic-behavior, for stable setups. In particular, the centralized solution to the BPPC problem, obtained in [34] via the S.A. approach, for stable setups, is characterized by a periodic pattern with respect to the subsets of activated links, their resulting differential backlogs and the power allocation. Exploiting this expected (quasi-)periodicity of the solution for stable setups, we develop here analogous distributed adaptive algorithms. The strategy used in [34] for the warm re-start of the adaptive algorithms, is the initialization at the one of the W solutions computed for the past W time slots, that is closer to the optimal one for the present slot. The length W of the window of previous solutions must be larger than the expected emerging period.

The difference in the case of the distributed adaptive algorithms is that the corresponding warm start must be also implemented in a distributed fashion. Each link must therefore utilize a local metric in order to pick the one of its already encountered solutions that is closer to the optimal one for the present slot, in terms of maximum stable end-to-end throughput. Note that a selfish initialization of each link at the solution that maximizes its own rate, for the present slot, is not optimal for the system's utility maximization. Since for stable setups the system returns to a previously visited state, each link should then pick the solution computed at the one of the previous states, that is closer to the current state of the system. This is possible by exploiting the periodicity of the differential backlogs arising in stable setups. Consequently, each link ℓ picks the past solution corresponding to the differential backlog that is closer to the current one. This idea is implemented in the following adaptive algorithms.

Algorithm 10. Distributed Adaptive high-SINR for BPPC*:*

For each time slot $t \geq 1$*:*

1. *Calculate the differential backlogs* $D_\ell(t)$, $\forall \ell \in \mathcal{L}$, *according to*
$$D_\ell(t) := \begin{cases} \max\{0, W_i(t) - W_j(t)\}, & j \neq N \\ W_i(t), & j = N. \end{cases}$$

2. *Initialization* $\forall \ell \in \mathcal{L}$*: set* $\alpha_\ell(t) = 1$ *and* $\beta_\ell(t) = 0$.

3. *Variables initialization:*

   For $t = 1$ *set* $\forall \ell \in \mathcal{L}$*:* $\lambda_{\ell,o} > 0$, $\{\gamma_{\ell k,o}\}_{k \neq \ell} > 0$, *and* $\tilde{\boldsymbol{i}}_{\ell,o}$ *random.*

   For $t \in [2, W]$ *set* $\forall \ell \in \mathcal{L}$*:*

$$\tau_o = \arg \min_{\tau \in \{t-1,\ldots,1\}} |D_\ell(t) - D_\ell(\tau)|$$

$$\lambda_{\ell,o} := \lambda_\ell^{opt}(t - \tau_o), \quad \{\gamma_{\ell k,o}\}_{k \neq \ell} := \left\{\gamma_{\ell k}^{opt}(t - \tau_o)\right\}_{k \neq \ell}, \quad \tilde{\boldsymbol{i}}_{\ell,o} := \tilde{\boldsymbol{i}}_\ell^{opt}(t - \tau_o)$$

   For $t \geq W + 1$ *set* $\forall \ell \in \mathcal{L}$*:*

$$\tau_o = \arg \min_{\tau \in \{t-1,\ldots,t-W\}} |D_\ell(t) - D_\ell(\tau)|$$

$$\lambda_{\ell,o} := \lambda_\ell^{opt}(t - \tau_o), \quad \{\gamma_{\ell k,o}\}_{k \neq \ell} := \left\{\gamma_{\ell k}^{opt}(t - \tau_o)\right\}_{k \neq \ell}, \quad \tilde{\boldsymbol{i}}_{\ell,o} := \tilde{\boldsymbol{i}}_\ell^{opt}(t - \tau_o).$$

4. *Run the* ***core step*** *algorithm for given* $\left\{D_\ell(t), \alpha_\ell(t), \beta_\ell(t), \lambda_{\ell,o}, \{\gamma_{\ell k,o}\}_{k \neq \ell}, \tilde{\boldsymbol{i}}_{\ell,o}\right\}_{\ell \in \mathcal{L}}$*;*
   get $p_\ell^{opt}(t)$ *from* $\tilde{p}_\ell^{opt}$, $\forall \ell \in \mathcal{L}$,
   set: $\lambda_\ell^{opt}(t) := \lambda_\ell^{opt}$, $\left\{\gamma_{\ell k}^{opt}(t)\right\}_{k \neq \ell} := \left\{\gamma_{\ell k}^{opt}\right\}_{k \neq \ell}$, $\tilde{\boldsymbol{i}}_\ell^{opt}(t) = \tilde{\boldsymbol{i}}_\ell^{opt}$, $\forall \ell \in \mathcal{L}$.

The distributed adaptive algorithm for the successive convex approximation approach, correspondingly, follows next.

Algorithm 11. Distributed Adaptive Successive Convex Approximation for BPPC (Adaptive S.A.)*:*

For each time slot $t \geq 1$*:*

1. *Calculate the differential backlogs* $D_\ell(t)$, $\forall \ell \in \mathcal{L}$, *according to*
$$D_\ell(t) := \begin{cases} \max\{0, W_i(t) - W_j(t)\}, & j \neq N \\ W_i(t), & j = N. \end{cases}$$

2. *Initialization: Reset iteration counter* $s = 1$.

3. *Variables-parameters initialization:*

   For $t = 1$ *set* $\forall \ell \in \mathcal{L}$*:* $\alpha_\ell(t,s) = 1$, $\beta_\ell(t,s) = 0$,

   $\lambda_{\ell,o} > 0$, $\{\gamma_{\ell k,o}\}_{k\neq\ell} > 0$, *and* $\tilde{\boldsymbol{i}}_{\ell,o}$ *random.*

   For $t \in [2, W]$ *set* $\forall \ell \in \mathcal{L}$*:*

$$\tau_o = \arg \min_{\tau \in \{t-1,\ldots,1\}} |D_\ell(t) - D_\ell(\tau)|$$

$$\alpha_\ell(t,s) := \alpha_\ell^{opt}(t-\tau_o), \quad \beta_\ell(t,s) := \beta_\ell^{opt}(t-\tau_o)$$

$$\lambda_{\ell,o} := \lambda_\ell^{opt}(t-\tau_o), \quad \{\gamma_{\ell k,o}\}_{k\neq\ell} := \left\{\gamma_{\ell k}^{opt}(t-\tau_o)\right\}_{k\neq\ell}, \quad \tilde{\boldsymbol{i}}_{\ell,o} := \tilde{\boldsymbol{i}}_\ell^{opt}(t-\tau_o)$$

   For $t \geq W+1$ *set* $\forall \ell \in \mathcal{L}$*:*

$$\tau_o = \arg \min_{\tau \in \{t-1,\ldots,t-W\}} |D_\ell(t) - D_\ell(\tau)|$$

$$\alpha_\ell(t,s) := \alpha_\ell^{opt}(t-\tau_o), \quad \beta_\ell(t,s) := \beta_\ell^{opt}(t-\tau_o)$$

$$\lambda_{\ell,o} := \lambda_\ell^{opt}(t-\tau_o), \quad \{\gamma_{\ell k,o}\}_{k\neq\ell} := \left\{\gamma_{\ell k}^{opt}(t-\tau_o)\right\}_{k\neq\ell}, \quad \tilde{\boldsymbol{i}}_{\ell,o} := \tilde{\boldsymbol{i}}_\ell^{opt}(t-\tau_o).$$

4. *Run the* ***core step*** *algorithm for given* $\left\{D_\ell(t), \alpha_\ell(t,s), \beta_\ell(t,s), \lambda_{\ell,o}, \{\gamma_{\ell k,o}\}_{k\neq\ell}, \tilde{\boldsymbol{i}}_{\ell,o}\right\}_{\ell\in\mathcal{L}}$*;*
   Update: $\tilde{p}_\ell^{opt}$, $\tilde{\boldsymbol{i}}_\ell^{opt}$, λ_ℓ^{opt}, $\left\{\gamma_{\ell k}^{opt}\right\}_{k\neq\ell}$, $\forall \ell \in \mathcal{L}$.

5. $\forall \ell \in \mathcal{L}$*: Obtain solution* $p_\ell(t,s)$ *from* $\tilde{p}_\ell^{opt}$,
   use solution $\tilde{\boldsymbol{i}}_\ell^{opt}$ *to calculate total received interference:* $I_\ell(t,s) = \sum_{\substack{k=1\\k\neq\ell}}^{L} e^{\tilde{i}_{\ell k}}$.

6. *Tightening step* $\forall \ell \in \mathcal{L}$*: Pick* $\alpha_\ell(t,s+1)$ *and* $\beta_\ell(t,s+1)$ *according to* (2.25),
   for $z_o = \log\left(1 + \frac{G_{\ell\ell}p_\ell(t,s)}{I_\ell(t,s)+V_\ell}\right)$.

7. $s = s + 1$

8. ***until*** $|\alpha_\ell(t,s) - \alpha_\ell(t,s-1)| \leq \epsilon$ *and* $|\beta_\ell(t,s) - \beta_\ell(t,s-1)| \leq \epsilon$, $\forall \ell \in \mathcal{L}$.

9. *Set* $\forall \ell \in \mathcal{L}$*:* $p_\ell(t)^{opt} := p_\ell(t,s)$, $\alpha_\ell^{opt}(t) := \alpha_\ell(t,s)$, $\beta_\ell^{opt}(t) := \beta_\ell(t,s)$,
   $\lambda_\ell^{opt}(t) := \lambda_\ell^{opt}$, $\left\{\gamma_{\ell k}^{opt}(t)\right\}_{k\neq\ell} = \left\{\gamma_{\ell k}^{opt}\right\}_{k\neq\ell}$, $\tilde{\boldsymbol{i}}_\ell^{opt}(t) := \tilde{\boldsymbol{i}}_\ell^{opt}$.

In the adaptive algorithms, either of the two versions of the core step algorithm may be used, whereas local convergence of each link and termination of the overall algorithm are determined as in the case of the corresponding batch algorithms. In case of the adaptive S.A. algorithm, if any link, at time slot t during the steady state of the system, is mapped via step 3 to a previous

solution where it was not activated (the resulting SINR was below machine precision), then this link remains idle for the present slot t, for the same reasons explained in the case of the batch S.A. algorithm.

2.3.5 Communication Overhead and Complexity

The distributed implementation of batch and adaptive algorithms developed for BPPC is possible in practice, as long as certain assumptions do hold. First of all, it is assumed that local link gain information is available at the transmitters of all links, i.e., $\mathrm{Tx}(\ell)$ has knowledge of $G_{\ell k}$ towards $\mathrm{Rx}(k)$, $\forall \ell, k \in \mathcal{L}$. Secondly, there are available low-rate control channels among the nodes of the network, so that each link can feed back information to all other links in the network.

We consider the case where relay nodes are not allowed to send packets back to the source, they may transmit to any other node including themselves, and the destination node acts as a sink. Then, there are $L = (N-1)^2$ possible links, where N is the number of nodes, and it is assumed that links interfering with link ℓ are all other links $k \neq \ell$, $\forall \ell \in \mathcal{L}$.

All algorithms for BPPC developed herein require that each link ℓ is able to calculate its differential backlog $D_\ell(t)$, at the beginning of each time slot t. Although $D_\ell(t)$ is considered local parameter for link ℓ, its calculation implies that the receiver of link $\ell = (i, j)$, $\mathrm{Rx}(\ell)$, feeds back to the transmitter $\mathrm{Tx}(\ell)$ its backlog $W_j(t)$, except for the case where the link is destined to the sink, i.e., $j = N$. Taking all possible links into account, the determination of differential backlogs requires $(N-1)(N-3)+1$ message exchanges among nodes, at the beginning of each slot. In cases where relays are allowed to transmit back to the source (where there are additionally $N-2$ links), the corresponding exchanges among nodes amount to $(N-1)(N-2)$. This is common for all algorithms developed for the BPPC problem.

In order for the optimization steps in the core step algorithms to be carried out in parallel across links, additional message exchanges among links are required over the control channels. Concerning core step 1 algorithm, according to step 5, each link ℓ transmits its dual variable $\gamma_{\ell k}$ to link k, and this holds for all its interfering links, $k \neq \ell$. In total, link ℓ has to transmit $L-1$ dual variables. The same holds for its auxiliary variables $\left\{\tilde{i}_{\ell k}\right\}_{k \neq \ell}$. Therefore, link ℓ transmits $2(L-1)$ variables per iteration of the algorithm. Taking all possible links into account, the communication overhead per iteration of the algorithm amounts to $2L(L-1)$ exchanges across

the network. In the worst case, depending on the operational setup, where each link can not estimate, but need to be passed around information $\tilde{G}_{k\ell}+\tilde{p}_k$ from all its interfering links $k \neq \ell$, required for steps 2 and 3, then, a total of $L(L-1)$ exchanges, accounting for all links, must be also encountered in the communication overhead per iteration. As long as core step 2 is concerned, links need to transmit, additionally, their penalty parameters to all their interfering links when these change, according to step 6 of the algorithm. Thus, extra $L(L-1)$ message exchanges must be taken into account, compare to core step 1.

Summarizing, the first version of core step algorithm requires $2L(L-1)$ exchanges among links in the most favorable case (depending on the operational setup), and $3L(L-1)$ in the worst case. The second version requires, accordingly, $3L(l-1)$, or $4L(L-1)$ exchanges per iteration of the algorithm. The communication overhead of the batch/adaptive high-SINR/S.A. algorithms can be evaluated only per iteration, which is equal to that determined by the respective core step algorithm employed.

The computational complexity of both core step algorithms is mainly determined by the method used for the unconstrained minimization with respect to $\left\{\tilde{\boldsymbol{i}}_\ell\right\}_{\ell\in\mathcal{L}}$ variables in steps (2.14) and (2.22), respectively. We used the damped Newton method as proved more efficient in practice than other quasi-Newton methods. The computation of the Newton step per link entails complexity $\mathcal{O}\left((L-1)^3\right)$, since the number of coefficients of $\tilde{\boldsymbol{i}}_\ell$ is $L-1$, for each ℓ. In addition, the damped Newton method involves function, gradient and Hessian evaluations of function in (2.14) (and (2.22)) with respect to $\tilde{\boldsymbol{i}}_\ell$, which are bounded roughly by $\mathcal{O}((L-1)^2)$ operations. The closed form update steps for each link are much cheaper, therefore the overall computational complexity per link and per iteration is bounded by $\mathcal{O}((L-1)^3)$. Taking all update steps carried out in parallel for all links, the worst-case computational complexity per iteration of the core step algorithms is roughly $\mathcal{O}(L^4)$.

Batch S.A. algorithm converges typically in up to 5 tightening steps per slot in our experiments, while, the total number of its core step iterations per slot is up to 2 times that of batch high-SINR algorithm. Therefore, the average computational complexity of both batch algorithms is again $\mathcal{O}(L^4)$, where in the big $\mathcal{O}$ notation the number of total core step iterations of the algorithms, per slot, is hidden. The adaptive algorithms have the same computational complexity order, per iteration, as their batch counterparts, due to the Newton method used in the core step algorithm. However, due to their warm re-start on past solutions, they require

significantly less iterations per slot, compare to the batch ones, which results in a corresponding reduction in their average complexity. This will be illustrated in the simulations section.

2.4 The Iteratively Weighted MMSE Approach for BPPC

The iteratively weighted MMSE approach in [50] can be utilized in our present multi-hop cross-layer networking context. In particular, it can be used to handle the NP-hard power control problem arising at the physical layer in the back-pressure power control policy. This is because the weighted sum-rate maximization problem is equivalent to a properly weighted sum-MSE minimization, in the sense that the two problems have identical global optimal solution, as proved in [50], for the general MIMO interfering broadcast channel. This equivalence holds also for the SISO interference channel, which models the wireless network in our context, viewed as a special case of the MIMO IBC channel. The iterative weighted-MMSE algorithm (WMMSE), proposed in [50], is proved therein to converge to at least a local optimal solution of the NP-hard weighted sum-rate maximization problem, by solving an iteratively weighted sum-MSE minimization problem. This algorithm can be used in our case in order to obtain a power allocation that corresponds to a stationary point of the differential backlog-weighted sum-rate maximization problem. Before presenting the WMMSE algorithm specialized in our context, let us give the main points of the iteratively WMMSE approach in [50], as they correspond to the special case of the SISO interference channel describing the wireless network we consider.

In the system model we consider in section 2.2 each node is equipped with a single antenna, thus each transmitter-receiver pair has a single transmit and receive antenna. Also, parameters $G_{k\ell}$ denote the power loss between $\mathrm{Tx}(k)$ and $\mathrm{Rx}(\ell)$, $\forall k, \ell \in \mathcal{L}$. The WMMSE approach in [50] involves complex channel coefficients between each transmitter $\mathrm{Tx}(k)$ and receiver $\mathrm{Rx}(\ell)$, denoted by $h_{k\ell}$. We may then generate $h_{k\ell}$ from the complex Gaussian distribution $\mathcal{CN}(0,1)$, but also scale them such that $|h_{k\ell}|^2 = G_{k\ell}$, $\forall \ell, k \in \mathcal{L}$, to be consistent with our context. The transmit and receive beamforming matrices used for the MIMO interfering broadcast channel in [50], become scalars in our case. Let, therefore, v_ℓ denote the complex gain used by the transmitter of link ℓ, and u_k the complex gain used by the receiver of link k. The positive semidefinite matrix variables, used also as optimization variables in [50], become positive scalar variables w_ℓ, used by the receiver of each link ℓ. Then, following the steps of the approach in [50]

in our problem, the differential backlog-weighted $\{w_\ell\}_{\ell\in\mathcal{L}}$ -weighted sum-MSE minimization problem, under per link power constraints, is described as

$$\min_{\{w_\ell,u_\ell,v_\ell\}_{\ell\in\mathcal{L}}} \sum_{\ell=1}^{L} D_\ell(t)\left(w_\ell e_\ell - \log(w_\ell)\right) \tag{2.26}$$

$$\text{subject to} \quad |v_\ell|^2 \le P_\ell \ , \quad \ell \in \mathcal{L}. \tag{2.27}$$

Here, e_ℓ is the mean squared estimation error for link ℓ, given by

$$e_\ell := |\overline{u}_\ell h_{\ell\ell} v_\ell - 1|^2 + \sum_{\substack{k=1\\k\neq\ell}}^{L} |u_\ell h_{k\ell} v_k|^2 + V_\ell |u_\ell|^2, \ \forall \ell \in \mathcal{L}, \tag{2.28}$$

where $\overline{u}_\ell$ denotes the complex conjugate, and V_ℓ denotes the background noise power at $\mathrm{Rx}(\ell)$.

The optimal w_ℓ and u_ℓ, for fixed v_ℓ, can be derived from the first order optimality condition applied to the objective (2.26), yielding

$$u_\ell^{opt} \equiv u_\ell^{mmse} = \frac{h_{\ell\ell} v_\ell}{\sum_{k=1}^{L} |h_{k\ell}|^2 |v_k|^2 + V_\ell}, \ \forall \ell \in \mathcal{L}. \tag{2.29}$$

$$w_\ell^{opt} = e_\ell^{-1}, \forall \ell \in \mathcal{L}. \tag{2.30}$$

Plugging the optimal u_ℓ^{opt} in the expression (2.28) for the mean squared error yields

$$e_\ell^{mmse} = 1 - \frac{|v_\ell|^2 |h_{\ell\ell}|^2}{\sum_{k=1}^{L} |h_{k\ell}|^2 |v_k|^2 + V_\ell}, \ \forall \ell \in \mathcal{L}. \tag{2.31}$$

Using (2.30) and (2.31) to simplify (2.26) in problem (2.26)–(2.27), results in the following optimization problem

$$\max_{\{v_\ell\}_{\ell\in\mathcal{L}}} \sum_{\ell=1}^{L} D_\ell(t) \log\left(1 - \frac{|h_{\ell\ell}|^2 |v_\ell|^2}{\sum_{k=1}^{L} |h_{k\ell}|^2 |v_k|^2 + V_\ell}\right)^{-1} \tag{2.32}$$

$$\text{subject to} \quad |v_\ell|^2 \le P_\ell \ , \quad \ell \in \mathcal{L}. \tag{2.33}$$

which is further equivalent to

$$\max_{\{v_\ell\}_{\ell\in\mathcal{L}}} \sum_{\ell=1}^{L} D_\ell(t) \log\left(1 + \frac{|h_{\ell\ell}|^2 |v_\ell|^2}{\sum_{\substack{k=1\\k\neq\ell}}^{L} |h_{k\ell}|^2 |v_k|^2 + V_\ell}\right) \tag{2.34}$$

$$\text{subject to} \quad |v_\ell|^2 \le P_\ell \ , \quad \ell \in \mathcal{L}. \tag{2.35}$$

Problem (2.34)–(2.35), upon a change of variables $p_\ell = |v_\ell|^2, \forall \ell \in \mathcal{L}$, and $G_{k\ell} = |h_{k\ell}|^2, \forall k, \ell \in \mathcal{L}$, coincides with our BPPC problem:

$$\max_{\{p_\ell\}_{\ell\in\mathcal{L}}} \sum_{\ell=1}^{L} D_\ell(t) \log\left(1 + \frac{G_{\ell\ell}p_\ell}{\sum_{\substack{k=1\\k\neq\ell}}^{L} G_{k\ell}p_k + V_\ell}\right) \tag{2.36}$$

$$\text{subject to} \quad p_\ell \leq P_\ell \ , \quad \ell \in \mathcal{L}. \tag{2.37}$$

2.4.1 The Iterative WMMSE Algorithm for BPPC

The WMMSE algorithm in [50] is an iterative algorithm that minimizes the weighted sum-MSE cost function in (2.26), using a block coordinate descent technique. The optimization problem (2.26)–(2.27) is in the space of $(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w})$ and it is convex with respect to each one of the variables, assuming the other two fixed. Therefore, the authors in [50] propose to optimize the constructed cost function with respect to only one of the variables each time, while keeping the other two fixed. The optimization with respect to each one of the variables is also decoupled across the links of the network, and it leads to closed form updates of the local variables of each link. Consequently, the algorithm is amenable to distributed implementation and has low computational complexity.

The update rule for the variable u_ℓ, for each link $\ell \in \mathcal{L}$, has been already derived, given by the expression (2.29). The update of the variable w_ℓ is given by (2.30), which can be further analyzed by rewriting the expression for the mean squared error e_ℓ as:

$$e_\ell^{mmse} = 1 - \frac{|v_\ell|^2|h_{\ell\ell}|^2}{\sum_{k=1}^{L}|h_{k\ell}|^2|v_k|^2 + V_\ell} \equiv 1 - \overline{u}_\ell h_{\ell\ell} v_\ell, \ \forall \ell \in \mathcal{L}. \tag{2.38}$$

Thus, the update for variable w_ℓ, $\forall \ell \in \mathcal{L}$, becomes

$$w_\ell^{opt} = (1 - \overline{u}_\ell h_{\ell\ell} v_\ell)^{-1} \ , \ \forall \ell \in \mathcal{L}, \tag{2.39}$$

requiring only the local variables of link ℓ.

The update of the transmit gain variables $\{v_\ell\}_{\ell\in\mathcal{L}}$ can also be decoupled across links, as long as the variables u_ℓ and w_ℓ are assumed fixed for each link $\ell \in \mathcal{L}$, since each w_ℓ is obtained by solving the following convex quadratic optimization problem

$$\min_{v_\ell} D_\ell(t) w_\ell |1 - \overline{u}_\ell h_{\ell\ell} v_\ell|^2 + \sum_{\substack{k=1 \\ k\neq\ell}}^{L} D_k(t) w_k |u_k|^2 |h_{\ell k}|^2 |v_\ell|^2 \tag{2.40}$$

$$\text{subject to} \quad |v_\ell|^2 \leq P_\ell \ , \quad \ell \in \mathcal{L}. \tag{2.41}$$

The associated local Lagrange function with the optimization problem (2.40)–(2.41), is given by

$$L_\ell\left(v_\ell, \lambda_\ell\right) = D_\ell(t) w_\ell |1 - \overline{u}_\ell h_{\ell\ell} v_\ell|^2 + \sum_{\substack{k=1 \\ k\neq\ell}}^{L} D_k(t) w_k |u_k|^2 |h_{\ell k}|^2 |v_\ell|^2 + \lambda_\ell \left(|v_\ell|^2 - P_\ell\right) \ , \ \forall \ell \in \mathcal{L}. \tag{2.42}$$

The first order optimality condition of (2.42) with respect to v_ℓ yields

$$v_\ell\left(\lambda_\ell\right) = \frac{D_\ell(t) w_\ell u_\ell \overline{h}_{\ell\ell}}{\sum_{k=1}^{L} D_k(t) w_k |u_k|^2 |h_{\ell k}|^2 + \lambda_\ell} \ , \ \forall \ell \in \mathcal{L}. \tag{2.43}$$

Plugging the optimal value of the lagrange multiplier λ_ℓ in (2.43) results in the final closed form expression for the optimal v_ℓ. The optimal λ_ℓ^* can be found by checking the associated power constraint. Thus, whether it holds that $|v_\ell(0)|^2 \leq P_\ell$, the optimal $\lambda_\ell^* = 0$, otherwise, λ_ℓ^* must be positive and satisfy the constraint $|v_\ell\left(\lambda_\ell^*\right)|^2 = P_\ell$. In the latter case, λ_ℓ^* can be found using an one dimensional search technique, such as the bisection method, since the function $v_\ell\left(\lambda_\ell\right)$ is decreasing in λ_ℓ, for $\lambda_\ell > 0$.

Alternatively, we can work in the real domain in our context, where all channel coefficients and design variables are real scalars. In this case $h_{\ell k}, \forall k, \ell \in \mathcal{L}$ denotes the real channel between $\mathrm{Tx}(\ell)$ and $\mathrm{Rx}(k)$, and can be taken equal to $\sqrt{G_{\ell k}}$. Similarly, the gains $h_{\ell k}^2$ can be substituted by the factors $G_{\ell k}$, $\forall k, \ell \in \mathcal{L}$. The complex conjugate operators and the norms of the complex scalar quantities can be removed. The problem of interest, in (2.26)–(2.27), now becomes

$$\min_{\{w_\ell, u_\ell, v_\ell\}_{\ell\in\mathcal{L}}} \sum_{\ell=1}^{L} D_\ell(t) \left(w_\ell e_\ell - \log(w_\ell)\right) \tag{2.44}$$

$$\text{subject to} \quad v_\ell^2 \leq P_\ell \ , \quad \ell \in \mathcal{L}. \tag{2.45}$$

The update expressions for the $\{w_\ell\}_{\ell\in\mathcal{L}}$ variables in (2.39), and for $\{u_\ell\}_{\ell\in\mathcal{L}}$ in (2.29), hold the same. The difference is in the case of the $\{v_\ell\}_{\ell\in\mathcal{L}}$ variables, where we can result in a closed form update without computing the optimal lagrange multiplier λ_ℓ^*, for each link ℓ. Avoiding the one dimensional search for λ_ℓ^* results in further computational efficiency. The solution for

v_ℓ is nevertheless the same. This can be easily seen by noting that the problem (2.40)–(2.41) in the real domain becomes

$$\min_{v_\ell} D_\ell(t) w_\ell \left(1 - u_\ell h_{\ell\ell} v_\ell\right)^2 + \sum_{\substack{k=1 \\ k\neq\ell}}^{L} D_k(t) w_k u_k^2 G_{\ell k} v_\ell^2 \tag{2.46}$$

$$\text{subject to} \quad v_\ell^2 \leq P_\ell \ , \quad \ell \in \mathcal{L}. \tag{2.47}$$

The formulation of problem (2.46)–(2.47) allows instead of considering the associated lagrange function, to solve, equivalently, directly with respect to v_ℓ, by taking the root of the gradient of the objective (2.46) with respect to v_ℓ, and projecting it into the interval $[0, \sqrt{P_\ell}]$, to respect the associated power constraint. This leads to the following update rule for v_ℓ

$$v_\ell^{opt} = \left[\frac{D_\ell(t) w_\ell u_\ell h_{\ell\ell}}{\sum_{k=1}^{L} D_k(t) w_k u_k^2 G_{\ell k}} \right]_0^{\sqrt{P_\ell}} \ , \ \forall \ell \in \mathcal{L}, \tag{2.48}$$

with the notation $[x]_a^b := \max\left(\min\left(x, b\right), a\right)$.

Testing the WMMSE algorithm for BPPC through simulation experiments in both complex and real domains, it turned out that it converges to identical solutions for any instance of the BPPC problem. We therefore adopt the version of the WMMSE algorithm in the real domain, due to its gain in computational efficiency and its lower complexity. The batch version of the WMMSE algorithm for BPPC, which solves approximately the original BPPC problem resulting at each scheduling slot, from scratch, can be summarized as

Algorithm 12. Distributed Batch WMMSE for BPPC

For each time slot $t \geq 1$:

1. *Calculate the differential backlogs $D_\ell(t)$, $\forall \ell \in \mathcal{L}$, according to*
$$D_\ell(t) := \begin{cases} \max\{0, W_i(t) - W_j(t)\}, & j \neq N \\ W_i(t), & j = N. \end{cases}$$
2. *Initialize $\{v_\ell\}_{\ell\in\mathcal{L}}$ such that $v_\ell = \sqrt{P_\ell}$;*
3. ***repeat***
4. $(w_\ell)' \leftarrow w_\ell$, $\forall \ell \in \mathcal{L}$;
5. $u_\ell \leftarrow \frac{h_{\ell\ell} v_\ell}{\sum_{k=1}^{L} G_{k\ell} v_k^2 + V_\ell}$, $\forall \ell \in \mathcal{L}$;

6. $w_\ell \leftarrow (1 - u_\ell h_{\ell\ell} v_\ell)^{-1}$, $\forall \ell \in \mathcal{L}$;

7. $v_\ell \leftarrow \left[\frac{D_\ell(t) w_\ell u_\ell h_{\ell\ell}}{\sum_{k=1}^{L} D_k(t) w_k u_k^2 G_{\ell k}} \right]_0^{\sqrt{P_\ell}}$, $\forall \ell \in \mathcal{L}$;

8. ***until*** $|\log(w_\ell) - \log((w_\ell)')| \leq \epsilon$, $\forall \ell \in \mathcal{L}$;

9. *set* $p_\ell(t) := v_\ell^2$, $\forall \ell \in \mathcal{L}$.

Let us note that, according to step 1, links destined to more congested nodes have zero differential backlog, therefore are not included in the objective function (2.26), sought to be minimized. Note that, otherwise, for links with $D_\ell(t) = 0$, the respective variables $\{w_\ell, u_\ell, v_\ell\}$ are boosted to the triple $\{1, 0, 0\}$, according to the update steps 5, 6, and 7, of the algorithm. Consequently, the algorithm may account only for the *scheduled* links, i.e., $\ell \in \mathcal{L}'$, where $\mathcal{L}' := \{\ell | D_\ell(t) > 0\}$. Therefore, in practice, the objective function in (2.26), is minimized with respect to the set of variables $\{w_\ell, u_\ell, v_\ell\}_{\ell \in \mathcal{L}'}$, where $\mathcal{L}' \subseteq \mathcal{L}$.

2.4.2 Distributed Implementation and Computational Complexity

The WMMSE algorithm specialized in our context is amenable to distributed implementation. The closed form update steps of the design variables $\{w_\ell, u_\ell, v_\ell\}_{\ell \in \mathcal{L}}$, have been shown, according to [50], to be decoupled across links, when for each variable update the other two are kept fixed. This allows for simultaneous update of the variables across the network. However, certain feedback requirements among links must be satisfied, and the same assumptions considered in case of the distributed S.A. algorithms must also hold here, in order for the algorithm to operate in a distributed fashion. Recall that these basic assumptions are that local link gain information is available at the transmitters of all links, i.e., $\mathrm{Tx}(\ell)$ has knowledge of $G_{\ell k}$ towards $\mathrm{Rx}(k)$, $\forall \ell, k \in \mathcal{L}$, and that there are available control channels among the nodes of the network, so that links can exchange information.

We next evaluate the communication overhead that WMMSE algorithm entails per iteration. The backlogs exchanges among nodes, for the determination of the differential backlog $D_\ell(t)$ of each link $\ell \in \mathcal{L}$, at the beginning of each time slot, according to step 1 of the WMMSE algorithm, are equal to that in case of the S.A. algorithms, since this is a common initialization step for all algorithms for BPPC. The update of variables $\{u_\ell, w_\ell\}$ in steps 5, 6, of the algorithm, require only local measurement of the total received signal, including background noise at $\mathrm{Rx}(\ell)$.

The v_ℓ update step 7 at $\mathrm{Tx}(\ell)$ of link ℓ requires knowledge of the information $D_k u_k^2 w_k$ for all its interfering links $k \neq \ell$. Thus, all links $k \neq \ell$ must send this information to $\mathrm{Tx}(\ell)$. For ease of comparison with the S.A. algorithm, we consider $D_\ell(t) u_\ell^2 w_\ell$ as local information for link ℓ, and therefore, we do not encounter the feedback of this information from $\mathrm{Rx}(\ell)$ to $\mathrm{Tx}(\ell)$ in the communication overhead. The simultaneous update of $\{v_\ell\}_{\ell \in \mathcal{L}}$, for all links, amounts to $L(L-1)$ exchanges across the network, per iteration of the algorithm. Note that the message exchanges among links per iteration of WMMSE amount to $\frac{1}{2}$ of the exchanges that S.A. requires via core step 1, per iteration, and to $\frac{1}{3}$ of the corresponding ones in case of the S.A. algorithm via core step 2.

Yet, in order for the algorithm to be fully decentralized, it is furthermore necessary that its convergence is achievable via local computation and communication among links. Any central synchronization for the termination of the iterates is undesirable. Moreover, a predetermined fixed number of iterations is not necessarily optimal in terms of solution accuracy, or requires experimentation for every specific setup considered, which is not practical. The termination criterion $|\sum_{k=1}^{L} \log(w_\ell) - \sum_{k=1}^{L} \log((w_\ell)')| \leq \epsilon$, proposed in [50], requires central computation and, therefore, is not suitable for our purpose. A local metric should be used instead, such as the variation of $\log(w_\ell)$, for each link ℓ. Another possible local metric is the variation of the local Lagrange function $L_\ell(w_\ell, u_\ell, v_\ell, \lambda_\ell)$ in (2.42), for each link ℓ. The iterates, then, should terminate only once all links reach convergence within a given tolerance, in terms of the adopted local metric. This can be accomplished by employing a consensus on-min-the-algorithm [10, 49] among links, in a corresponding manner as in the case of the distributed S.A. algorithms.

As long as the computational complexity of the algorithm is concerned, we need to encounter the update steps carried out by all links in one iteration of the algorithm. The update of $\boldsymbol{u}$, $\boldsymbol{v}$, and $\boldsymbol{w}$ variables yield computational complexity equal to $\mathcal{O}(L^2 + L^2 + L)$, or roughly $\mathcal{O}(L^2)$, per iteration of the algorithm. Recall that the per iteration computational complexity of the S.A.-based algorithms is $\mathcal{O}(L^4)$.

2.4.3 Distributed Adaptive WMMSE Algorithm for BPPC

Inspired by the development of the adaptive versions of the centralized and distributed successive convex approximation algorithms for BPPC, we develop here an analogous adaptive version of the batch WMMSE algorithm. The main motivation, towards this end, remains the same as in

the case of the S.A. approach; the differential backlogs can change dynamically at subsequent time slots, as packets are routed in the network, even when physical layer propagation conditions are fixed, or vary slowly with time. Thus, the differential backlog-weighted weighted sum-MSE cost function in 2.26 changes, and a new problem arises at each time slot. Although the computational complexity of the algorithm is low, an adaptive version, avoiding solving the problem from scratch at each time slot, is not meaningless. Furthermore, its initial form in [50] allows for a warm initialization of the algorithm, which is expected to improve further its average complexity.

Since, theoretically, the WMMSE approach converges to a stationary point of the differential backlog-weighted sum-rate maximization problem, as the S.A. approach does, we expect an analogous steady state of the system's evolution under control of the WMMSE algorithm, for stable setups, as the one under control of the S.A. algorithms. Then, in the adaptive version, WMMSE can be initialized according to the S.A. adaptive algorithms, exploiting this way the potential quasi-periodicity of the solution arising in stable setups, due to the push-pull fashion of the system's operation. Utilizing the same strategy for the initialization step used in the case of the S.A. algorithms, and using a window of W past solutions, large enough to capture the emerging period, the adaptive WMMSE algorithm can be summarized as

Algorithm 13. Distributed Adaptive WMMSE for BPPC

For each time slot $t \geq 1$:

1. *Calculate the differential backlogs $D_\ell(t)$, $\forall \ell \in \mathcal{L}$, according to*

$$D_\ell(t) := \begin{cases} \max\{0, W_i(t) - W_j(t)\}, & j \neq N \\ W_i(t), & j = N. \end{cases}$$

2. *Initialization:*

   For $t = 1$ set: $v_\ell = \sqrt{P_\ell}$, $\forall \ell \in \mathcal{L}$;

   For $t \in [2, W]$ set:

$$\tau_o = \arg \min_{\tau \in \{t-1,\ldots,1\}} |D_\ell(t) - D_\ell(\tau)|, \; v_\ell = \sqrt{p_\ell(t-\tau_o)}, \forall \ell \in \mathcal{L}.$$

   For $t \geq W + 1$ set:

$$\tau_o = \arg \min_{\tau \in \{t-1,\ldots,t-W\}} |D_\ell(t) - D_\ell(\tau)|, \; v_\ell = \sqrt{p_\ell(t-\tau_o)}, \forall \ell \in \mathcal{L}.$$

3. ***repeat***

4. $(w_\ell)' \leftarrow w_\ell$, $\forall \ell \in \mathcal{L}$;

5. $u_\ell \leftarrow \frac{h_{\ell\ell} v_\ell}{\sum_{k=1}^{L} G_{k\ell} v_k^2 + V_\ell}$, $\forall \ell \in \mathcal{L}$;

6. $w_\ell \leftarrow (1 - u_\ell h_{\ell\ell} v_\ell)^{-1}$, $\forall \ell \in \mathcal{L}$;

7. $v_\ell \leftarrow \left[\frac{D_\ell(t) w_\ell u_\ell h_{\ell\ell}}{\sum_{k=1}^{L} D_k(t) w_k u_k^2 G_{\ell k}} \right]_0^{\sqrt{P_\ell}}$, $\forall \ell \in \mathcal{L}$;

8. ***until*** $|\log(w_\ell) - \log((w_\ell)')| \leq \epsilon$, $\forall \ell \in \mathcal{L}$;

9. *set* $p_\ell(t) := v_\ell^2$, $\forall \ell \in \mathcal{L}$.

The per-iteration computational complexity of the adaptive WMMSE, as well as the communication overhead it entails per iteration, remain equal to the ones of its batch counterpart. However, as will be illustrated in the simulations section, it typically requires 1-3 iterations per problem instance during steady state, and for stable setups, reducing thus by far the average complexity. This renders the adaptive algorithm more efficient than the original one, and lighter in terms of the overall communication overhead per slot.

2.5 A Heuristic Scheduling Tool

Our experience from a rather comprehensive set of simulation experiments with the centralized S.A algorithms for the BPPC problem, is that for the specific setups considered in our context in section 1.8, where fixed physical layer propagation conditions and deterministic fixed-rate arrivals are assumed, the system converges, for stable rates, to a periodic behavior. In addition, due to no-listen-while-you-talk considerations, we observed that in the emerging periodic patterns, non-interfering subsets of links were eventually activated in subsequent slots. Consider now the case of two subsets of links, namely $\mathcal{M}$ and $\mathcal{N}$, with $\mathcal{M}$ inserting to node i and $\mathcal{N}$ departing from i. Recall that, according to the no-listen-while-you-talk constraint, node i is able to either receive or transmit packets. Consequently, in such a case, only one of the two candidate subsets is the winner. What we observed in our experiments with the centralized algorithms is that the optimization at the physical layer (concerning the S.A. approximation of the BPPC objective), in such cases, consistently favors the subset with the higher sum of its links' differential backlogs. Of course, this is only a result concluded from our simulations in

section 1.8, under the specific setups considered, and does not hold in general. However, it can be exploited whenever the same conditions, as the ones we considered, do hold. In simulations with the distributed algorithms developed herein, under these specific setups, this result may constitute a heuristic, but useful scheduling tool.

Based on the above idea, and given the specific fixed physical layer propagation conditions, scheduling decisions can be taken beforehand at each node of the network, except for the sink. This is possible under the necessary condition that each node (except for the destination which is unnecessary) is aware of the backlogs of all other nodes in the network, required to calculate the differential backlogs of all possible links inserting to it, or departing from it. This condition, however, in the general case where relays are allowed to send packets back to the source, is already satisfied, since, for the purpose of distributed implementation, each link ℓ is able to calculate its differential backlog $D_\ell(t)$, at each time slot t. To gain some insight, note that according to (2.3), for each link $\ell = (i, j)$, where $j \neq N$, its receiver, $\mathrm{Rx}(\ell) = j$, feeds back its backlog, $W_j(t)$, to the transmitter $\mathrm{Tx}(\ell) = i$. Since this holds for all possible links departing from node i, node i has knowledge of the backlogs $W_{j'}(t)$ of all other nodes $j' \neq i, N$. This carries over verbatim to all nodes in the network, apart from node N, which acts as a sink, and does not need such information. In case where the source node only transmits and links destined to the source do not exist, then, in order for the relays to be aware of the source backlog, source must feed back its backlog to them, at the beginning of each time slot. Since all nodes know on a per-slot basis all the backlogs in the network, each one can calculate the differential backlogs of all possible links inserting to, or, departing from it. Then, each node may schedule links according to their priorities (differential backlogs) and the no-listen-while-you-talk constraint, as follows

For each node $i \in \{\mathcal{N}\} \setminus N$:

- Calculate the differential backlogs $D_\ell(t)$, $\forall \ell : \mathrm{Tx}(\ell) = i$, according to
$$D_\ell(t) := \begin{cases} \max\{0, W_i(t) - W_j(t)\}, & j \neq N \\ W_i(t), & j = N. \end{cases}$$
- Calculate the differential backlogs $D_k(t)$, $\forall k : \mathrm{Rx}(k) = i$, according to
$D_k(t) := \max\{0, W_j(t) - W_i(t)\}, j \neq N.$
- If $\sum_{\ell:\mathrm{Tx}(\ell)=i} D_\ell(t) > \sum_{k:\mathrm{Rx}(k)=i} D_k(t)$,

set $D_k(t) = 0, \forall k : \mathrm{Rx}(k) = i$,

else

set $D_\ell(t) = 0$, $\forall \ell : \mathrm{Tx}(\ell) = i$.

The benefit from this scheduling scheme, whenever it can be safely employed without performance-loss, is that it improves the computational efficiency of the distributed algorithms, since it reduces the dimension of the network and, consequently, the total of the variables involved in the optimization problem. This will be illustrated in the simulations section. The above step can follow right after step 1 in the adaptive and batch S.A / high-SINR algorithms, as well as after step 1 in the batch / adaptive WMMSE algorithms. In our simulations, we will refer to this implementation of the aforementioned algorithms as mode 2, while the original implementation, where all links with positive differential backlogs are scheduled, will be referred to as mode 1.

2.6 Simulation results

Extensive simulation results are provided in this section concerning all distributed algorithms developed for the BPPC problem. Judicious simulations illustrate the merits of the proposed algorithms, while discussion of results and insights are also added. We begin with the core step algorithms in subsection 2.6.1, the successive convex approximation algorithms follow in subsection 2.6.2, and finally, WMMSE-based algorithms are visited in subsection 2.6.3, where comparison between the two approaches is also included.

2.6.1 Performance evaluation of distributed core step algorithms

In this section we examine the convergence properties and quality of approximation of the proposed distributed version of the core step of successive convex approximation of BPPC problem, through simulations. We compare the distributed solution with the centralized one proposed in chapter 1. We include both versions of the distributed core step algorithm. In the sequel, we refer to the version using a global constant penalty parameter as core step 1 algorithm, and the version using varying local penalty parameters as core step 2. The specific setup used in our simulations is described next.

• Scenario 1 - small network with moderate interference: We consider a small network comprising $N = 6$ nodes, randomly drawn from a uniform distribution over a 100×100 square. The resulting topology is shown in Fig. 2.1. The network is assumed fixed throughout simulations. The lower-left node is considered the source, and the upper-right node the destination. All intermediates nodes are relays. We also assume a single flow (source-destination pair), and deterministic fixed arrivals at the source. There are $L = 25$ possible links, since we assume that the destination node act as a sink and relays are not allowed to sent packets back to the source. We take direct and crosstalk power losses inversely proportional to d^4, where d denotes the distance between any transmitter and receiver. We also assume that each receiver and transmitter is equipped with a single antenna. Furthermore, it is assumed that nodes are not allowed to transmit and receive packets simultaneously. Then, the no-listen-while-you-talk scenario is modeled by setting $G_{k\ell}$ equal to $1/eps$, if $\mathrm{Rx}(\ell) = \mathrm{Tx}(k)$, where eps is machine precision. A spreading gain $G = 128$ is assumed to moderate interference. The background noise power at the receiver of each link ℓ is set to $V_\ell = 10^{-12}$. The transmit power budget for

each link ℓ is set to $P_\ell = 5$.

We simulate the network for 100 packet/control slots under control of the centralized Batch high-SINR algorithm in [34], which corresponds to setting $\alpha_\ell(t) = 1$ and $\beta_\ell(t) = 0$, for each link ℓ and time slot t. We assume deterministic arrivals at a rate of 9 packets/slot at the source. The resulting differential backlogs $\{D_\ell(t)\}_{\ell\in\mathcal{L}}$, at each time slot $t \in \{1, 100\}$, are then used to solve 100 corresponding problem instances via the distributed algorithms.

Parameter choices for the distributed core step algorithms are as follows. A constant step-size $\delta_s = \delta = 0.01$, $\forall s$ was used for the gradient step in (2.16) of core-step 1, and in (2.24) of core-step 2. For initialization, the dual parameters $\{\lambda_\ell\}_{\ell\in\mathcal{L}}$ and $\{\gamma_{\ell k}\}_{k\neq\ell}^{\ell\in\mathcal{L}}$ were set equal to 1, while we picked randomly auxiliary variables $\left\{\tilde{\boldsymbol{i}}_\ell\right\}_{\ell\in\mathcal{L}}$. For the termination criterion in both algorithms, each link keeps track of i) the norm of its residual vector $\mathbf{r}_\ell$, and ii) successive differences of the value of its local augmented Lagrangian. Both must drop under $\epsilon = 10^{-2}$ for the protocol to terminate. The reason is that in the case of core step 1 (global constant ρ) higher ρ places a large penalty on the violation of the equality constraints and the residual vectors $\mathbf{r}_\ell$, for all ℓ, are forced quickly to zero, however convergence towards an optimal (vis-a-vis admissible) solution can be slow, because the $\gamma_{\ell k}$ update in (2.15) is coarse in the earlier stages. Recall also, from the definition of the dual residual vector $\mathbf{d}_\ell$, that large values of ρ tend to produce large dual residuals, whereas both residuals must converge to zero for convergence of ADMoM. We used both these convergence metrics in the case also of core step 2, while the parameters in the varying scheme in (2.18) were set to $\mu = 10$ and $\tau = 2$.

Our experiments concerning the core step 1 algorithm verified that convergence of ADMoM to a solution of (2.6)–(2.7) is assured for any ρ, however the choice of ρ affects significantly the number of iterations required to drop below a given tolerance threshold in terms of the aforementioned convergence metrics, and slightly the value of (2.6) attained at termination. To appreciate this, we present indicative results for the 30^{th} time slot in Table 2.1. Solving the problem for various values of ρ, we examine its impact on convergence speed and the finally attained value of (2.6). Notice that, as ρ is increased from 0.002 to 5, there is an initial decrease in the required number of iterations, reaching a 'sweet spot' at $\rho \sim 0.1$, followed by an increase. This notch-type behavior is typical. Also note that the higher ρ is the (slightly) lower the final objective value of (2.6), because the quadratic regularizing term is more heavily weighted for a given tolerance ϵ.

On the other hand, core step 2 algorithm turned out to be less dependent on the initial choice of the penalty parameters $\{\rho_{\ell,o}\}_{\ell\in\mathcal{L}}$, in terms of both convergence speed and solution accuracy. Similar to the case of core step 1, we varied the initial value of $\rho_{\ell,o}$, common for all links ℓ, from 0.002 up to 5. The respective results obtained from core step 2, for the same problem instance, are also given in Table 2.1. It is clear that due to the varying scheme applied, core step 2 can handle even extreme values chosen for the initialization of the penalty parameters. Convergence is achieved faster compare to core step 1, and in a moderate number of iterations for all initial values considered. Notice also from Table 2.1 that core step 2 algorithm yields almost the same objective value in all cases. The final solution attained is not affected by the initial $\{\rho_{\ell,o}\}_{\ell\in\mathcal{L}}$, since the regularizing term can be differently weighted from iteration to iteration.

Our experience from a rather comprehensive set of simulations is that the proposed distributed algorithms consistently yield essentially the same solution as the centralized one - not only in terms of the set of activated links (whose SINR turns out above machine precision), but also in terms of numerical values of the corresponding SINRs, within a reasonable accuracy for the ϵ- tolerance used. As an illustration, Table 2.2 summarizes results for the 30^{th} time slot.

Convergence of the distributed algorithms is illustrated in Fig. 2.2 for core step 1, and in Fig. 2.3 for core step 2. Specifically, Fig. 2.2 (left), concerning core step 1, depicts the progress of the sum of augmented Lagrange functions for all links (left panels) versus iterations, and the average, over all links, norm of the primal residual vector $\mathbf{r}_\ell(s)$ and dual residual vector $\mathbf{d}_\ell(s)$ (right panels), for $\rho = 0.01$ (top), and $\rho = 0.1$ (bottom). The respective results for core step 2 algorithm are plotted in Fig. 2.3 (left), in corresponding panels. As can be seen, convergence is achieved even at a moderate number of iterations, for both algorithms. Fig. 2.2 (right) and Fig. 2.3 (right), for core-step 1 and core-step 2 respectively, depict the progress of the objective function in (2.6) versus iterations, together with the value of (2.6) attained by the centralized solution (left panels), for $\rho = 0.01$ (top), and $\rho = 0.1$ (bottom). For better visualization, the difference of the two curves is plotted in Fig. 2.2 (right) for core step 1 (right panels), and in Fig.2.3 (right) for core step 2 (right panels), for $\rho = 0.01$ (top), and $\rho = 0.1$ (bottom). Clearly, performance of core step 1 comes closer to the centralized one for $\rho = 0.01$, at the cost of a higher number of iterations till ϵ-convergence (notice the different scaling of the x-axes and y-axes). Instead, core step 2 reaches the same solution accuracy for the two cases of the parameters initialization, since achieves ϵ-convergence in almost the same number of iterations.

Finally, the value of the objective in (2.6) obtained from core step 1 and the centralized algorithm, for all 100 problem instances, is plotted in Fig. 2.4 (left-top), for $\rho = 0.01$. For better appreciation of the difference, Fig. 2.4 (left-bottom) plots the absolute value of the difference of the two curves in 2.4 (left-top). Respective results for $\rho = 0.1$, are shown in Fig. 2.4 (right). For the case of $\rho = 0.01$, the average number of iterations was 437, while for $\rho = 0.1$, it was 185. Core step 2 algorithm was also tested in the same experiment. In this case the parameters $\{\rho_{\ell,o}\}_{\ell\in\mathcal{L}}$ were initialized at 0.1. Fig. 2.5 (top) compares the objective value attained from core step 2 and the centralized algorithm, while the absolute value of the two curves is plotted in Fig. 2.5 (bottom). In this experiment the varying local penalty parameters version averaged 180 iterations per problem instance.

Simulations suggest that that the distributed core step algorithms keep up with their centralized counterpart. The choice of the constant penalty parameter ρ when using the standard form of ADMoM is particularly important in this context, as there appears to be a sweet spot minimizing the number of iterations without much loss in terms of accuracy. On the other hand, the variation using different varying penalty parameter for each link seems appealing in practice, since it does not require experimentation on the initialization of the penalty parameters for every specific setup considered. Convergence in a moderate number of iterations can be assured this way, but at the cost of higher communication overhead. Our experimentation also showed that tuning of ϵ, in case of both algorithms, and ρ, in case of core step 1, can be used to trade-off solution accuracy for convergence speed, realizing favorable trade-offs.

Table 2.1: Objective value and ♯ of iterations, for various ρ and initial $\{\rho_{\ell,o}\}_{\ell\in\mathcal{L}}$. Tolerance $\epsilon := 10^{-2}$. Objective value for Batch high-SINR: 97.6388.

ρ / $\{\rho_{\ell,o}\}_{\ell\in\mathcal{L}}$	0.002	0.003	0.01	0.05	0.1	0.5	1	5
core step 1: ♯ iterations	1058	732	318	226	106	311	463	1212
core step 1: objective value	97.712	97.7	97.673	97.636	97.523	97.493	97.427	97.076
core step 2: ♯ iterations	127	103	103	98	97	125	127	256
core step 2: objective value	97.583	97.564	97.58	97.573	97.573	97.576	97.575	97.654

Table 2.2: SINR in dB attained at dominant links: ρ, $\{\rho_{\ell,o}\}_{\ell\in\mathcal{L}} := 0.01$.

Algorithm	ℓ_1	ℓ_2	ℓ_3	ℓ_4	ℓ_5
Centralized	18.23	10.76	11.1	10.78	10.92
core step 1	18.27	10.8	11	10.85	10.87
core step 2	18.27	10.8	11	10.85	10.88

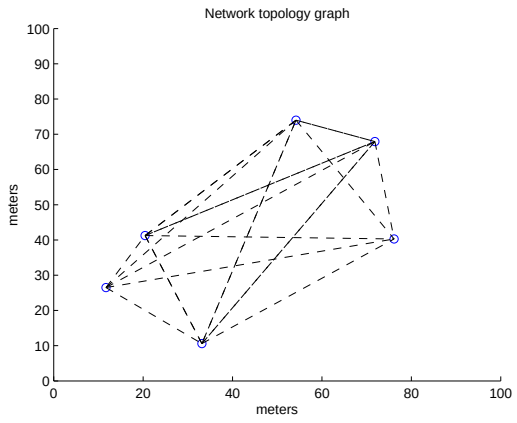

Figure 2.1: Network topology

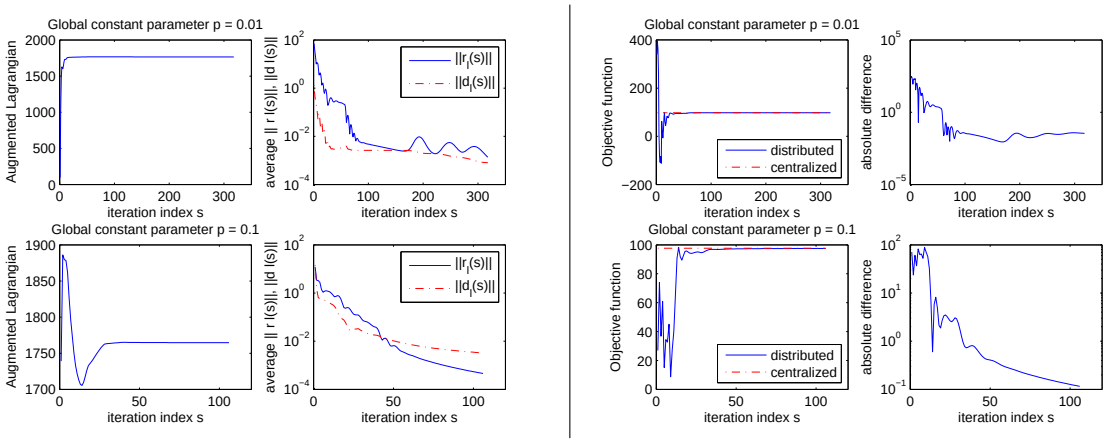

Figure 2.2: Core step 1: Left: Sum of Augmented Lagrange functions (left panels) and average norms of residual vectors (right panels) versus iterations, for $\rho = 0.01$ (top), and $\rho = 0.1$ (bottom). Right: Objective function (left panels) and approximation error (right panels) versus iterations, for $\rho = 0.01$ (top), and $\rho = 0.1$ (bottom).

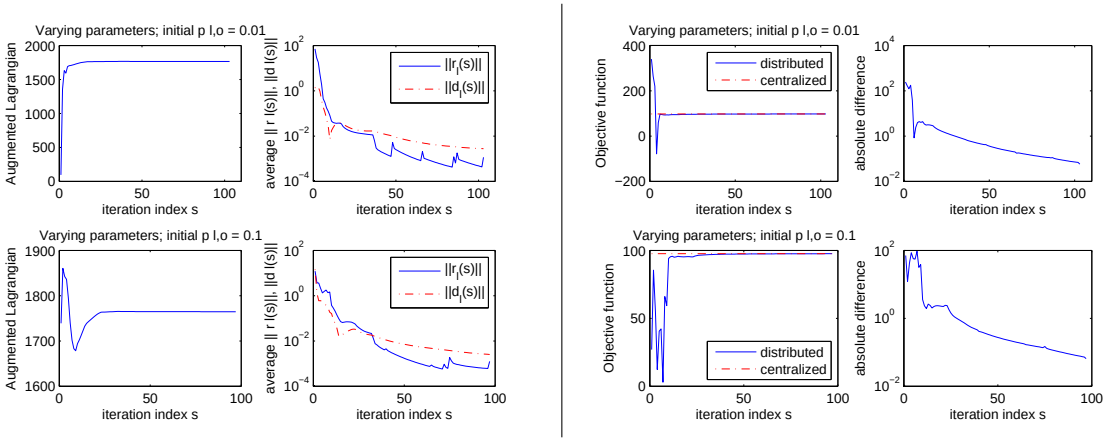

Figure 2.3: Core step 2: Left: Sum of Augmented Lagrange functions (left panels) and average norms of residual vectors (right panels) versus iterations, for $\rho = 0.01$ (top), and $\rho = 0.1$ (bottom). Right: Objective function (left panels) and approximation error (right panels) versus iterations, for $\rho = 0.01$ (top), and $\rho = 0.1$ (bottom).

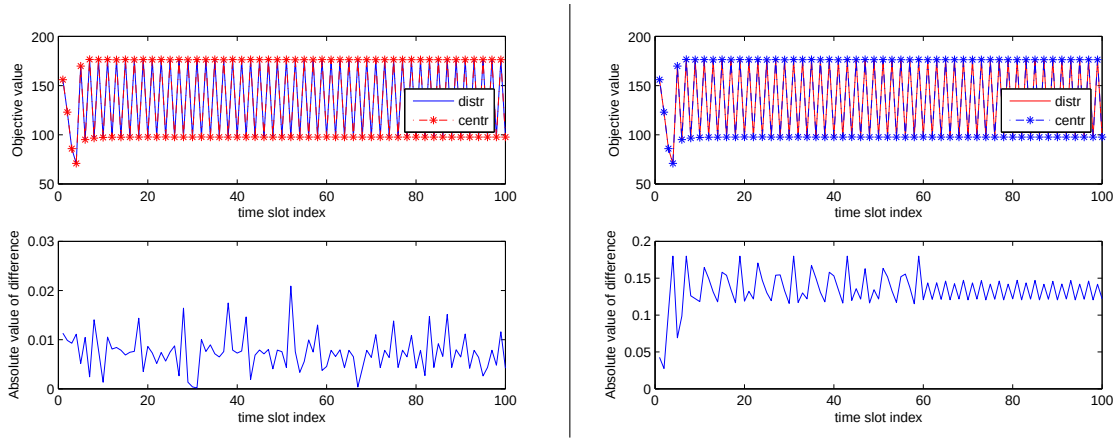

Figure 2.4: Comparison between core step 1 and centralized algorithm: Left: Objective value attained at 100 problem instances (top), absolute value of difference (bottom), $\rho = 0.01$. Right: Objective value attained at 100 problem instances (top), absolute value of difference (bottom), $\rho = 0.1$.

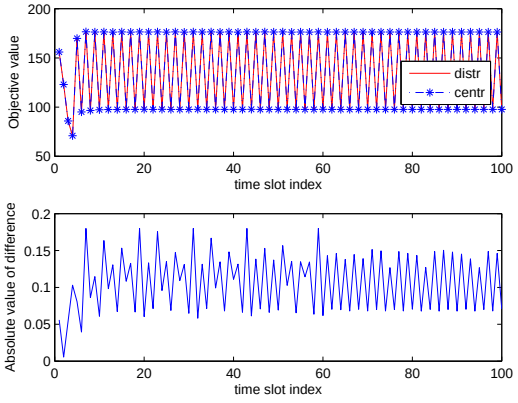

Figure 2.5: Comparison between core step 2 and centralized algorithm: Objective value attained at 100 problem instances (top), absolute value of difference (bottom), $\{\rho_{\ell,o}\}_{\ell\in\mathcal{L}} = 0.1$.

2.6.2 Performance evaluation of distributed Successive Convex Approximation algorithms

In this section we assess the performance of the distributed successive convex approximation (S.A.) - based algorithms for the BPPC problem. Simulation experiments have been performed in the three different scenarios of wireless networks considered in 1.8 for the performance evaluation of the centralized successive convex approximation algorithms for BPPC. However, the specific setup used in each scenario will be briefly described again. In our simulations we include the distributed batch high-SINR, batch S.A., as well as their adaptive versions. Both two versions of the core step algorithm are employed separately, yielding two variations for each batch and adaptive algorithm. Unless otherwise noted, all links with positive differential backlogs are scheduled at the beginning of each slot (mode 1), according to the initial form of the algorithms given in sections 2.3.3 and 2.3.4. Core step algorithms are employed as described in section 2.6.1, i.e., we use both convergence metrics for termination. In our simulations experiments we used $\rho = 0.1$ for core step 1 algorithm, while for core step 2 we initialized the penalty parameters at 1, and we used $\mu = 10$ and $\tau = 2$ for the varying scheme in 2.18.

• Scenario 1 - small network with moderate interference: The network we consider in this scenario is identical to the one used for the performance evaluation of the core step algorithms, which is described in section 2.6.1. The network comprises $N = 6$ nodes and there is a total of $L = 25$ possible links. The same assumptions hold here, i.e., we consider a single flow, fixed physical layer propagation conditions and deterministic fixed-rate arrivals at the source. We consider a no-listen-while-you-talk scenario which is modeled as described in the previous section. We use again a spreading gain $G = 128$ to model moderate interference. All parameters of the setup are taken identical to the ones of the setup in 2.6.1. The network is simulated under control of each algorithm considered for 100 packet/control slots.

We begin with the distributed batch high-SINR and batch S.A. algorithms. The tolerance used for the termination criterion of the core step algorithms was set to $\epsilon = 10^{-3}$, and so was the respective ϵ for the termination criterion in step 8 of the batch S.A. Simulations verified that batch distributed algorithms keep up with their centralized counterparts developed in 1; they managed to stabilize the system up to the same arrival rates, and we observed that the system operates in the same two-stage push-pull fashion, as under control of the centralized algorithms, exhibiting a periodic behavior for stable setups. In particular, batch high-SINR

managed to stabilize the system for arrival rates up to 9.7 packets per slot (pps), and did so in case either core step 1, or core step 2, is employed. Higher input loads could not be supported since queues become unstable. Fig. 2.6 depicts the evolution of the source and relay backlogs, the power allocation, and end-to-end throughput, for the arrival rate 9.7 pps (left) and 9.9 pps (right), in the case of using varying local penalty parameters (core step 2). The respective results obtained from the version employing core step 1, are plotted in Fig. 2.7 (left) for the stable setup at 9.7 pps, and in Fig. 2.7 (right) for the unstable setup at 9.9 pps. Note that node backlogs and end-to-end throughput settle in to a periodic pattern, with a period of two time slots. The weighted sum-rate objective is also periodic, and the respective plot is common for both variations. Fig. 2.14 (left) plots the objective value attained by batch high-SINR core step 2, together with the respective obtained by its adaptive version, and for the maximum stable rate (9.7 pps).

Similar to its centralized counterpart, the distributed batch S.A. algorithm (via both core step versions) managed to support higher stable throughput than the high-SINR one. In particular, batch S.A. can stabilize the system for arrival rates up to 10.4 pps, while queues become unstable at higher arrival rates, as can be easily visualized for the rate 10.8 pps. Fig. 2.8 (left) illustrates the evolution of all relevant quantities for the case of batch S.A. when core step 2 is employed (varying local penalty parameters), for the rate 10.4 pps, and Fig. 2.8 (right) plots the respective results for the rate 10.8 pps. Batch S.A. via core step 1 (constant penalty parameter) yields qualitatively similar results, as can be seen in Fig. 2.9. The plot of the objective value in 2.6 (common for the two batch S.A. versions) attained by batch S.A. core step 2 is depicted in 2.15 (left), together with that obtained from its adaptive counterpart. As illustrated for the stable setup, node backlogs, end-to-end throughput, as well as the weighted sum-rate attained, converge shortly (after 5 slots) to a periodic pattern and the average throughput is equal to the incoming traffic at the source. The emerging period is 2 time slots. It is worth noting here that throughout the steady state, in case of both high-SINR and S.A. algorithms, two non-interfering subsets of links are activated at subsequent time slots. In the one stage the links from source to all nodes are activated, and in the other stage, links from source and relays to destination are activated. While all other relevant quantities are periodic, the resulting power allocation may present quasi-periodicity (with respect to the values of link powers), as is the case when using varying local penalty parameters. For unstable setups, the average throughput is lower than

the incoming traffic, as can be discerned in Fig. 2.6 (right) and Fig. 2.7 (right).

The adaptive algorithms are expected to have the same performance with their batch counterparts in terms of maximum stable throughput attained, but to outperform in terms of average complexity, for stable setups, due to their warm re-start. This turned out to hold true. In their case, we used a window of $W = 5$ slots for initialization. We set the tolerance equal to $\epsilon = 10^{-3}$ for the termination criterion of core step algorithms, and in step 8 of adaptive S.A. algorithm, during the transitional phase of the system, and equal to 10^{-2}, throughout the steady state, in order to improve further the efficiency of the algorithms. The rest parameters of the adaptive high-SINR/S.A., and of the core step algorithms employed, are the same ones used in the batch algorithms. Adaptive high-SINR attains the same stable throughput as its batch counterpart. Results concerning the stable setup and the varying penalty parameters case (core step 2) are illustrated in Fig. 2.10 (left) for the rate 9.7 pps, while Fig. 2.10 (right) plots the results for rate 9.9 pps. The respective plots for the global penalty parameter case (core step 1) are given in 2.11 (left) for 9.7 pps, and in Fig. 2.11 (right) for 9.9 pps. Likewise, adaptive S.A. algorithm keeps up with its batch counterpart, as shown in Fig. 2.12 for the case of core step 2 used (compare to Fig. 2.8), and in Fig. 2.13 for the case of core step 1 (compare to Fig. 2.9). The weighted sum-rate obtained at each time slot by the adaptive high-SINR/S.A. algorithm is plotted together with the objective attained by its batch counterpart, for the stable setup, in Fig. 2.14 (left), for the adaptive high-SINR (core step 2), and in Fig. 2.15 (left) for the adaptive S.A. (core step 2).

Our simulations verified that the system under control of all adaptive algorithms (high-SINR, S.A.) operates in the same two-stage push-pull fashion as does in the case of the batch ones, and exhibits a similar periodic behavior with two time slots period. Notice, also, that although the same maximum stable throughput is attained by each adaptive algorithm and its batch counterpart, the resulting power allocations do not coincide. The same holds also when comparing the two variations of each batch or adaptive algorithm (via core step 1 and 2). First of all, power allocations need not coincide, because the underlying problem does not have a unique solution in general, as explained in 1.8. Nevertheless, from our simulations it turned out that all versions of each high-SINR, or S.A. algorithm, activate the same subsets of links in corresponding time slots. However, for each pair of algorithms under comparison, the differential backlogs of links resulting at each time slot from one algorithm do not coincide precisely with

the corresponding ones resulting from the other algorithm, but are close within $\epsilon-$ accuracy. This is due to the different progress of each algorithm, affected by the number of total iterations per slot, till $\epsilon-$ convergence, and by the specific penalty parameters used in each iteration. That implies that the underlying problems resulting at each time slot for the two algorithms under comparison are not identical, and therefore, the power allocations they yield are different. The lower accuracy used, during steady state, in the case of adaptive algorithms compared to their batch counterparts, should be taken also into account. Comparing the instantaneous weighted sum-rate obtained from batch S.A. (core step 2) with the respective one from adaptive S.A. (core step 2), as shown in Fig. 2.15 (right), it is clear that the two algorithms do not yield exactly the same solution, but very close ones, within $\epsilon-$ accuracy, which justifies, furthermore, the different resulting power allocations. The same also can be concluded from Fig. 2.14, concerning the respective high-SINR algorithms. Numerical results concerning the maximum stable average throughput and the average weighted sum-rate attained by each algorithm, for all scenarios considered, are gathered in Table 2.3. Notice that loss in accuracy with resect to the average weighted sum-rate attained by the adaptive algorithms, compare to their batch counterparts, is not significant.

The transitional phase until the system settles in to a periodic steady state is up to 10 time slots for the high-SINR algorithms, while for the S.A. algorithms is 5 time slots. The advantage of the adaptive versions of the algorithms is their remarkably reduced complexity compare to their batch counterparts. Table 2.4 summarizes, for each algorithm separately, the average run-time, average number of total iterations of core steps, and average number of convex approximations of the objective (for S.A. algorithms), per slot, during transitional phase and steady state. Notice that the adaptive high-SINR algorithms reduce the average run-time during steady state by 2 orders of magnitude relative to their batch counterparts, while adaptive S.A. algorithms do so by 3 orders of magnitude. When a higher accuracy of $\epsilon = 10^{-3}$ is required also during steady state, the complexity advantage of the adaptive versions of all algorithms over their batch counterparts remains in the same order of magnitude. In this case, adaptive algorithms (both high-SINR and S.A.) require only a few more iterations for convergence within $\epsilon-$ accuracy, and their average run-time during steady state is typically doubled. It is also typical that the adaptive S.A. algorithms do not tighten further the link rates lower bounds during steady state, but remain at the operating point of the previously visited

state of the system at which they are initialized, whatever the $\epsilon-$ tolerance for convergence.

• Scenario 2 - small network with strong interference: In this scenario we consider the same network as that in scenario 1, with the difference that here we use a spreading gain equal to $G = 8$ instead of 128, in order to increase the interference level. All other parameters of the setup are identical to those of scenario 1.

Our simulations showed that batch and adaptive high-SINR algorithms result in the same two-stage push-pull evolution of the system, activating the two subsets of links described in scenario 1 at subsequent time slots. However, due to stronger interference, the network can be stabilized only for arrival rates up to 2.4 pps. We plot simulation results illustrating the evolution of the system only for the case of adaptive high-SINR via core step 2 algorithm, as all four versions have qualitatively similar performance. Instead, we provide numerical results concerning maximum stable throughput, and averaged weighted sum-rate attained by all algorithms in Table 2.3, while complexity results are summarized in Table 2.4. Fig. 2.16 (left) illustrates the results for rate 2.4 pps, and Fig. 2.16 (right) for rate 2.6 pps. The objective value attained per slot is plotted in Fig. 2.17 (left), for the stable setup.

The successive convex approximation algorithms in this scenario managed to support arrival rates up to 7.8 pps. The system, under control of all S.A. algorithms, converges to a steady state after a transitional phase of about 30 time slots. During steady state only one link is activated: source transmits packets to the destination node, at maximum power. However, the system can not support higher input loads and source backlog grows to infinity for rate 7.9, and beyond. The same behavior of the system was observed under control of the centralized S.A. algorithms, however, due to the longer transient of the source backlog in that case, the stable setup at 7.5 pps was depicted, for ease of visualization. The evolution of the network under control of the adaptive S.A. algorithm (core step 2) is illustrated in Fig. 2.18 for the rate 7.8 pps (left), and for 7.9 pps (right). The objective value attained for the stable setup is plotted in Fig. 2.17 (right). Numerical results for all variations of the distributed algorithms are gathered in Tables 2.3 and 2.4. As can be seen in Table 2.4, the adaptive algorithms keep their advantage in complexity over their batch counterparts. Notice also, from the results in Table 2.3, that the successive convex approximation approach yields significantly higher stable average throughput and average weighted sum-rate than the high-SINR one.

• Scenario 3 - larger network, moderate interference: We consider a larger network in this scenario,

consisting of $N = 12$ nodes and $L = 121$ possible links. We keep the interference level moderate, as in scenario 1, by using a spreading gain $G = 128$. The setup is otherwise identical to the one considered so far.

Under control of both high-SINR and S.A. algorithms, the network in this scenario exhibits the same periodic behavior as the one observed in scenario 1. The system likewise evolves in two stages during steady state: all nodes receive packets from the source during the one stage, and all nodes transmit packets to the destination, during the next. The maximum stable throughput that high-SINR algorithms can attain is 12.6 pps. As an illustration, Fig.2.19 (left) plots the results for the rate 12.6 pps obtained from adaptive high-SINR core step 2, and Fig. 2.19 (right) the respective results for 12.7 pps. The S.A. algorithms proved again much better, since the higher rate that they can support is 15.7 pps. Fig. 2.20 depicts the results obtained from adaptive S.A. core step 2 algorithm, for the rate 15.7 pp (left) and 16.1 pps (right). The weighted sum-rate attained by the aforementioned algorithms, for their corresponding stable setups, is periodic with a period of two time slots, as all other relevant quantities. We skip the associated plots for brevity, but provide instead the average value of the objectives in Table 2.3. Complexity results for all versions of the high-SINR and S.A. algorithms are gathered in Table 2.5. Notice that the adaptive versions of all algorithms reduce the average complexity during steady state by three orders of magnitude relative to the batch versions, and that variations with local varying penalty parameters outperform in terms of computational efficiency and average run-time.

Our simulation experiments concerning the distributed high-SINR and S.A. algorithms show that the distributed algorithms keep up with their centralized counterparts, in all scenarios considered. It has been also shown, as in the centralized implementation, that the successive convex approximation strategy delivers manifold improvement in end-to-end throughput and in attainable weighted sum-rate, relative to the high-SINR approximation. Moreover, the distributed adaptive algorithms reproduce the results of their batch counterparts, at far lower complexity in stable setups. Comparing the two types of the proposed algorithms, resulting from the two versions of the core step algorithm employed, with respect to average complexity, based on the results provided in Tables 2.4 and 2.5, we can conclude that using varying local penalty parameters for the ADMoM method used in the core step, is favorable in case of large scale problems, as complexity results concerning the larger network suggest. The same conclusion

holds in case of stronger interference scenarios for the S.A. approach, as results show for the network considered in scenario 2.

We have also examined the performance of the algorithms when the heuristic scheduling tool in section 2.5 is applied. Our simulations verified that when scheduling decisions are taken by the nodes of the network, at the beginning of each time slot, in our specific setups, the efficiency of the distributed algorithms is improved significantly, with no performance loss. In particular, it reduces the average complexity of the batch algorithms, in terms of both iterations required (until convergence within the same accuracy used in mode 1 implementation of the algorithms) and of run-time per slot, typically by a factor of 4, in all scenarios considered. However, this gain widens with the size of the network; in case of the larger network in scenario 3, the corresponding reduction amounts to an order of magnitude. Concerning the adaptive versions, this scheduling scheme affects mainly their average complexity during the transitional phase of the system evolution, where it results in a similar reduction of the average run-time, as the one in case of the batch algorithms. Numerical results concerning the average throughput and average weighted sum-rate attained this way, are summarized in Table 2.7, in subsection 2.6.3, for all scenarios considered.

Table 2.3: Maximum stable throughput and average weighted sum-rate attained by all algorithms in all scenarios. (1):= core step 1, (2):= core step 2, w.s.r. := weighted sum-rate.

Scenario 1	Batch high-SINR (1) / (2)	Adapt. high-SINR (1) / (2)	Batch S.A. (1) / (2)	Adapt. S.A. (1) / (2)
Max stable rate	9.7 / 9.7	9.7 / 9.7	10.4 / 10.4	10.4 / 10.4
Avg. w.s.r	151.32 / 151.32	151.31/ 151.31	184.46 / 184.46	184.45 / 184.45
Scenario 2				
Max stable rate	2.4 / 2.4	2.4 / 2.4	7.8 / 7.8	7.8 / 7.8
Avg. w.s.r.	0.95 / 0.96	0.95 / 0.96	107.48 / 107.48	107.47 / 107.47
Scenario 3				
Max stable rate	12.6 / 12.6	12.6 / 12.6	15.7 / 15.7	15.7 / 15.7
Avg. w.s.r.	276.19 / 276.21	276.18 / 276.2	454.31 / 454.31	454.3 / 453.3

Table 2.4: Average complexity of all algorithms for respective stable setups in scenarios 1 and 2. (1) := core step 1, (2): core step 2.

Scenario 1	Batch high-SINR (1) / (2)	Adapt. high-SINR (1) / (2)	Batch S.A. (1) / (2)	Adapt. S.A. (1) / (2)
Trans. phase: avg. time (sec)	56 / 60	30 / 32	62 / 73	58 / 61
Steady state: avg. time (sec)	56 / 60	0.33 / 0.2	62 / 73	0.19 / 0.075
Trans. phase: avg. ♯ iter.	263 / 298	158 / 173	329 / 420	305 / 360
Steady state: avg. ♯ iter.	263 / 298	2.8 / 1.6	329 / 420	2.77 / 1
Trans. state: avg. ♯ approx.	-	-	3.98 / 3.98	3.6 / 3.6
Steady state: avg. ♯ approx.	-	-	3.98 / 3.98	1 / 1
Scenario 2				
Trans. phase: avg. time (sec)	53 / 58	28 / 31	35 / 32	38 / 31
Steady state: avg. time (sec)	53 / 58	0.14 / 0.12	35 / 32	0.0065 / 0.0065
Trans. phase: avg. ♯ iter.	255 / 281	162 / 184	283 / 267	314 / 244
Steady state: avg. ♯ iter.	255 / 281	1.2 / 1	283 / 267	1 / 1
Trans. state: avg. ♯ approx.	-	-	5.35 / 5.34	5 / 5
Steady state: avg. ♯ approx.	-	-	5.35 / 5.34	1 / 1

Table 2.5: Average complexity of all algorithms for respective stable setups in scenario 3. (1) := core step 1, (2): core step 2.

Scenario 3	Batch high-SINR (1) / (2)	Adapt. high-SINR (1) / (2)	Batch S.A. (1) / (2)	Adapt. S.A. (1) / (2)
Trans. phase: avg. time (sec)	216 / 210	120 / 85	264 / 245	254 / 242
Steady state: avg. time (sec)	216 / 210	0.65 / 0.57	264 / 245	0.44 / 0.24
Trans. phase: avg. ♯ iter.	454 / 454	278 / 200	635 / 610	615 / 605
Steady state: avg. ♯ iter.	454 / 454	2.1 / 2	635 / 610	2.9 / 1.2
Trans. state: avg. ♯ approx.	-	-	4.46 / 4.46	4.12 / 4.12
Steady state: avg. ♯ approx.	-	-	4.46 / 4.46	1 / 1

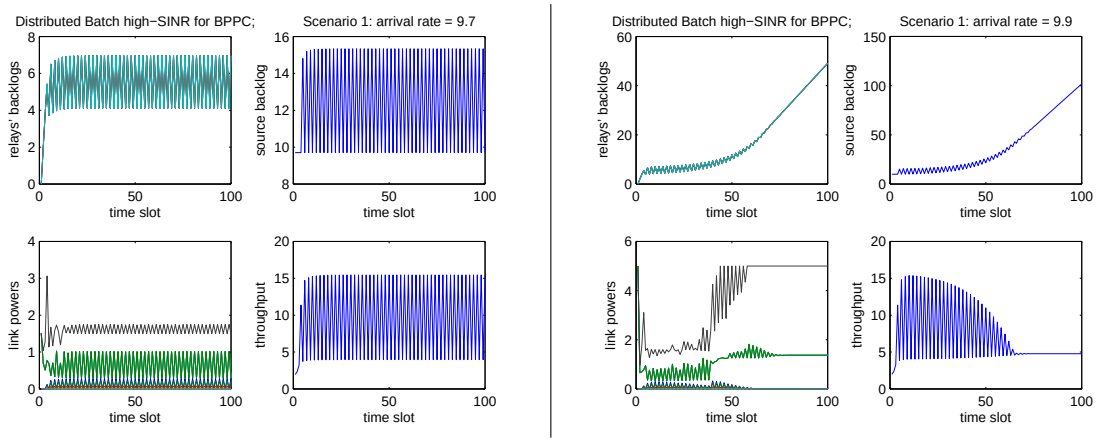

Figure 2.6: Scenario 1: Distributed Batch high-SINR for BPPC, varying local penalty parameters version (core step 2), arrival rate = 9.7 pps (left) and 9.9 pps (right).

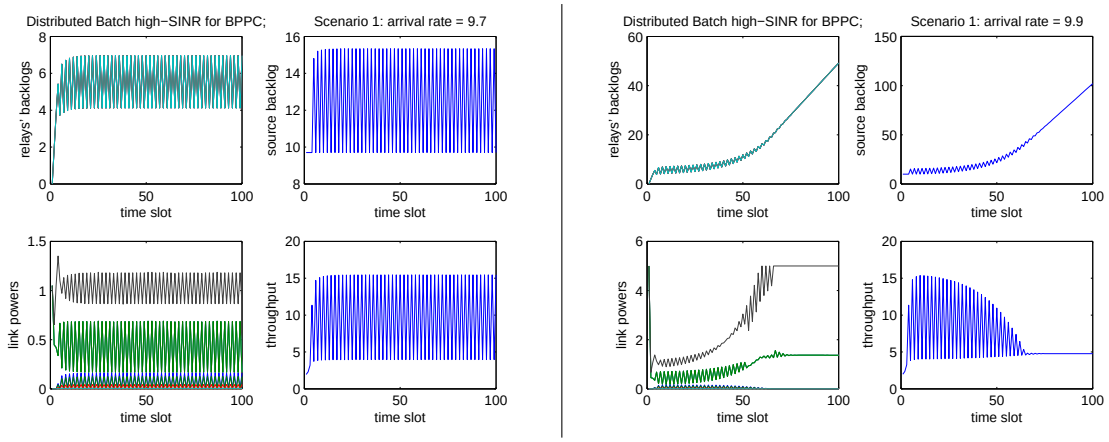

Figure 2.7: Scenario 1: Distributed Batch high-SINR for BPPC, global constant penalty parameter version (core step 1), arrival rate = 9.7 pps (left) and 9.9 pps (right).

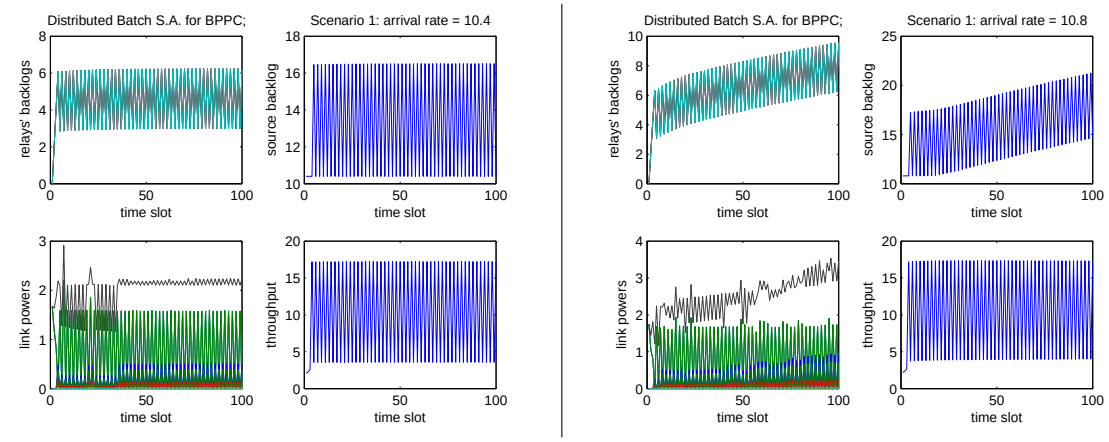

Figure 2.8: Scenario 1: Distributed Batch S.A. for BPPC, varying local penalty parameters version (core step 2), arrival rate = 10.4 pps (left) and 10.8 pps (right).

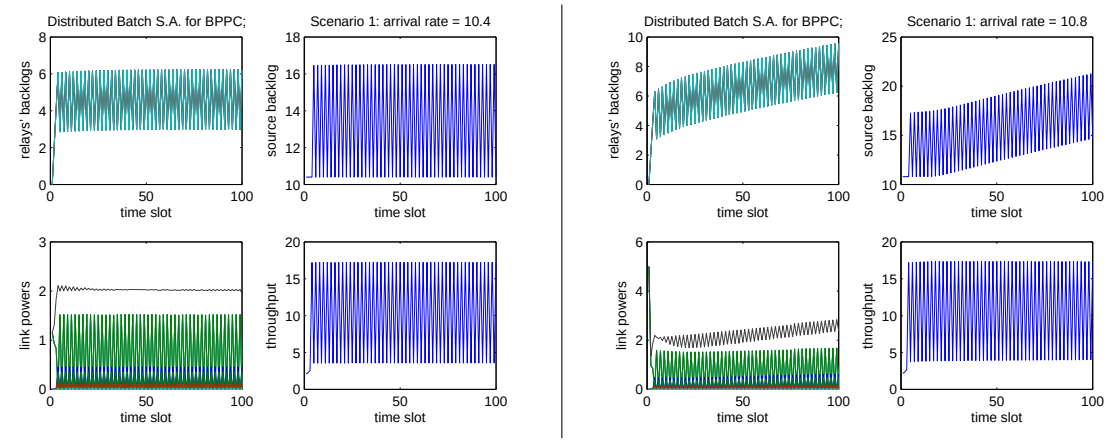

Figure 2.9: Scenario 1: Distributed Batch S.A. for BPPC, global constant penalty parameter version (core step 1), arrival rate = 10.4 pps (left) and 10.8 pps (right).

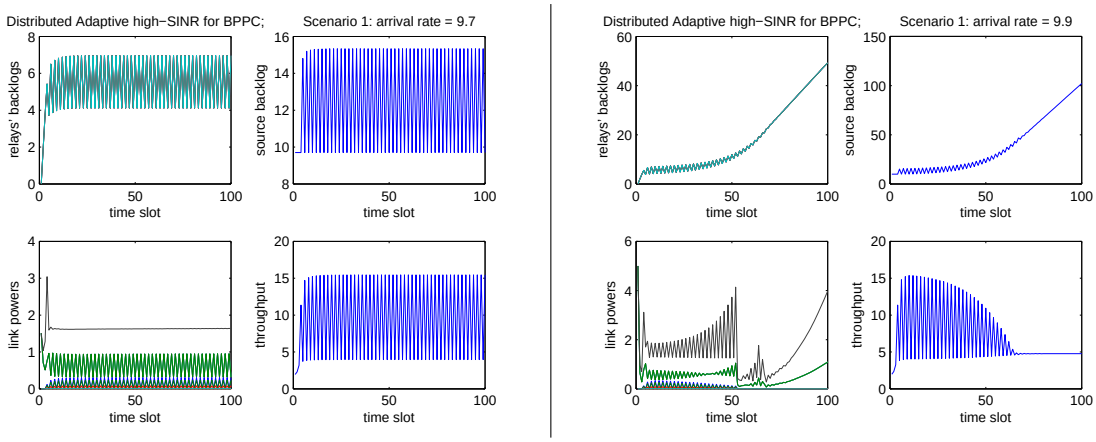

Figure 2.10: Scenario 1: Distributed Adaptive high-SINR for BPPC, varying local penalty parameters version (core step 2), arrival rate = 9.7 pps (left) and 9.9 pps (right).

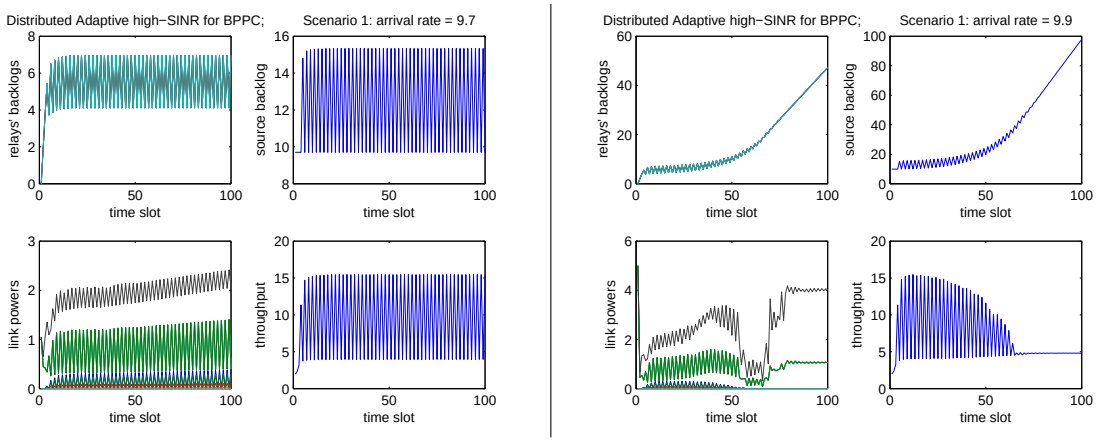

Figure 2.11: Scenario 1: Distributed Adaptive high-SINR for BPPC, global constant penalty parameter version (core step 1), arrival rate = 9.7 pps (left) and 9.9 pps (right).

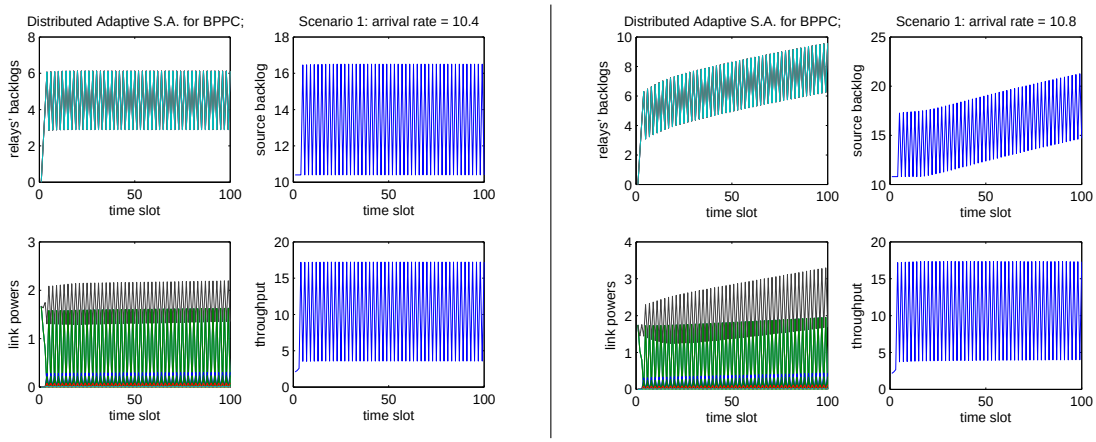

Figure 2.12: Scenario 1: Distributed Adaptive S.A. for BPPC, varying local penalty parameters version (core step 2), arrival rate = 10.4 pps (left) and 10.8 pps (right).

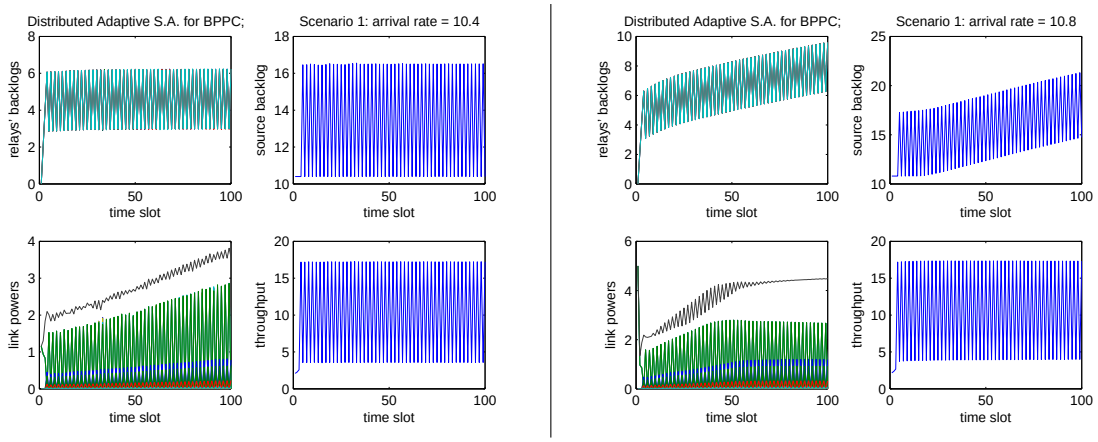

Figure 2.13: Scenario 1: Distributed Adaptive S.A. for BPPC, global constant penalty parameter version (core step 1), arrival rate = 10.4 pps (left) and 10.8 pps (right).

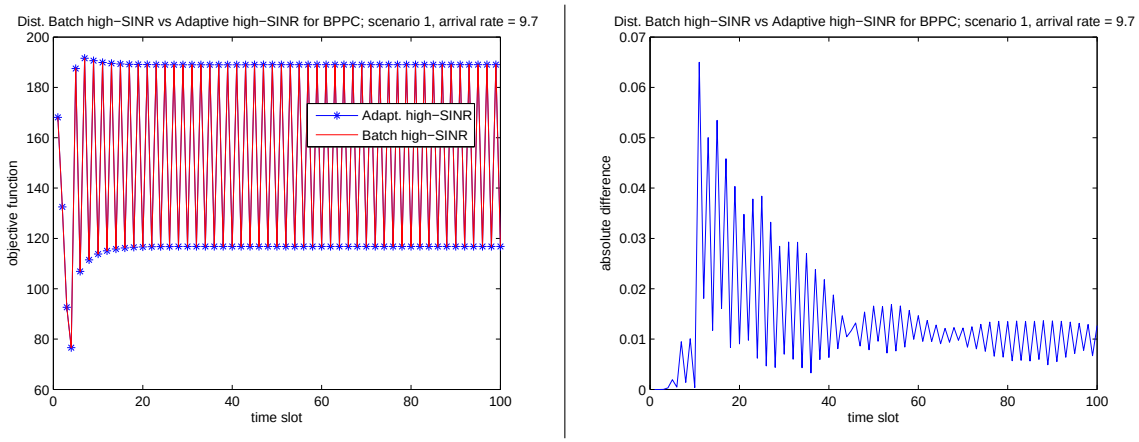

Figure 2.14: Scenario 1, rate 9.7 pps: Comparison between Batch high-SINR and Adaptive high-SINR (core step 2) w.r.t. weighted sum-rate (left), absolute value of difference of the objectives attained (right).

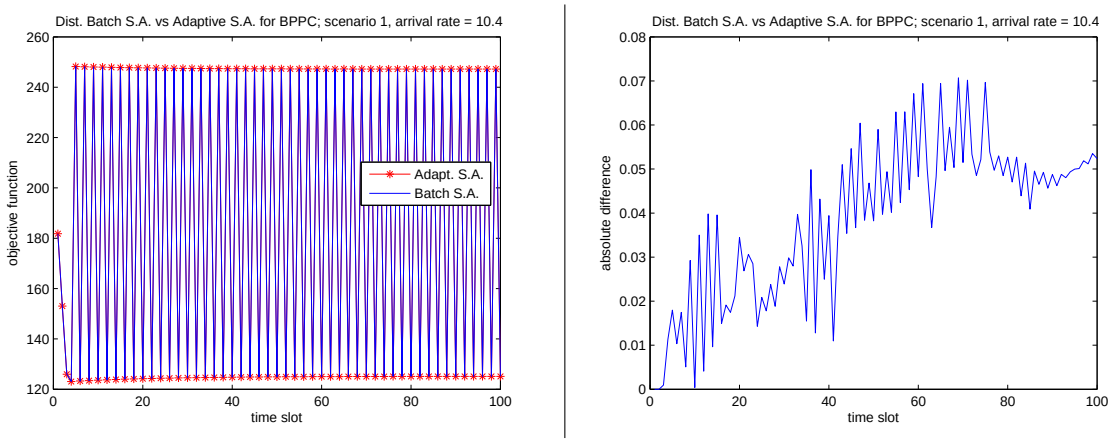

Figure 2.15: Scenario 1, rate 10.4 pps: Comparison between Batch S.A. and Adaptive S.A. (core step 2) w.r.t. weighted sum-rate (left), absolute value of difference of the objectives attained (right).

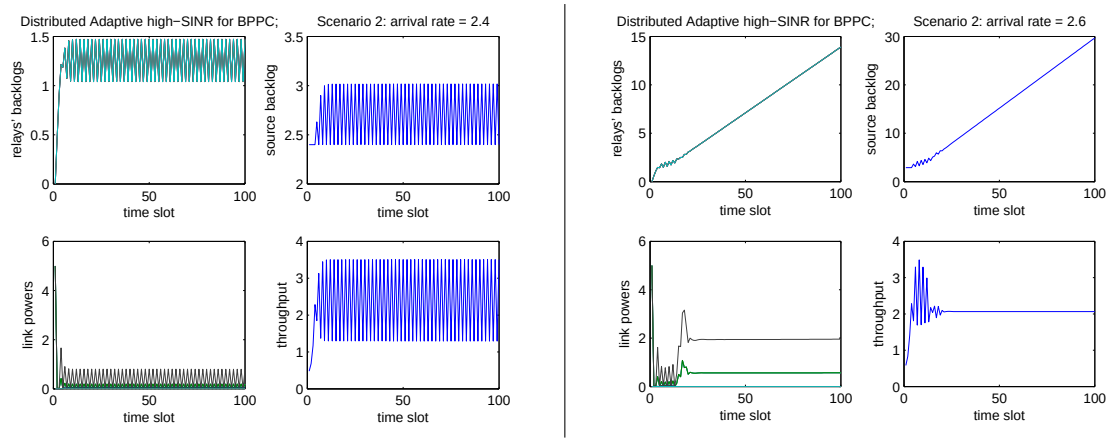

Figure 2.16: Scenario 2: Distributed Adaptive high-SINR for BPPC, varying local penalty parameters version (core step 2), arrival rate = 2.4 pps (left) and 2.6 pps (right).

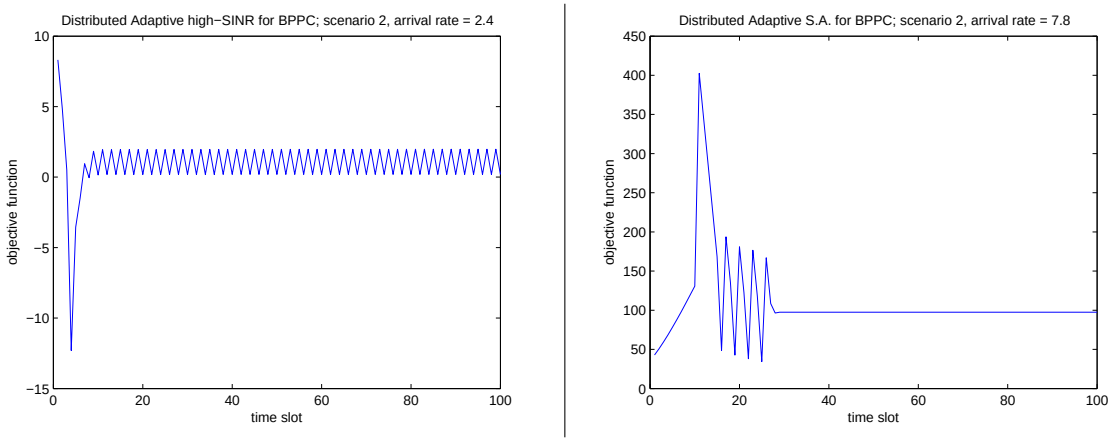

Figure 2.17: Scenario 2: Weighted sum-rate objective attained by distributed Adaptive high-SINR for BPPC (core step 2) at 2.4 pps (left), and by Adaptive S.A. for BPPC (core step 2) at 7.8 pps (right).

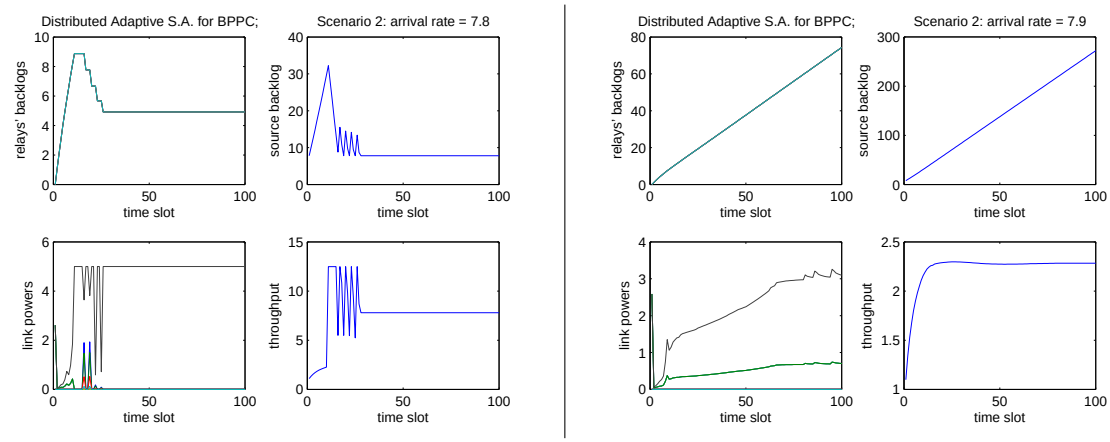

Figure 2.18: Scenario 2: Distributed Adaptive S.A. for BPPC, varying local penalty parameters version (core step 2), arrival rate = 7.8 pps (left) and 7.9 pps (right).

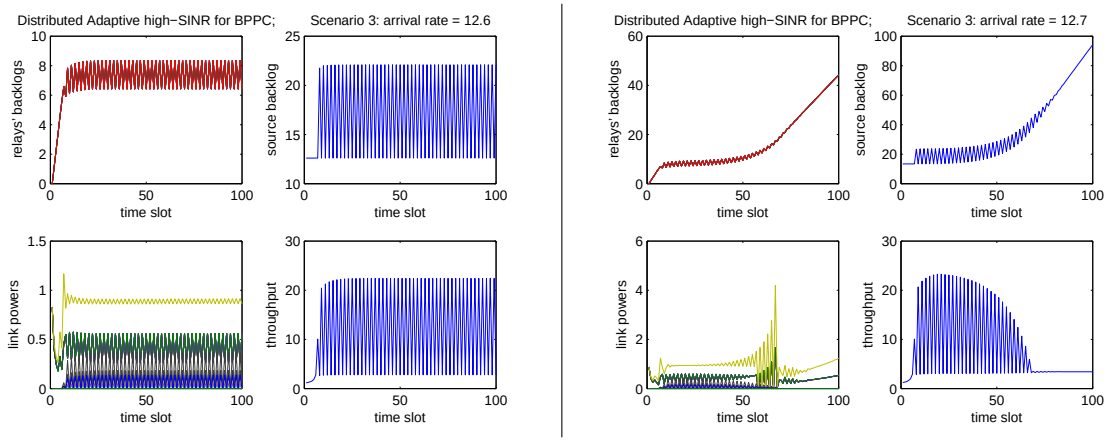

Figure 2.19: Scenario 3: Distributed Adaptive high-SINR for BPPC, varying local penalty parameters version (core step 2), arrival rate = 12.6 pps (left) and 12.7 pps (right).

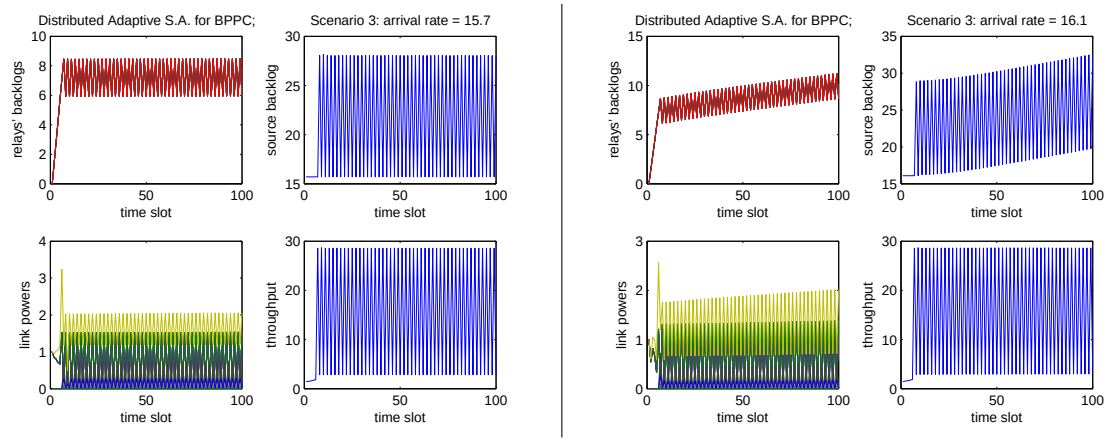

Figure 2.20: Scenario 3: Distributed Adaptive S.A. for BPPC, varying local penalty parameters version (core step 2), arrival rate = 15.7 pps (left) and 16.1 pps (right).

2.6.3 Performance evaluation of distributed WMMSE-based algorithms and comparisons

The performance of the WMMSE-based algorithms for the BPPC problem is examined in this section. For ease of comparison with the distributed S.A.-based algorithms, simulation experiments have been performed in the three different scenarios of wireless networks considered in section 2.6.2. We include the batch and adaptive version of the WMMSE algorithm for BPPC. Different implementations of the algorithms are examined thoroughly through simulations, while, we also compare WMMSE and S.A. algorithms under identical problem instances of BPPC.

Unless otherwise noted, implementation of the WMMSE algorithms is according to their initial forms in subsections 2.4.1 and 2.4.3, i.e., all links with positive differential backlogs are scheduled at the beginning of each slot. We will refer to this implementation as mode 1. In simulation experiments concerning this implementation, and for all three scenarios, the desired accuracy in step 6 of the batch and adaptive WMMSE algorithms was set equal to $\epsilon = 10^{-2}$. For the adaptive version, the window parameter was set to $W = 4$.

• Scenario 1 - small network with moderate interference: The first set of simulations concerns the setup considered in scenario 1, and all parameters are identical. It turned out that in this scenario, both batch and adaptive WMMSE algorithms yield the same solution: they keep only one link on, the link from source to destination node, transmitting at maximum power. Source transmits packets to all nodes in the network only in the first packet/control slot. Beyond, source transmits directly only to the destination node, and this holds for all arrival rates examined. This way, the network can be stabilized for arrival rates up to 12.4 packets per slot (pps). This is the maximum stable throughput that can be achieved, since at arrival rate equal to 12.5 pps and beyond, the source backlog grows to infinity. The evolution of node backlogs, the power allocation, and end-to-end throughput, are illustrated in Fig. 2.21 (left) for arrival rate 12.4 pps, and in Fig. 2.21 (right) for arrival rate 12.5 pps, for the case of batch WMMSE. Its adaptive counterpart reproduces the results, as can be seen in Fig. 2.22 (left) for arrival rate 12.4 pps, and in Fig. 2.22 (right) for arrival rate 12.5 pps. The weighted sum-rate attained is constant with time in this case, therefore we skip the respective plots. Both algorithms yield objective equal to 155.45, for the stable maximum rate. Numerical results concerning maximum stable rate and average weighted sum-rate, for all WMMSE-based algorithms, in all simulation

experiments, are provided in Table 2.6.

WMMSE algorithms proved very fast, as expected, however the adaptive version improves further the average complexity. Table 2.8 summarizes results concerning average complexity of all WMMSE-based algorithms, in all scenarios considered. As an indication, the batch version averaged 117 iterations per slot, requiring 0.045 seconds per slot, on average. On the other hand, the adaptive version did just 1 iteration per slot at the steady state (after first two time slots), requiring on average 0.0064 seconds per slot. Obviously, the adaptive algorithm is significantly lighter in terms of complexity during steady state, compared to its batch counterpart, due to its warm initialization. Note also that this amounts to a reduction in average run-time by an order of magnitude.

Comparing the batch (or adaptive) S.A. algorithm with the batch (or adaptive) WMMSE in this scenario, it is worth noting that the former yields an average weighted sum-rate (184.46) that it is significantly higher than that of WMMSE (155.45), however S.A. achieves lower stable average throughput (10.4 pps) than that of WMMSE (12.4, pps). This may seem surprising, but there is no contradiction here. According to the *throughput-optimal policy* in [53, 54], solving to *optimality* the differential backlog-weighted sum-rate maximization problem, yields the maximum stable throughput. Here we compare two *suboptimal* algorithms which can attain only approximate solutions to the NP-hard BPPC problem. Besides, if we compare the respective instantaneous weighted sum-rate objectives attained by the two algorithms, throughout the network's evolution, which is the objective of Fig. 2.23, it can be clearly seen that in half time slots the objective of S.A. hovers above the respective of WMMSE, while the reverse holds in the other half time slots.

This result implies that when comparing two suboptimal algorithms for the NP-hard BPPC problem, picking the algorithm which yields higher average weighted sum-rate need not necessarily result in the maximum possible stable throughput.

• Scenario 2 - small network with strong interference: WMMSE algorithms exhibit in this scenario the same behavior as in scenario 1. The solution is again that of keeping only the link from source to destination, after the first packet/control slot, on, and transmitting at maximum power. The maximum stable rate in this case is 12.4 pps. Both batch and adaptive algorithms do not stabilize the network for higher arrival rates. For the rate 12.4 pps, the source backlog is stabilized after a transient of 80 slots, therefore we provide the respective plots for the arrival

rate 12.3 pps. The adaptive algorithm keeps up with its batch counterpart, so we skip the associated plots for the batch algorithm. Fig. 2.24 plots the results for arrival rate 12.3 pps (left), and 12.5 pps (right). The weighted sum-rate is constant and equals to 168.62, as obtained from both algorithms. The adaptive algorithm keeps its complexity advantage relative to the batch one in this scenario, reducing the average run-time by an order of magnitude, as can be concluded from Table 2.8.

The WMMSE algorithms (batch and adaptive) turned out to be better, performance-wise, than the respective S.A. algorithms, in this scenario. WMMSE algorithms attain higher average weighted sum-rate (168.62) compared to the S.A. ones (107.48), and they also achieve higher stable throughput (12.3 pps) than S.A. ones (7.8 pps). Fig. 2.26 (left) shows that, almost in all time slots, the instantaneous weighted sum-rate attained by the WMMSE algorithms is significantly higher. It is worth noting that although in this scenario S.A. algorithms converge, at the steady state, to the same solution as WMMSE ones (one link at max. power), the system, under control of the S.A. algorithms, evolves differently during the long transient, which has impact on the eventual stability and its overall evolution.

• Scenario 3 - larger network, moderate interference: WMMSE algorithms activate, in this case also, only the link from source to destination, after the first time slot, and the system converges to a steady state. They both attain this way the same maximum stable throughput, which amounts to 12.1 pps. The queues become unstable at higher arrival rates. The respective results can be visualized in Fig. 2.25, for the arrival rate 12.1 pps (left), and for 12.2 pps (right), concerning the adaptive version. We skip the associated plots for the batch algorithm, since are identical. The resulting weighted sum-rate in this case is 149.03. Both versions of the WMMSE algorithm proved very fast in this scenario, also. Respective complexity results are given in Table 2.8.

In the case of the larger network considered here, the S.A. algorithms clearly outperform the WMMSE ones, in terms of stable throughput, as well as of the attainable weighted sum-rate. Recall that the maximum stable rate that S.A. algorithms can support in this case is 15.7 pps. The average weighted sum-rate obtained from S.A. algorithms is 454.3, while from the WMMSE ones is 149. Comparison between the two types of algorithms, for all scenarios considered, with respect to maximum stable rate and average weighted sum-rate, can be easily made by the results summarized in Table 2.7. Notice also in Fig. 2.26 (right), where the objective function

is plotted with time for both algorithms, that the instantaneous weighted sum-rate obtained by batch S.A., although periodic, hovers significantly above that of batch WMMSE algorithm, in all time slots.

So far the WMMSE algorithms consistently converge, in all three scenarios considered, to the same qualitatively solution. Only the direct link from source to destination is favored. This solution may be throughput optimal, as proved, in both two cases of the small network we considered, however, it proved inferior to the solution of the S.A. approach, in terms of the average weighted sum-rate, in the case of moderate interference (scenario 1). Furthermore, it has been proved that in the case of the setup in scenario 3, keeping only one link on is far worse than activating a subset of links, in terms of both maximum stable throughput and attainable weighted sum-rate. This result motivated us to pursue possibly different solutions to the BPPC problem via the WMMSE algorithm. One possibility, towards this end, is to set a fixed number of iterations for convergence. In case of scenario 1, we found that for 105 iterations per slot, the system converges to a periodic behavior, similar as in the case of S.A. algorithm, where five links are activated periodically. The maximum stable average throughput that WMMSE attains this way is 10.3 pps. Higher rates can only be supported by letting the algorithm terminate in more iterations. In such cases, it eventually results in the one-link solution in order to stabilize the network. However, setting the appropriate number of iterations for every specific setup considered requires experimentation, and renders the algorithm impractical. Another choice is to adopt the centralized termination criterion used in the initial form of the algorithm in [50]. In this implementation the results are accurate at convergence of the algorithm within accuracy of $\epsilon = 10^{-4}$, in which case the same one-link solution is obtained. WMMSE can not, therefore, do any better under the centralized criterion, which is anyway unsuitable for distributed implementation, since it would require more information exchanges among links, increasing the communication overhead.

Upon further experimentation on the WMMSE algorithm, it turned out that it can produce solutions that are qualitatively similar to the ones of the S.A. algorithms. This is possible when the termination criterion adopted here (step 8) is used, but the heuristic scheduling scheme in section 2.5 is also applied. We will refer to this implementation as mode 2. It should be noted that this implementation is possible in the specific setups considered here, since physical layer propagation conditions are assumed fixed, and the optimization that takes place at the physical

layer favors, under the given propagation conditions, the subset of links at each node with the maximum sum of differential backlogs, as proved from our simulations so far.

In this set of experiments we set the window parameter to $W = 5$ for the adaptive version, and the desired accuracy in step 8, of the batch and adaptive WMMSE algorithms, equal to $\epsilon = 10^{-4}$. Simulations have been performed in all three different networks considered, and showed that the adaptive version keeps up with its batch counterpart, consistently. Moreover, WMMSE algorithms attain the same stable average throughput as the S.A. ones, in all cases. A similar periodic behavior of the system also emerges. Fig. 2.27 concerns scenario 1 and the arrival rates 10.4 pps (left) and 10.8 pps (right), for the adaptive WMMSE algorithm (mode 2). The maximum sustainable rate for scenario 2 is 7.8 pps, and for scenario 3 is 15.7 pps. The associated results are illustrated in Fig. 2.28 (left) and Fig. 2.28 (right), for these two stable setups, respectively. Numerical results, concerning the stable rates and the average weighted sum-rate attained by both versions of WMMSE (batch and adaptive), are given in Table 2.6. The average weighted sum-rate that WMMSE algorithms yield is very close to that of the S.A. ones, given in Table 2.7 (within 10^{-1} in the worst case), however, WMMSE algorithms converge to a different power allocation than that of the S.A. ones. Note that the resulting power allocations from the two types of algorithms need not coincide, due to the same reasons explained in the case of the S.A.-based algorithms, in section 2.6.2. Complexity results are also summarized in Table 2.8. Notice there that the adaptive version still reduces the average run-time, during steady state, by an order of magnitude compare to the batch one. Comparing the average complexity of the WMMSE algorithms under implementation of mode 1 and of mode 2, in corresponding stable setups, as shown in Table 2.8, it can be concluded that the adaptive version of WMMSE via mode 2 gains in average run-time over its counterpart via mode 1, and significantly more during steady state. This holds despite the increased number of iterations till $\epsilon-$ convergence, in case of mode 2, and is due to the reduced number of optimization variables (because scheduling scheme is applied). However, the same doesn't hold for the batch algorithms, but note that in case of mode 2 a higher accuracy is used for convergence.

A fair way to compare the S.A. approach with the WMMSE one, with respect to the attainable weighted sum-rate (the objective function of BPPC), is assuming the same weighting factors, i.e., differential backlogs. This way, the two algorithms are about to solve identical problem instances, given that direct and crosstalk power losses are assumed fixed in our setups.

Therefore, we considered the following experiment. We let the network evolve under control of the batch S.A. algorithm (core step 2) without employing the scheduling scheme in section 2.5, i.e., taking all links with positive differential backlogs into account (mode 1). Then, we use the differential backlogs resulting at each packet/control slot, in order to solve the resulting problem instance via batch WMMSE algorithm. We tested both the original implementation of WMMSE algorithm (mode 1) and its implementation according to mode 2 (scheduling scheme applied). We also considered the stable setups, according to the S.A. algorithms, in all three scenarios. We used the same accuracy, $\epsilon = 10^{-3}$, in the respective termination criterion of each algorithm, for ease of comparison with respect to solution accuracy.

The objectives attained by batch S.A. and batch WMMSE (mode 1), in the case of the stable setup of scenario 1 (10.4 pps), are compared in Fig. 2.29 (left). The respective comparison with WMMSE via mode 2 is shown right next, in Fig. 2.29 (right). For ease of visualization of the results, we plot in different panels, besides the instantaneous weighted sum-rates plotted together, their corresponding absolute difference per slot, as well as their average value, over all time slots, together. The respective plots concerning the stable case of scenario 2 (7.8 pps) are gathered in Fig. 2.30, and in Fig. 2.31, for scenario 3 (15.7 pps). As long as WMMSE via mode 2 is concerned, it is worth noting that it keeps up with the S.A. algorithm, as figures show, in all setups considered. Via mode 2, WMMSE activates the same subsets of links in subsequent slots, as S.A. does, in all cases. The weighted sum-rate attained by WMMSE (mode 2) is constantly close, within 10^{-2} accuracy, to the one resulting from the S.A. algorithm. This suggests that the two approaches yield essentially the same approximate solution to the NP-hard BPPC problem. On the other hand, batch S.A. clearly outperforms batch WMMSE via mode 1, in terms of the attainable weighted sum-rate, in all scenarios considered. This is due to the fact that WMMSE (mode 1) does not always activate the same subset of links as S.A. does, under the same problem instance. The completely different behavior that WMMSE presents under these two implementations is due to the different sets of optimization variables used in each case (mode 1 and mode 2), in combination with the different number of iterations it eventually requires until convergence within the given $\epsilon-$ tolerance.

From a complexity point of view, WMMSE algorithms clearly outperform the S.A. ones, as can be easily seen comparing the results in Table 2.4 and Table 2.5, concerning S.A. algorithm, with the corresponding ones in Table 2.8, for WMMSE. The gain in efficiency of the batch

WMMSE algorithm over the batch S.A. amounts to orders of magnitude. This is reasonable since the computational complexity per iteration of the WMMSE algorithm is two orders of magnitude lower than the respective of S.A. However, the adaptive S.A. versions reduce significantly their average complexity during steady state, for stable setups, and doing so, they remain comparable to the WMMSE algorithms, in certain cases.

We now summarize some noteworthy insights and take-home points from our simulations:

- Comparing suboptimal BPPC solutions on the basis of average attained weighted sum-rate may be a fallacy - as this does not predict which algorithm is better in terms of stable throughput.

- While WMMSE is clearly better than S.A.-BPPC complexity-wise, weighted sum-rate and throughput performance-wise, the results are inconclusive. WMMSE clearly outperforms in the strong interference scenario, while S.A.-BPPC holds the edge for larger networks.

- The two different suboptimal solutions to the BPPC problem obtained suggest that activating only the direct link from source to destination in case of strong interference-limited scenarios is better, form a throughput and weighted sum-rate perspective, whereas it is far from optimal in large networks with moderate interference.

- Surprisingly, our proposed scheduling heuristic improves the throughput performance of WMMSE in the larger network scenario.

- The performance of WMMSE is significantly affected by the specific implementation adopted. Thus, when the scheduling heuristic is employed for WMMSE, the WMMSE and S.A. approaches to the BPPC objective maximization yield essentially the same approximate solution.

- The scheduling heuristic improves the average complexity/feedback cost of S.A.-BPPC, with no performance loss.

- The improvement in the original version of WMMSE algorithm, which is possible in our context, is simple but worthwhile. The gain in communication/computational complexity of the algorithm comes from its modification in the real domain, as well as from the warm re-start in the adaptive version proposed.

Table 2.6: Maximum stable throughput and average weighted sum-rate attained by WMMSE algorithms in all scenarios. w.s.r. := weighted sum-rate.

Scenario 1	Batch WMMSE mode 1	Adapt. WMMSE mode 1	Batch WMMSE mode 2	Adapt. WMMSE mode 2
Max stable rate	12.4	12.4	10.4	10.4
Avg. w.s.r	155.48	155.48	184.48	184.48
Scenario 2				
Max stable rate	12.3	12.3	7.8	7.8
Avg. w.s.r.	168.62	168.62	107.32	107.32
Scenario 3				
Max stable rate	12.1	12.1	15.7	15.7
Avg. w.s.r.	149.03	149.03	454.42	454.41

Table 2.7: Comparison between WMMSE and S.A. algorithms with respect to average throughput and average weighted sum-rate attained, in all scenarios. w.s.r. := weighted sum-rate.

Scenario 1	WMMSE (Batch / Adapt) mode 1	WMMSE (Batch / Adapt) mode 2	S.A. (Batch / Adapt) mode 1	S.A. (Batch / Adapt) mode 2
Max stable rate	12.4	10.4	10.4	10.4
Avg. w.s.r	155.4	184.4	184.4	184.4
Scenario 2				
Max stable rate	12.3	7.8	7.8	7.8
Avg. w.s.r.	168.6	107.3	107.4	107.4
Scenario 3				
Max stable rate	12.1	15.7	15.7	15.7
Avg. w.s.r.	149	454.4	454.3	454.3

Table 2.8: Average complexity of WMMSE algorithms for respective stable setups in all scenarios.

Scenario 1	Batch WMMSE mode 1	Adapt. WMMSE mode 1	Batch WMMSE mode 2	Adapt. WMMSE mode 2
Trans. phase: avg. time (sec)	0.045	0.128	0.0794	0.0905
Steady state: avg. time (sec)	0.045	0.0064	0.0794	0.0052
Trans. phase: avg. ♯ iter.	118	41	376	210
Steady state: avg. ♯ iter.	118	1	376	3
Scenario 2				
Trans. phase: avg. time (sec)	0.015	0.1128	0.055	0.0409
Steady state: avg. time (sec)	0.015	0.0061	0.055	0.0045
Trans. phase: avg. ♯ iter.	20	6	242	117
Steady state: avg. ♯ iter.	20	1	242	1
Scenario 3				
Trans. phase: avg. time (sec)	0.06	0.1182	0.1148	0.097
Steady state: avg. time (sec)	0.06	0.0125	0.1148	0.0083
Trans. phase: avg. ♯ iter.	73	25	297	154
Steady state: avg. ♯ iter.	73	1	297	1

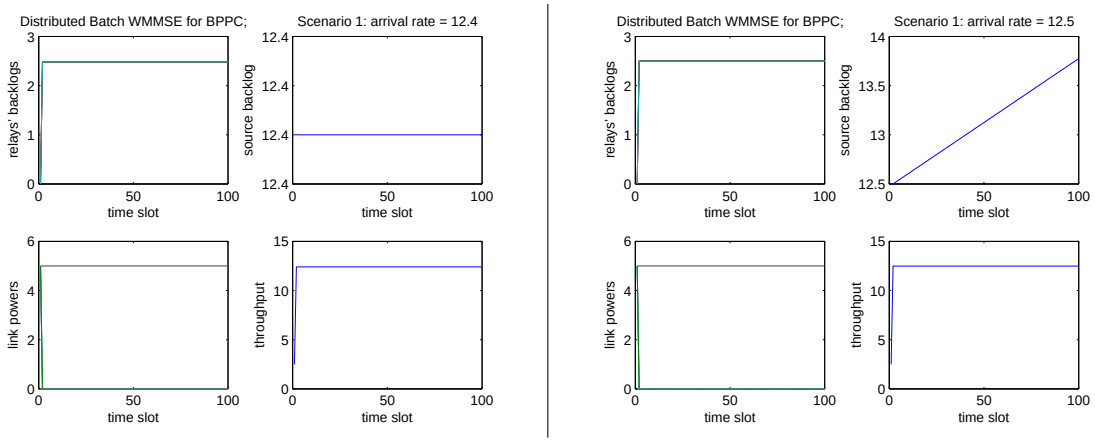

Figure 2.21: Scenario 1: Distributed Batch WMMSE for BPPC, arrival rate = 12.4 (left) and 12.5 (right).

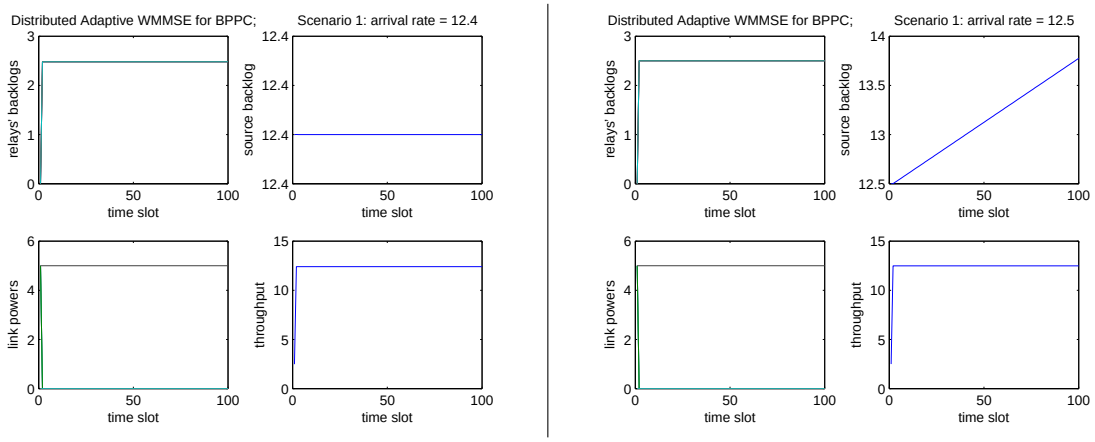

Figure 2.22: Scenario 1: Distributed Adaptive WMMSE for BPPC, arrival rate = 12.4 (left) and 12.5 (right).

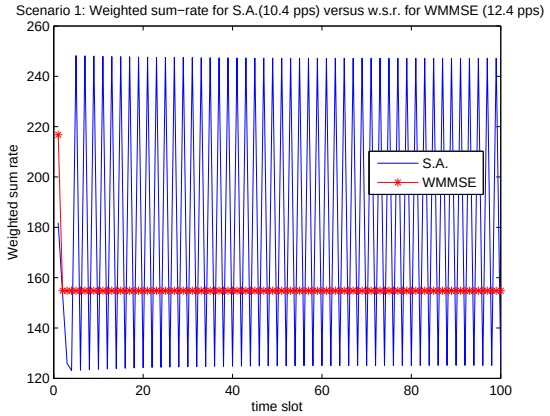

Figure 2.23: Scenario 1: Comparison between objectives attained by Batch S.A. (10.4 pps) and Batch WMMSE (12.4 pps).

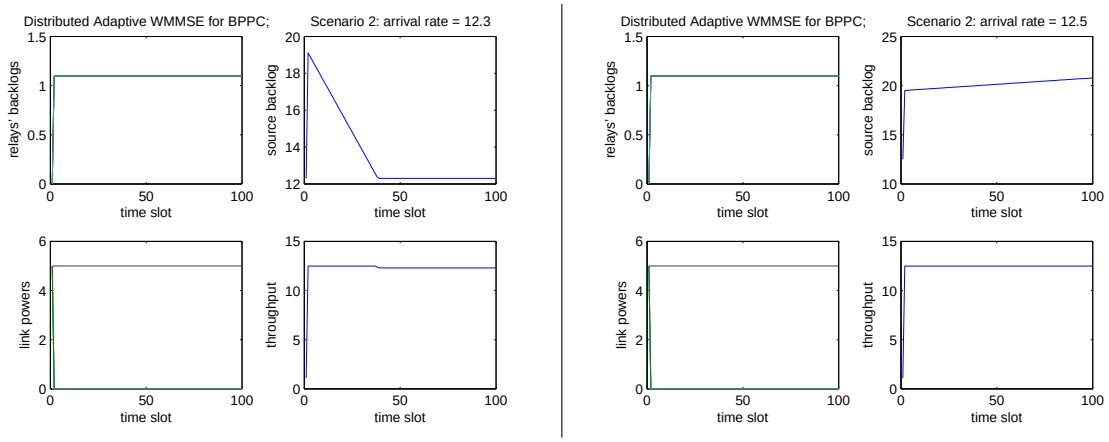

Figure 2.24: Scenario 2: Distributed Adaptive WMMSE for BPPC, arrival rate = 12.3 (left) and 12.5 (right).

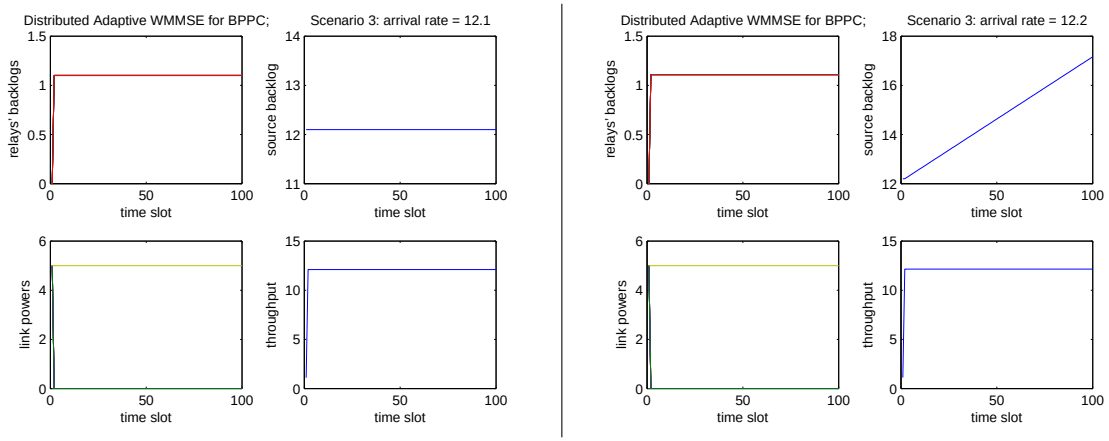

Figure 2.25: Scenario 3: Distributed Adaptive WMMSE for BPPC, arrival rate = 12.1 (left) and 12.2 (right).

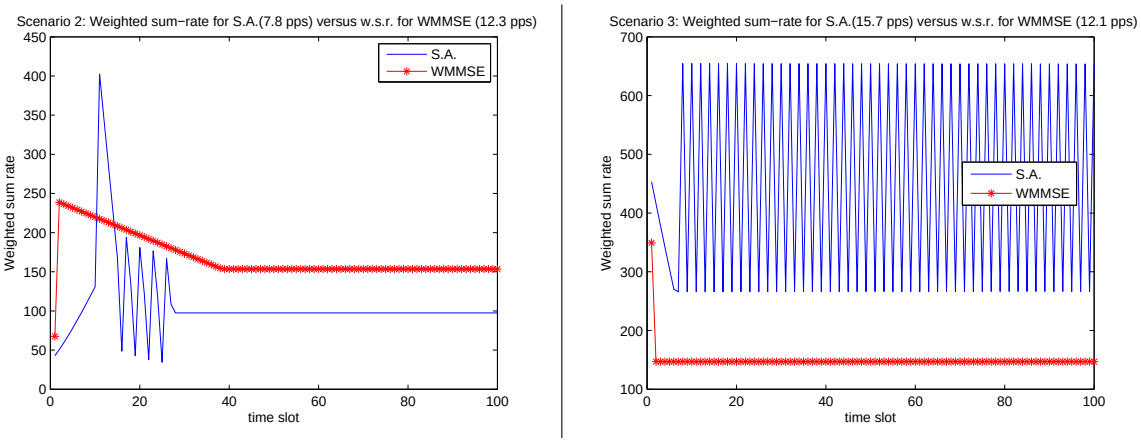

Figure 2.26: Batch S.A. vs Batch WMMSE: comparison between objectives attained. Scenario 2, Batch S.A. (7.8 pps) and Batch WMMSE (12.3 pps) (left), Scenario 3: Batch S.A.(15.7 pps) and Batch WMMSE (12.1 pps) (right).

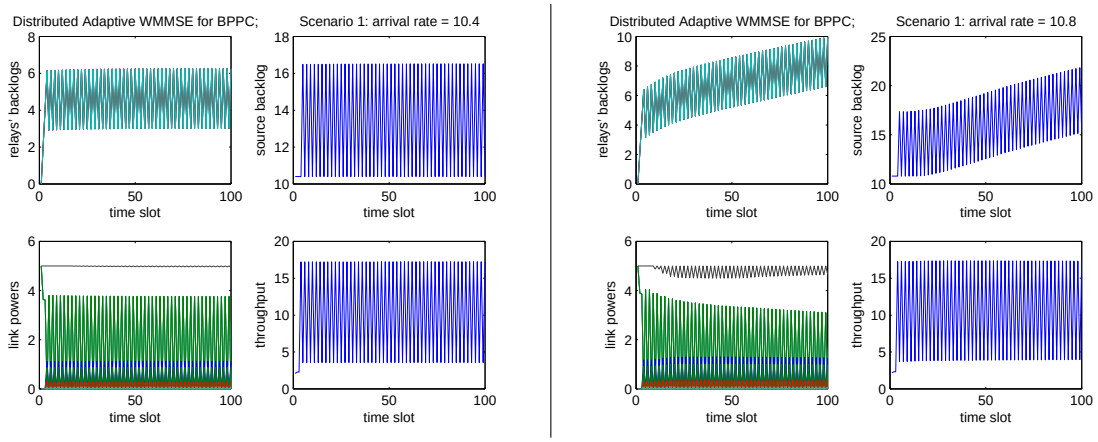

Figure 2.27: Scenario 1: Distributed Adaptive WMMSE for BPPC (mode 2), arrival rate = 10.4 (left) and 10.8 (right).

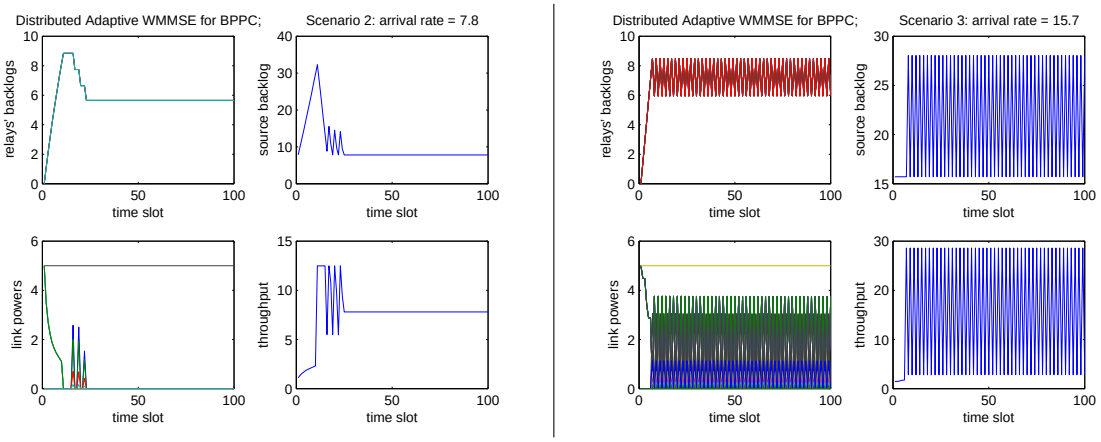

Figure 2.28: Distributed Adaptive WMMSE for BPPC (mode 2): scenario 2, arrival rate = 7.8 (left) and scenario 3, arrival rate = 15.7 (right).

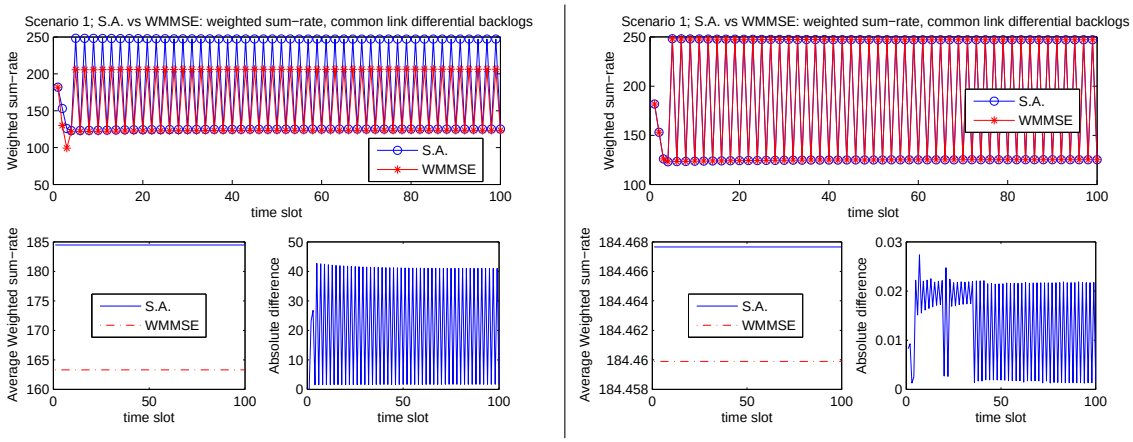

Figure 2.29: Scenario 1, rate = 10.4 pps; Batch S.A. vs Batch WMMSE: objective values attained, Batch WMMSE mode 1 (left) and Batch WMMSE mode 2 (right).

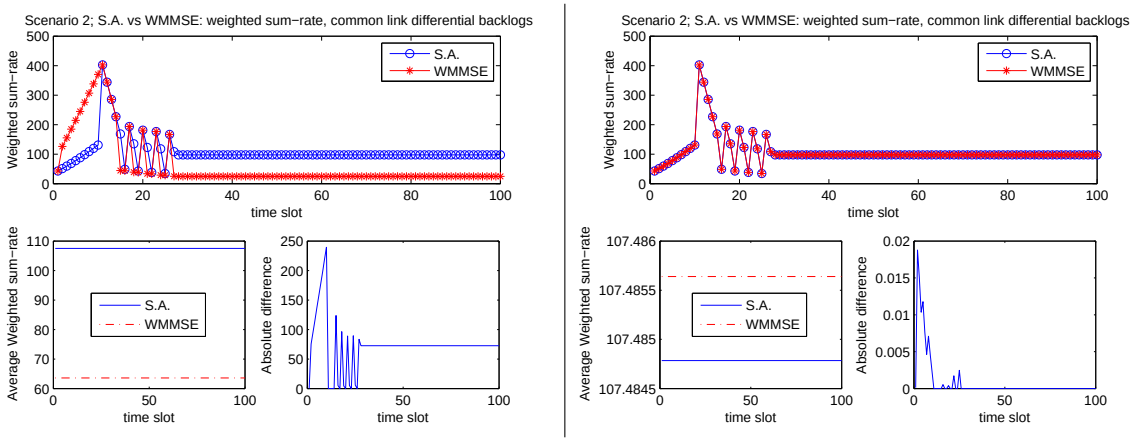

Figure 2.30: Scenario 2, rate = 7.8 pps; Batch S.A. vs Batch WMMSE: objective values attained, Batch WMMSE mode 1 (left) and Batch WMMSE mode 2 (right).

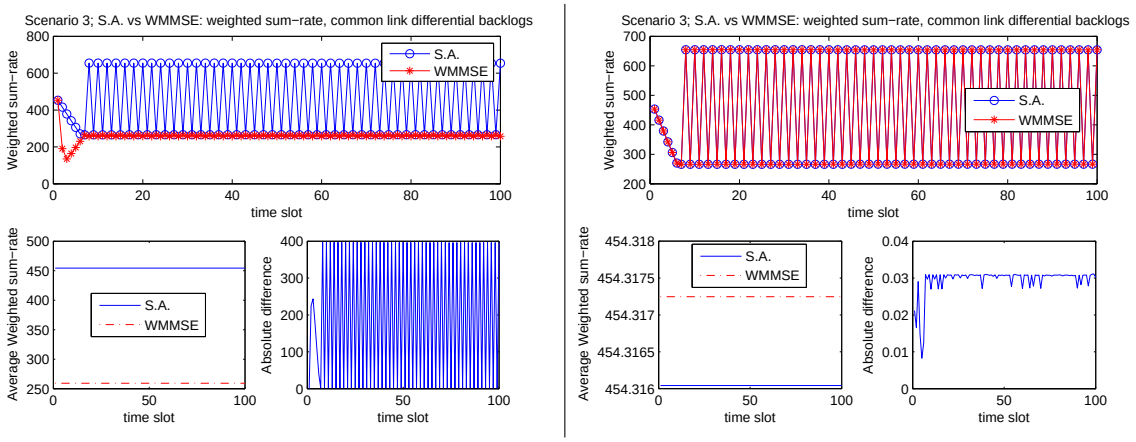

Figure 2.31: Scenario 3, rate = 15.7 pps; Batch S.A. vs Batch WMMSE: objective values attained, Batch WMMSE mode 1 (left) and Batch WMMSE mode 2 (right).

2.7 Conclusions

The distributed implementation of the back-pressure power control policy in wireless multi-hop networks has been considered, aiming to maximize stable end-to-end throughput. Back-pressure power control problem has been shown in [34] to be NP-hard, yet effective approximate solutions have been also proposed therein, based on the successive convex approximation approach. Although this strategy delivers manifold improvement in end-to-end throughput relative to the prior art in wireless networking, the drawback is that existing implementations are centralized. The contribution here is the distributed implementation of the successive convex approximation of BPPC, motivated by the need of distributed power control in wireless networking. The main concern, towards this end, is to derive in a decentralized fashion a solution to the convex approximation of the differential backlog-weighted sum-rate maximization problem, to which BPPC amounts at the physical layer. This proved possible upon utilizing the alternating direction method of multipliers, in view of its favorable properties. Two fully decentralized algorithms for the core step of the successive convex approximation to BPPC have been developed, which match the performance of their centralized counterpart, as judicious simulations reveal.

Elaborating on distributed warm start and on consensus-on-termination issues, as well as exploiting the quasi-periodicity of the solution due to the push-pull operation of the system in stable setups in our context, we come up, next, with efficient distributed adaptive algorithms, that allow tight approximation of BPPC in the general interference regime. The computational complexity of the algorithms and the communication overhead they entail are reasonable, since they yield high-quality approximate solutions to an NP-hard problem, in a fully decentralized fashion. Extensive simulation results show that the distributed adaptive algorithms keep up with their centralized counterparts, and do so with low average complexity in stable setups, which renders them operational for real-time network control, considering slow time-varying channels.

It has also been shown that the iteratively re-weighted Minimum Mean Square Error approach in [50] to weighted sum-rate maximization for the MIMO interference channel, applies in our context, and can be utilized for the approximation of the BPPC problem at the physical layer. One advantage of the WMMSE algorithm in [50], specialized in our context, over the S.A.-based algorithms, is that it provides an one-shot approximate solution to the weighted

sum-rate maximization problem. A second one is its low computational complexity. Despite its efficiency, we have proposed an improved version of the original algorithm in [50], inspired by the development of adaptive solutions to BPPC via the S.A. approach, which reduces further its complexity and feedback cost. Both batch and adaptive versions of WMMSE algorithm, however, enjoy low average complexity and, therefore, are operational in general physical layer propagation conditions. WMMSE is also lighter in terms of communication overhead, relative to the S.A.-based algorithms. Yet, our simulation experiments, considering fixed physical layer propagation conditions, showed that the WMMSE algorithm, in its original formulation, does not attain the best approximate solution to the BPPC problem, from a throughput and weighted sum-rate perspective, in large scale problems. This is due to activating only one link in the network, which is not preferable from other perspectives as well, such as robustness and sharing of the load in the network, for compatibility with the buffering capabilities of the nodes.

However, specific implementations of the WMMSE algorithm, in the setups considered here, have revealed the potential of the WMMSE approach to yield different possible approximate solutions to the BPPC problem, and to achieve the same high-quality approximation to BPPC, as the S.A. approach, even for large scale problems.

Chapter 3

Efficient Batch and Adaptive Approximation Algorithms for Joint Multicast Beamforming and Admission Control

Wireless multicasting is becoming increasingly important for efficient distribution of streaming media and location-aware services to mobile and hand-held devices, network management, and software updates over cellular (UMTS-LTE) and indoor/outdoor wireless networks (e.g., 802.11/16). Multicast beamforming was recently proposed as a means of exploiting the broadcast nature of the wireless medium to boost spectral efficiency and meet Quality of Service (QoS) requirements. Infeasibility is a key issue in this context, due to power or mutual interference limitations. We therefore consider the joint multicast beamforming and admission control problem for one or more co-channel multicast groups, with the objective of maximizing the number of subscribers served and minimizing the power required to serve them. The problem is NP-hard even for an isolated multicast group and no admission control; but drawing upon our earlier work for the multiuser SDMA downlink, we develop an efficient approximation algorithm that yields good solutions at affordable worst-case complexity. For the special case of an isolated multicast, Lozano proposed a particularly simple adaptive algorithm for implementation in UMTS-LTE. We identify strengths and drawbacks of Lozano's algorithm, and propose two

simple but worthwhile improvements. All algorithms are carefully tested on publicly available indoor/outdoor measured channel data.

3.1 Introduction

Wireless multicasting is gaining ground as an enabling technology for mass content distribution (Internet TV, streaming media, pay-per-view, network management, software updates) over wireless networks. Multicasting lies in between two widely used information dissemination modalities: broadcasting, where common information is addressed to all nodes in a network, and parallel (orthogonal or co-channel) unicast transmissions. The middle ground between the two is important for existing and emerging applications.

Multicasting has been traditionally viewed as a network-layer issue, addressed by multicast routing. This viewpoint is natural for wired networks, but wireless is different due to the so-called *broadcast advantage* (a node's transmission may reach multiple receivers) and its flip-side: co-channel interference.

Unlike traditional broadcast radio, wireless networks nowadays incorporate feedback mechanisms that provide varying grades of Channel State Information at the Transmitter (CSI-T). The availability of CSI-T coupled with the proliferation of antenna arrays open the door for multicast beamforming at the transmitter. The idea is to shape the transmit beampattern in a way that steers power in the directions of multicast subscribers while minimizing leakage in other directions. It is also possible to design multiple beampatterns to transmit simultaneously to more than one multicast groups, provided that the different groups are spatially well-separated. Multicast beamforming under Quality of Service (QoS) constraints, measured via the received Signal to Noise Ratio (SNR), was treated in [51], where it was shown that the problem is NP-hard, yet amenable to convex approximation tools. The case of multiple co-channel multicast groups was treated in [22]. A key problem that arises in this context is potential infeasibility, due to power or inter-group crosstalk limitations. If the QoS constraints are rigid, one is led to consider the problem of joint transmit beamforming and admission control. This has been considered in [37] for the multiuser Space-Division Multiple Access (SDMA) downlink scenario, comprising interfering co-channel unicast transmissions.

In this paper, we consider the joint co-channel multicast beamforming and admission control

problem from the viewpoint of maximizing the number of subscribers served at or above a prescribed Signal to Interference plus Noise Ratio (SINR), under an overall power constraint. We pay special attention to the case of a single multicast group, which is important in view of ongoing standardization activity [26,40]. The Evolved Multimedia Broadcast/Multicast Service (E-MBMS) in the context of 3GPP[1] / UMTS-LTE[2] specifically provisions point-to-multi-point physical layer multicasting [40]. Motivated by [51], Lozano [26] proposed a particularly simple alternating gradient iteration for the case of a single multicast group. This being an NP-hard problem, the results reported in [26] are intriguing given the simplicity of Lozano's algorithm.

Our specific contributions in this paper can be summarized as follows:

- We generalize the convex approximation framework developed in [37] for the multiuser SDMA downlink to the co-channel multicast context. This yields an efficient approximation algorithm that is directly applicable in a far broader range of problems: from single-group multicast to the SDMA downlink and everything in between.

- We take a closer look at Lozano's alternating gradient iteration, and identify strengths and weaknesses. We show, by means of simple but instructive examples, that it is sensitive with respect to initialization and can exhibit limit cycle behavior.

- We propose two simple improvements of Lozano's iteration that mitigate these drawbacks and significantly boost overall performance. The resulting algorithm is an ideal candidate for practical implementation in next-generation cellular systems.

- We present a comprehensive suite of carefully designed numerical experiments using publicly available measured channel data, for both indoor and outdoor scenarios.

3.2 Problem Formulation

Consider a base station or access point equipped with N transmit antennas and a population of K subscribers, each with a single receive antenna. Let $\mathbf{h}_i$ be the $N \times 1$ complex channel vector from the transmit antenna array to receiver i, $i \in \{1, \cdots, K\}$. We begin by assuming instantaneous CSI-T, but our formulation and algorithms can be readily adapted to work with long-term CSI-T, as will be explained in the sequel.

[1] Third Generation Partnership Project.

[2] Universal Mobile Telecommunications System - Long Term Evolution.

Consider $1 \leq G \leq K$ multicast groups, $\{\mathcal{G}_1, \ldots, \mathcal{G}_G\}$, where $\mathcal{G}_m$ contains the indices of receivers that wish to subscribe to multicast m. Since the transmissions are co-channel, we may assume without loss of generality that $\mathcal{G}_m \cap \mathcal{G}_l = \emptyset$, $l \neq m$, $\cup_m \mathcal{G}_m = \{1, \ldots, K\}$, and with $G_m := |\mathcal{G}_m|$, $\sum_{m=1}^{G} G_m = K$. Let $\mathbf{w}_m^H$ be the weight vector applied to the N transmitting elements to beamform towards group m, $m \in \{1, \cdots, G\}$, where $(\cdot)^H$ denotes Hermitian transpose. Streaming media applications demand a minimum instantaneous[3] Signal to Interference plus Noise Ratio (SINR). With this in mind, the multicast beamformer design problem can be cast as follows.

$$\min_{\{\mathbf{w}_m \in \mathbb{C}^N\}_{m=1}^{G}} \sum_{m=1}^{G} \|\mathbf{w}_m\|_2^2 \tag{3.1}$$

$$\text{subject to}: \sum_{m=1}^{G} \|\mathbf{w}_m\|_2^2 \leq P, \tag{3.2}$$

$$\frac{|\mathbf{w}_m^H \mathbf{h}_i|^2}{\sum_{\ell \neq m} |\mathbf{w}_\ell^H \mathbf{h}_i|^2 + \sigma_i^2} \geq c_i, \ \forall i \in \mathcal{G}_m, \ \forall m \in \{1, \cdots, G\}, \tag{3.3}$$

where σ_i^2 is the additive noise power at receiver i and c_i stands for the associated minimum SINR requirement. The objective function reflects the desire to pick the minimum power solution when the problem is feasible, while the explicit sum power constraint accounts for regulatory and equipment limitations[4].

The above problem is NP-hard, for it contains the case of a single multicast group ($G = 1$), which is already NP-hard as shown in [51]. Still, it has been shown [22, 51] that it is possible to compute high-quality approximate solutions via *convex (semidefinite) approximation*. The idea is to approximate the non-convex and NP-hard problem using a suitable convex problem, then use the solution of the convex problem to guide the search for a good feasible solution of the original NP-hard problem. The second step in [22, 51] is based on *Gaussian randomization*, which can provide provably good approximate solutions in the sense that the distance to the optimum can be analytically bounded [27]. The case of $G = K$ corresponds to the multiuser Space Division Multiple Access (SDMA) downlink [13] whose convexity was shown in [1].

(In)Feasibility is a key issue with the above formulation. Infeasibility may arise for a number of reasons: due to proximity of channel vectors of users interested in different multicast streams, scattering of users belonging to a given multicast group, degrees of freedom (few transmit

[3]Lower-priority services such as software updates can operate under a long-term average SINR constraint, which can be guaranteed given only long-term CSI-T, as will be discussed in the sequel.

[4]Per-antenna power amplifier constraints can be easily incorporated in our approach; we skip them for brevity.

antennas relative to the number of multicast groups), spatial group interleaving, and power limitations. The power constraint alone may limit coverage even for a single multicast group. While in most cases infeasibility can be detected using the tools developed in [22], what one does next is far less obvious. If the SINR constraints are infeasible, then some form of admission control is needed; but this should ideally be considered together with the beamformer design problem, for the two are obviously coupled.

Towards this end, it makes sense to consider maximizing the number of subscribers that can be served at their desired SINR, and then minimizing the power required to serve those selected in the first step. This approach can be mathematically formulated in two stages.

$$S_o = \text{argmax}_{S \subseteq \{1,\cdots,K\}, \{\mathbf{w}_m \in \mathbb{C}^N\}_{m=1}^G} |S| \tag{3.4}$$

$$\text{subject to : } \sum_{m \mid \mathcal{G}_m \cap S \neq \emptyset} \|\mathbf{w}_m\|_2^2 \leq P, \tag{3.5}$$

$$\frac{|\mathbf{w}_m^H \mathbf{h}_i|^2}{\sum_{\ell \mid \mathcal{G}_\ell \cap S \neq \emptyset, \ell \neq m} |\mathbf{w}_\ell^H \mathbf{h}_i|^2 + \sigma_i^2} \geq c_i, \forall i \in \mathcal{G}_m \cap S, \ \forall m, \tag{3.6}$$

where $|S|$ denotes the cardinality of S. Note that if $\mathcal{G}_m \cap S = \emptyset$, then no constraints are imposed for the given m. Given S_o, we then wish to

$$\min_{\{\mathbf{w}_m \in \mathbb{C}^N\}_{m \mid \mathcal{G}_m \cap S_o \neq \emptyset}} \sum_{m \mid \mathcal{G}_m \cap S_o \neq \emptyset} \|\mathbf{w}_m\|_2^2 \tag{3.7}$$

$$\text{s.t. } \frac{|\mathbf{w}_m^H \mathbf{h}_i|^2}{\sum_{\ell \mid \mathcal{G}_\ell \cap S_o \neq \emptyset, \ell \neq m} |\mathbf{w}_\ell^H \mathbf{h}_i|^2 + \sigma_i^2} \geq c_i, \ \forall i \in \mathcal{G}_m \cap S_o, \ \forall m. \tag{3.8}$$

The second optimization stage in (3.7)-(3.8) is feasible (provided S_o is a solution of (3.4)-(3.6)), but remains NP-hard, as per [22], [51]. This is different from the multiuser SDMA downlink case, considered in [37]. In addition, we have the following result.

Claim 2. *The problem in (3.4)-(3.6) is NP-hard.*

Proof. Consider the decidability version of problem (3.4)-(3.6) for $G = 1$, $S = \{1, \cdots, K\}$ (i.e., $|S| = K$, the maximal value), and $c_i = c$, $\forall i$: does there exist a vector $\mathbf{w}$ such that

$$||\mathbf{w}||_2^2 \leq P,$$

and for which

$$\frac{|\mathbf{w}^H \mathbf{h}_i|^2}{\sigma_i^2} \geq c, \ \forall i \in S?$$

This problem is the decidability version of the following problem

$$\max \min_{i \in S} \frac{|\mathbf{w}^H \mathbf{h}_i|^2}{\sigma_i^2}$$

$$\text{subject to : } ||\mathbf{w}||_2^2 \leq P,$$

which has been shown to be NP-hard in [51]. ∎

The two-stage formulation in (3.4)-(3.6) and (3.7)-(3.8) is not particularly convenient, for a number of reasons. Nested optimization problems are awkward to work with; perhaps more importantly, the nested formulation does not suggest a way to approach the problem from a convex approximation perspective. Towards this end, we will use a technique originally developed in [37] for the multiuser SDMA downlink - which is a special case of our present formulation for $G = K$.

Introduce binary admission control ("slack") variables s_i, one for each receiver $i \in \{1, \cdots, K\}$. When $s_i = -1$ ($s_i = +1$) receiver i is scheduled (rejected, respectively). Consider the following optimization problem.

$$\min_{\left\{\mathbf{w}_m \in \mathbb{C}^N, \{s_i \in \{-1,+1\}\}_{i \in \mathcal{G}_m}\right\}_{m=1}^G} \mathcal{J}\left(\left\{\mathbf{w}_m, \{s_i\}_{i \in \mathcal{G}_m}\right\}_{m=1}^G\right) :=$$

$$\epsilon \sum_{m=1}^{G} \|\mathbf{w}_m\|_2^2 + (1-\epsilon) \sum_{m=1}^{G} \sum_{i \in \mathcal{G}_m} (s_i + 1)^2 \tag{3.9}$$

$$\text{subject to : } \sum_{m=1}^{G} \|\mathbf{w}_m\|_2^2 \leq P, \tag{3.10}$$

$$\frac{|\mathbf{w}_m^H \mathbf{h}_i|^2 + \delta^{-1}(s_i+1)^2}{\sum_{\ell \neq m} |\mathbf{w}_\ell^H \mathbf{h}_i|^2 + \sigma_i^2} \geq c_i, \ \forall i \in \mathcal{G}_m, \ \forall m \in \{1, \cdots, G\} \tag{3.11}$$

Claim 3. *With*

$$\delta \leq \min_{m \in \{1,\cdots,G\}} \min_{i \in \mathcal{G}_m} \frac{4c_i^{-1}}{P \max_{n \in \{1,\cdots,K\}} \|\mathbf{h}_n\|_2^2 + \sigma_i^2},$$

and $\epsilon < \frac{1}{P/4+1}$*, the problem in (3.9)–(3.11) is always feasible, and solution of (3.9)–(3.11) is* equivalent *to first solving (3.4)-(3.6) and then solving (3.7)-(3.8). If there are multiple solutions of (3.4)-(3.6), i.e., if the maximal subset of users that can be served is not unique, then (3.9)–(3.11) will automatically pick a maximal subset requiring minimal total power.*

Proof is deferred to the Appendix.

By virtue of Claim 3, solution of (3.9)-(3.11) could be used to obtain a solution of (3.4)-(3.6), which is NP-hard per Claim 2. It follows that

Claim 4. *The problem in (3.9)-(3.11) is NP-hard.*

This means that if we insist on polynomial complexity in the worst case, we have to give up optimality (unless P=NP); that is, we can only hope for an approximate solution, rather than an optimal one. A key benefit of the single-stage formulation in (3.9)-(3.11) is that it is naturally amenable to convex approximation, as explained next.

3.3 A Semidefinite Relaxation Approach

Define $\mathbf{H}_i := \mathbf{h}_i\mathbf{h}_i^H$, and introduce rank-one positive semidefinite matrix variables $\mathbf{W}_m := \mathbf{w}_m\mathbf{w}_m^H$, and $\mathbf{S}_i := \mathbf{s}_i\mathbf{s}_i^T$, where $\mathbf{s}_i := [s_i \quad 1]^T$. Then (e.g., cf. [31,37]) the optimization problem in (3.9)–(3.11) can be transformed to the following *equivalent* form

$$\min_{\left\{\mathbf{W}_m\in\mathbb{C}^{N\times N},\{\mathbf{S}_i\in\mathbb{R}^{2\times 2}\}_{i\in\mathcal{G}_m}\right\}_{m=1}^{G}} \phi\left(\left\{\mathbf{W}_m,\{\mathbf{S}_i\}_{i\in\mathcal{G}_m}\right\}_{m=1}^{G}\right) :=$$

$$\epsilon\sum_{m=1}^{G}\mathrm{Tr}(\mathbf{W}_m)+(1-\epsilon)\sum_{m=1}^{G}\sum_{i\in\mathcal{G}_m}\mathrm{Tr}(\mathbf{1}_{2\times 2}\mathbf{S}_i) \tag{3.12}$$

$$\text{subject to}: \sum_{m=1}^{G}\mathrm{Tr}(\mathbf{W}_m)\le P, \tag{3.13}$$

$$\frac{\mathrm{Tr}(\mathbf{H}_i\mathbf{W}_m)+\delta^{-1}\mathrm{Tr}(\mathbf{1}_{2\times 2}\mathbf{S}_i)}{\sum_{\ell\neq m}\mathrm{Tr}(\mathbf{H}_i\mathbf{W}_\ell)+\sigma_i^2}\ge c_i,\ \forall i\in\mathcal{G}_m,\ \forall m, \tag{3.14}$$

$$\mathbf{W}_m\ge 0, \mathrm{rank}(\mathbf{W}_m)=1,\ \forall m, \tag{3.15}$$

$$\mathbf{S}_i\ge 0, \mathrm{rank}(\mathbf{S}_i)=1, \mathbf{S}_i(1,1)=\mathbf{S}_i(2,2)=1,\ \forall i\in\mathcal{G}_m,\ \forall m. \tag{3.16}$$

The above is a quadratically constrained quadratic programming (QCQP) problem, in which only the rank-one constraints are non-convex. Wolkowicz [58] has shown that simply dropping the rank-one constraints in this case yields the Lagrange bi-dual problem, which is the strongest convex relaxation of the QCQP problem in the Lagrangian class. Dropping the rank-one constraints yields a semidefinite programming (SDP) problem, which can be efficiently solved using modern interior point methods [5].

$$\min_{\left\{\mathbf{W}_m\in\mathbb{C}^{N\times N},\{\mathbf{S}_i\in\mathbb{R}^{2\times 2}\}_{i\in\mathcal{G}_m}\right\}_{m=1}^{G}} \phi\left(\left\{\mathbf{W}_m,\{\mathbf{S}_i\}_{i\in\mathcal{G}_m}\right\}_{m=1}^{G}\right) =$$

$$\epsilon \sum_{m=1}^{G} \mathrm{Tr}(\mathbf{W}_m) + (1-\epsilon) \sum_{m=1}^{G} \sum_{i \in \mathcal{G}_m} \mathrm{Tr}(\mathbf{1}_{2\times 2}\mathbf{S}_i) \tag{3.17}$$

$$\text{subject to}: \sum_{m=1}^{G} \mathrm{Tr}(\mathbf{W}_m) \leq P, \tag{3.18}$$

$$\frac{\mathrm{Tr}(\mathbf{H}_i\mathbf{W}_m) + \delta^{-1}\mathrm{Tr}(\mathbf{1}_{2\times 2}\mathbf{S}_i)}{\sum_{\ell \neq m} \mathrm{Tr}(\mathbf{H}_i\mathbf{W}_\ell) + \sigma_i^2} \geq c_i, \ \forall i \in \mathcal{G}_m, \ \forall m, \tag{3.19}$$

$$\mathbf{W}_m \geq 0, \ \forall m, \tag{3.20}$$

$$\mathbf{S}_i \geq 0, \mathbf{S}_i(1,1) = \mathbf{S}_i(2,2) = 1, \ \forall i \in \mathcal{G}_m, \ \forall m. \tag{3.21}$$

Remark 1. *Being a relaxation of (3.12)-(3.16), the problem in (3.17)-(3.21) is also feasible when ϵ and δ are chosen as per Claim 3.*

Solution of the relaxed problem in (3.17)-(3.21) in general only provides a lower bound on the cost of an optimal solution to the original problem in (3.12)-(3.16). If a particular solution of (3.17)-(3.21) consists of rank-one $\mathbf{W}_m$, $\forall m$ *and* rank-one $\mathbf{S}_i$, $\forall i$, then it is also a solution (3.12)-(3.16); although this situation does happen in practice, it does not always happen. What is needed is a way to "convert" an optimal solution of (3.17)-(3.21) into a good feasible solution of (3.12)-(3.16). This step is presently an art guided by partial results. A commonly used technique is *Gaussian randomization*, which can generate provably good approximate solutions in certain cases (e.g., see [27] which is applicable in the special case of an isolated multicast scenario, albeit not accounting for admission control). In other cases reasonable heuristics are often used, and the overall solution is benchmarked against the relaxation lower bound[5] and/or exhaustive search when applicable - in particular for small problem instances.

In our present context, the following algorithm seems to work best in practice, among numerous options we have tried:

Algorithm 14. *Multicast* M*embership* D*eflation by* R*elaxation (MDR):*

1. $\mathcal{U} \leftarrow \{1, ..., K\}$

2. $\mathcal{G}_m \leftarrow \mathcal{G}_m \cap \mathcal{U}, \forall m$

[5]We know that an optimal solution of the original NP-hard problem can never have lower cost than that attained by the possibly higher-rank solution of the corresponding rank relaxation.

3. *Solve (3.41)-(3.45), and let* $\check{\mathbf{W}}_m$ *denote the resulting transmit covariance matrices*

4. *For each* m *such that* $\mathcal{G}_m \neq \emptyset$*, extract the principal component of* $\check{\mathbf{W}}_m$*, and scale it to power* $Tr(\check{\mathbf{W}}_m)$*; i.e., set* $\check{\mathbf{w}}_m := \sqrt{Tr(\check{\mathbf{W}}_m)}\check{\mathbf{u}}_m$*, where* $\check{\mathbf{u}}_m$ *is the unit-norm principal component of* $\check{\mathbf{W}}_m$*.*

5. *For all* m *such that* $\mathcal{G}_m \neq \emptyset$*, and each* $i \in \mathcal{G}_m$*, check whether*

$$\frac{|\check{\mathbf{w}}_m^H \mathbf{h}_i|^2}{\sum_{\ell \mid \mathcal{G}_\ell \neq \emptyset, \ell \neq m} |\check{\mathbf{w}}_\ell^H \mathbf{h}_i|^2 + \sigma_i^2} \geq c_i$$

holds. If this is true for all $i \in \mathcal{G}_m$*, and all* m *such that* $\mathcal{G}_m \neq \emptyset$*, stop (a feasible solution has been found); else pick the user with largest gap to its target SINR (smallest attained SINR if all the SINR targets are equal), remove from* $\mathcal{U}$ *and go to step 2.*

3.3.1 Implementation Complexity

The worst-case complexity of MDR is $O(K^{4.5}\log(1/\epsilon))$, where ϵ is the required relative accuracy of the duality gap at termination [37, 59].

Note that it is possible to further reformulate (3.17)-(3.21) to reduce complexity by a factor of two, as shown in the Appendix.

3.4 Special case: Single Multicast

The special case of a single (isolated) multicast group ($G = 1$) is of particular interest, due to on-going standardization activity in the context of UMTS-LTE. This has been considered in [51], but without regard to infeasibility / admission control issues. In the case of a single multicast, assuming that $\mathbf{h}_i \neq \mathbf{0}$, $\forall i$, infeasibility arises only due to the transmit power constraint - there is no interference from other groups. An alternative scenario where the same problem arises is when the beamformers of the different groups are sequentially optimized, and interference is clumped together with the additive noise terms. Letting $\mathcal{U}$ denote the set of potential subscribers, the two-stage formulation in (3.4)-(3.8) reduces to the following problem

$$S_o = \text{argmax}_{S \subseteq \mathcal{U}, \mathbf{w} \in \mathbb{C}^{\mathcal{N}}} |S| \tag{3.22}$$

$$\text{subject to : } \|\mathbf{w}\|_2^2 \leq P, \tag{3.23}$$

$$\frac{|\mathbf{w}^H \mathbf{h}_i|^2}{\sigma_i^2} \geq c_i, \forall i \in S, \tag{3.24}$$

and given S_o

$$\min_{\mathbf{w}\in\mathbb{C}^N} \|\mathbf{w}\|_2^2 \tag{3.25}$$

$$\text{subject to}: \frac{|\mathbf{w}^H\mathbf{h}_i|^2}{\sigma_i^2} \geq c_i,\ \forall i \in S_o. \tag{3.26}$$

The single-stage reformulation in (3.9)-(3.11) reduces to

$$\min_{\mathbf{w}\in\mathbb{C}^N,\{s_i\in\{-1,+1\}\}_{i\in\mathcal{U}}} \epsilon\|\mathbf{w}\|_2^2 + (1-\epsilon)\sum_{i\in\mathcal{U}}(s_i+1)^2 \tag{3.27}$$

$$\text{subject to}: \|\mathbf{w}\|_2^2 \leq P, \tag{3.28}$$

$$\frac{|\mathbf{w}^H\mathbf{h}_i|^2 + \delta^{-1}(s_i+1)^2}{\sigma_i^2} \geq c_i,\ \forall i \in \mathcal{U}, \tag{3.29}$$

where $\delta \leq \min_{i\in\mathcal{U}} \frac{4c_i^{-1}}{\sigma_i^2}$, $\epsilon < \frac{1}{P/4+1}$, as per Claim 3. The final optimization problem after "squaring" of variables and rank relaxation is the following SDP.

$$\min_{\mathbf{W}\in\mathbb{C}^{N\times N},\{\mathbf{S}_i\in\mathbb{R}^{2\times 2}\}_{i\in\mathcal{U}}} \epsilon\text{Tr}(\mathbf{W}) + (1-\epsilon)\sum_{i\in\mathcal{U}}\text{Tr}(\mathbf{1}_{2\times 2}\mathbf{S}_i) \tag{3.30}$$

$$\text{subject to}: \text{Tr}(\mathbf{W}) \leq P, \tag{3.31}$$

$$\frac{\text{Tr}(\mathbf{H}_i\mathbf{W}) + \delta^{-1}\text{Tr}(\mathbf{1}_{2\times 2}\mathbf{S}_i)}{\sigma_i^2} \geq c_i,\ \forall i \in \mathcal{U}, \tag{3.32}$$

$$\mathbf{W} \geq 0, \tag{3.33}$$

$$\mathbf{S}_i \geq 0, \mathbf{S}_i(1,1) = \mathbf{S}_i(2,2) = 1,\ \forall i \in \mathcal{U}. \tag{3.34}$$

The overall convex approximation approach for $G = 1$ entails a trimmed-down version of MDR, listed below for clarity.

Algorithm 15. *Single group MDR:*

1. $\mathcal{U} \leftarrow \{1, ..., K\}$
2. *Solve the relaxed problem (3.30)-(3.34), and let* $\check{\mathbf{W}}$ *denote the resulting transmit covariance matrix*
3. $\check{\mathbf{w}}$ = *principal component of* $\check{\mathbf{W}}$, *scaled to power* $Tr(\check{\mathbf{W}})$.
4. *For each* $i \in \mathcal{U}$, *check whether* $|\check{\mathbf{w}}^H\mathbf{h}_i|^2/\sigma_i^2 \geq c_i$. *If true* $\forall\ i \in \mathcal{U}$, *stop (feasible solution has been found); else pick user with largest gap to its target SNR, remove from* $\mathcal{U}$ *and go to step 2.*

Extensive experiments with measured channels (cf. Section **??**) indicate that MDR provides very satisfactory performance (about one user less than optimum, on average, in our experiments) at a moderate average and worst-case complexity that grow gracefully with the problem size. Still, MDR takes order of second to execute on a typical PC, and, perhaps more importantly, it is a batch algorithm that solves the problem from scratch every time. Even though warm-start options can be readily envisioned and could be used to track small variations inexpensively, a practitioner would naturally prefer a solution that works well at very low complexity - an adaptive filtering-type algorithm, preferably.

Is it possible to have good performance at really low complexity for an NP-hard problem? This seems *a priori* highly unlikely. Yet Lozano recently proposed an intriguing adaptive filtering algorithm that apparently works well in limited but realistic experiments in the context of UMTS-LTE [26]. In the following section, we take a closer look at Lozano's algorithm.

3.5 Lozano's Algorithm

In the sequel, let $\mathbf{H}_i$ denote either $\mathbf{h}_i\mathbf{h}_i^H$ or its expectation, depending on whether instantaneous CSI-T or long-term CSI-T is assumed. When instantaneous CSI-T is available, $\mathbf{w}^H\mathbf{H}_i\mathbf{w}/\sigma_i^2$ is the instantaneous SNR at receiver i for weight vector $\mathbf{w}$; with long-term CSI-T, $\mathbf{w}^H\mathbf{H}_i\mathbf{w}/\sigma_i^2$ is the expected SNR at receiver i for weight vector $\mathbf{w}$. Lozano's algorithm is a very simple alternating gradient iteration: at each step it sorts users according to presently attained SNRs, discards the users with the poorest SNRs, and makes a gradient step in the direction of the *weakest retained* user. The choice of users to drop is based on either a fixed SNR threshold, or (better) a fixed number / percentage of users to keep at each iteration. Dropped users participate as candidates in the next iteration. Lozano's algorithm can be summarized as follows.

Algorithm 16. *Lozano's iteration for single-group multicast [26]:*

1. *Initialize:* $\mathbf{w} = [1\ 0\ \cdots 0]^T$
2. *Compute* $SNR_i(t-1) = \frac{P}{\sigma_i^2}\mathbf{w}_{t-1}^H\mathbf{H}_i\mathbf{w}_{t-1},\ \forall i \in \mathcal{U}$
3. *Sort* $SNR_i(t-1),\ \forall i \in \mathcal{U}$.
4. *Drop a fixed proportion of users with lowest attained* SNR*s.*

5. *Find weakest link among remaining ones ($\rightarrow k$)*

6. *Take step in its direction:* $\mathbf{w}_t = \mathbf{w}_{t-1} + \mu \mathbf{H}_k \mathbf{w}_{t-1}$*; then* $\mathbf{w}_t = \mathbf{w}_t / \|\mathbf{w}_t\|_2$

7. *Repeat until no significant change in minimum SNR.*

The ingenuity of this algorithm lies precisely on giving up on the weakest links - to focus on the remaining, more promising ones. Performance is far worse when rejection is not employed. The choice of number / percentage of users to keep at each iteration has a significant impact on performance. The choice of step-size parameter μ is also important. Unfortunately, theoretical analysis of Lozano's algorithm appears difficult, precisely due to the rejection step; but even empirical choice of parameters is difficult, as illustrated next.

3.5.1 A closer look to Lozano's Algorithm

Despite its conceptual simplicity, Lozano's algorithm exhibits intricate convergence behavior. Consider the following contrived but instructive scenario: there are $N = 2$ transmit antennas and $K = 2$ users, with channels $\mathbf{h}_1 = [1\ 0]^T$ and $\mathbf{h}_2 = [0\ 1]^T$ (each user only listens to a single transmit antenna). Let $\sigma_1 = \sigma_2 = P = 1$. If both users should be served, the optimal solution is $\mathbf{w} = \frac{1}{\sqrt{2}}[1\ 1]^T$, attaining an SNR of $\frac{1}{2}$ for each user. Lozano's algorithm initialized with $\mathbf{w}_0 = \mathbf{h}_1$ (say, because it was previously serving only user 1, and now user 2 comes into the system) has a fixed point at $\mathbf{h}_1$, which is in the null space of $\mathbf{H}_2$ - thus $\mathbf{w}_t = \mathbf{w}_0$ for all $t \geq 0$ and all μ, implying that user 2 is simply shut off from the system. This shows that the algorithm can converge to a suboptimal and unfair solution.

A small perturbation of either $\mathbf{h}_2$ or $\mathbf{w}_0$ takes the algorithm away from this undesirable fixed point; for small enough μ, the iterates typically approach the optimum solution, albeit slowly. Beyond the usual speed - misadjustment trade-off, however, in this simple scenario Lozano's algorithm typically exhibits limit cycle behavior when randomly initialized. Choosing a smaller μ helps reduce the magnitude of the oscillation, but naturally reduces the speed of adaptation, as illustrated in Figure 3.1. Limit cycles are particulary annoying because they make it hard to select an appropriate tolerance threshold.

Evidently, Lozano's algorithm may fail to converge, or converge to a suboptimal or unfair solution, and is sensitive with respect to initialization, problem instantiation, and the choice of parameters. These issues do arise in realistic scenarios, however the algorithm performs

considerably better, on average, than what the above example may suggest, and its simplicity is certainly appealing. It therefore makes sense to consider modifying it to mitigate its two major shortcomings (sensitivity to initialization, and the potential for limit cycle behavior) while maintaining its simplicity.

3.6 Improving Lozano's Algorithm

We propose the following two simple but worthwhile improvements to Lozano's algorithm.

- Lopez [25] considered multicast beamforming from the viewpoint of maximizing *average* SNR. While *minimum* SNR is what determines the common multicast rate [51], the average SNR solution can serve as a reasonable starting point for further improvement via adaptive algorithms. Lopez has shown [25] that maximizing the average SNR in a multicast context reduces to determining the principal eigenvector of the (normalized) channel correlation matrix. In the case of instantaneous CSI-T, this is defined as $\sum_{k=1}^{K} \mathbf{h}_k \mathbf{h}_k^H / \sigma_k^2 = \mathbf{H}\mathbf{H}^H$, where $\mathbf{H} := [\mathbf{h}_1/\sigma_1, \cdots, \mathbf{h}_K/\sigma_K]$; whereas for long-term CSI-T it is $\sum_{k=1}^{K} E[\mathbf{h}_k \mathbf{h}_k^H / \sigma_k^2]$. Finding the principal eigenvector can be accomplished via adaptive algorithms (e.g., based on the power method), which can also be employed in tracking mode. As a result, the overall solution remains simple and adaptive in nature. We call this algorithm LLI, for *Lozano with Lopez Initialization*. The only difference between LLI and Lozano's original algorithm is in the initialization - step 1), where the average SNR beamformer of Lopez is used in LLI.

- A simple and effective way to suppress limit cycle behavior is to damp μ according to a predefined back-off schedule. This should be balanced against our primary objective, which is to find a good solution. Aggressively damping μ limits how much of the search space we can explore.

  The weight update of Lozano's algorithm can be interpreted as taking a step in the direction of the *local* subgradient of $\min_{i \in \mathcal{A}_t} SNR_i(t-1)$, where the minimum is taken over the currently active user set $\mathcal{A}_t$. There are two difficulties here: this is not a subgradient in the usual sense, because it is only a local, not a global under-estimator of $\min_{i \in \mathcal{A}_t} SNR_i(t-1)$; and $\mathcal{A}_t$ can change as iterations progress. These difficulties arise because we are dealing

with a non-convex and NP-hard problem. For convex problems it is known that subgradient optimization using a step-size sequence μ_t such that $\sum_{t=1}^{\infty} \mu_t = \infty$ but $\sum_{t=1}^{\infty} \mu_t^2 < \infty$ (e.g., $\mu_t = \frac{1}{t}$, $t > 0$) yields an algorithm that converges to the optimum. In our context, the choice of back-off schedule is not obvious. We have tried the following options ($\mu_0 = 1$ in all cases):

1. $\mu_t = \frac{1}{t}$;
2. $\mu_t = \frac{1}{\text{floor}(t/10)}$ (thus μ_t is reduced every 10 iterations);
3. $\mu_t = \frac{\mu_{t-1}}{2}$ (exponential back-off);
4. $\mu_t = \frac{\mu_{t-1}}{2}$ if t mod $10 = 0$, else $\mu_t = \mu_{t-1}$ (exponential back-off every 10 iterations);
5. $\mu_t = \frac{\mu_{t-1}}{t}$ (even more aggressive);
6. $\mu_t = \frac{\mu_{t-1}}{t/10}$ if t mod $10 = 0$, else $\mu_t = \mu_{t-1}$ (same as the previous one but back-off every 10 iterations).

We have tested these options (with Lopez initialization) in *extensive* experiments with simulated and measured channel data (cf. simulations section). Options 1, 2, 6 performed equally well, whereas 3, 4, 5 were worse. Between 1, 2, and 6, option 6 was two orders of magnitude faster than the other two. We therefore settled on option 6, in which μ is aggressively damped every 10 iterations. This is of course *ad-hoc*, but so is the overall algorithm - and it is hard to argue with something simple that works very well in practice, as we will show in our experiments. We call this variant dLLI (for *damped* LLI). dLLI differs from Lozano's original algorithm in that it uses Lopez initialization in step 1), and step-size back-off option 6 for the weight update in step 6).

Remark 2. *Lozano's algorithm and LLI use a fixed step-size, thus having the potential to track changes in the operational environment - e.g., due to users coming in and out of the system and/or user mobility. The use of a vanishing step-size, on the other hand, implies that dLLI* per se *is not capable of tracking. In our particular context, however, channel vector updates will generally be infrequent (relative to the downlink signaling rate), and users will drop or attempt to join at an even slower time scale. In between such updates, we have to solve a static problem. Each static problem can be solved with dLLI, using either the average SNR beamformer (Lopez) or the beamvector computed for the previous "slot" (problem instance) as initialization. Which*

initialization is best will depend on the type of update (e.g., new user or updated channel vector for existing user, and user mobility). Since dLLI is a cheap algorithm, the pragmatic approach would be to run two parallel iterations with both initializations and choose the best in the end.

3.7 Experiments

The problems we are aiming to solve are important in practice, albeit NP-hard. MDR is only a well-motivated approximation, and the adaptive algorithms (Lozano's, LLI, dLLI) are merely common-sense engineering. We have conducted extensive and carefully designed experiments to assess the performance of all algorithms using measured channel data. We further tested all algorithms in simulations with i.i.d. Rayleigh channels. We do not report the i.i.d. Rayleigh results for brevity, but note that these were consistent with those obtained using measured channels.

Approximate solutions should ideally be compared to exact (optimal) ones to assess the quality of approximation. Unfortunately, this is not possible in our context, because even the single-group version of the problem without admission control is NP-hard. Still, for fixed SINR targets, we can enumerate over all possible subsets of users using the potentially higher-rank SDP relaxation in [22] to test each subset. This will be referred as ENUM in the sequel, and it yields an upper bound on the number of users that can be served (and a lower bound on the power required to serve them) under the given SINR and power constraints. This bound is the tightest that can be obtained via duality theory, but remains optimistic in general, because it allows for higher-rank transmit covariances (beamforming corresponds to rank-one transmit covariance).

If after testing all subsets ENUM returns a set of transmit covariances which are all rank-one, then this set is an exact (optimal) solution of the original NP-hard problem in (3.4)-(3.6), and thus ENUM yields the ultimate benchmark. This is because it is not possible to serve any more users in this case, even using higher-rank transmit covariances - this possibility has already been tested during ENUM. This is very important, because it happens in the vast majority of cases considered in our experiments. Only in rare cases does ENUM return higher-rank solutions, as we will see in the sequel. The drawback of ENUM is of course its exponential complexity in the number of users, K, which makes it prohibitive for K over $10-12$ on a current PC.

Two different kinds of wireless scenarios are considered for both single multicast and multiple multicast groups: measured outdoor channel data, and measured indoor channel data. Measured channel data were downloaded from the iCORE HCDC Laboratory, University of Alberta, at http://www.ece.ualberta.ca/ mimo/ (see also [16]). The outdoor scenario ('Quad') is illustrated in Figure 3.2 and described in [35, 37]. The indoor scenario ('2nd Floor ECERF') is illustrated in Figure 3.3, and is briefly described next. In both Figures, Tx denotes the (four-element) transmit antenna array location, whereas the numbers denote the positions of each user's single receive antenna. Data selection and pre-processing follows [35, 37].

'2nd Floor ECERF' is a typical office environment. The floor includes many small offices, divided by thin wooden plates with embedded windows. There are many small corridors as well. The whole room is mainly used by staff and professors of the University of Alberta. The transmitter is placed at the reception area, where many people walked into during measurements. Both the transmitter and the receivers are fixed; positions where measurements have been taken are marked in the floor plan. The distance of the transmitter to a concrete wall, which is covered by wooden plates, is less than 0.5 meters. The main corridor, in which locations 1, 2 and 6 are marked, has a width of 2.5 meters and a height of 4 meters to a concrete ceiling. Locations 1 and 2 are about 18 and 34 meters, respectively, from the transmitter. Location 6 is halfway between the transmitter and location 2. Locations 3,4 and 5 have a distance of 23, 19, and 10 meters, respectively to the main corridor. No measurements are available for location 7. Both the transmitter and the receivers are equipped with antenna arrays, each comprising four vertically polarized dipoles, spaced $\lambda/2$ ($\approx$ 16 cm) apart. As described in [37], at each Rx Location, 9 different measurements were taken by shifting the Rx antenna array on a 3×3 square grid with $\lambda/4$ spacing. Each measurement contains about 100 4×4 channel snapshots, recorded 3 per second. We used measurements corresponding to the locations that are marked in Figure 3.3.

We next present experiments for the case of multiple co-channel multicast groups, in which the simpler adaptive algorithms are not applicable; the case of an isolated multicast follows.

3.7.1 Multiple co-channel multicast groups

In experiments 1 and 2 we consider the case of $G = 3$ co-channel multicast groups. The number of transmit antennas is $N = 4$, while the number of single-antenna receivers is $K = 12$, in all cases. We use instantaneous channel vectors (rank-one channel covariance matrices). The

reported results are averages over 30 temporal channel snapshots, spanning 30 seconds.

Experiment 1 concerns the 'Quad' measured outdoor scenario. Users are split in the following three groups of four users each: $\mathcal{G}_1 = \{1, 3, 13, 15\}$, $\mathcal{G}_2 = \{4, 6, 10, 16\}$, $\mathcal{G}_3 = \{7, 8, 9, 19\}$, see Figure 3.2. The remaining parameters are as follows: $P = 10^3$; $\sigma_i^2 = \sigma^2 = 1$, $c_i = c$, $\forall i$; for MDR, $e = 10^{-9} < \frac{1}{P/4+1}$, and $\delta = \frac{1/24c^{-1}}{P \max_m \|\mathbf{h}_m\|_2^2 + \sigma^2}$. Performance of MDR is compared to that of ENUM. The detailed results are reported in Table 3.1. For ease of visualization, Figure 3.4 is a plot of the average number of users served, as a function of target SINR in dB, for both ENUM and MDR. It is important to note here that ENUM returned only rank-one transmit covariance matrices in 95% of the cases considered. Only in the rest 5% of the cases were higher-rank covariance matrices returned by ENUM, and in all these cases it was possible to serve only one additional user using these higher-rank covariances, and this required significant excess power. In this experiment, MDR serves on average about half a user less than ENUM when the target SINR is high, and about one and a half user less when the target SINR is low.

Experiment 2 concerns the '2nd Floor ECERF' measured indoor scenario. Users are split in the three groups $\mathcal{G}_1 = \{1, 2, 3, 13\}$, $\mathcal{G}_2 = \{5, 14, 6, 16\}$, $\mathcal{G}_3 = \{4, 15, 9, 8\}$, see Figure 3.3. Detailed results are reported in Table 3.2, and summary plots in Figure 3.5. The parameters are the same as in Experiment 1. In this indoor scenario, MDR serves on average about one user less than ENUM, which returns higher-rank covariance matrices in 9% of cases.

Summarizing the multi-group multicast experiments, MDR appears to work well in the cases considered, albeit the gap to ENUM is not as small as in the multiuser SDMA downlink case considered in [37]. This is natural, because multicasting is a much harder problem - even its plain-vanilla version is NP-hard. The gap in MDR performance relative to ENUM should be considered in light of the associated complexities: MDR terminates in under 1 second in all cases considered, whereas ENUM takes 5-50 minutes for $K = 12$, on a current PC (see tables 3.1, 3.2, for more detailed execution time results).

It is also worthwhile to note that ENUM indeed yields the exact solution of the original NP-hard problem in the vast majority of cases considered - which was rather unexpected. It appears that *subset selection diversity* plays a role here: among the many possibilities of choosing a subset of users of fixed cardinality, there is typically one for which the optimal beamforming problem is "easy" - rank relaxation is not a relaxation after all; see also [11], where again a suboptimal solution operates close to the optimal one. In the few cases that ENUM returned a higher-rank

solution, this served exactly one additional user, and required significant excess power to do so. The conclusion is that ENUM indeed yields a tight upper bound on the achievable performance.

3.7.2 Single multicast

We now turn to single group multicasting ($G = 1$), and compare ENUM, MDR , Lozano's algorithm, LLI , and dLLI.

The three adaptive algorithms (Lozano's, LLI, dLLI) fix coverage (number of users served) and attempt to maximize the minimum SNR among those served under the transmit power constraint. MDR, on the other hand, attempts to maximize coverage subject to received SNR and transmit power constraints, while ENUM yields a (usually tight) upper bound on the number of users that can be served under the same constraints. A meaningful way to compare all algorithms is via the respective minimum SNR - coverage curves, parameterized by transmit power P. This is analogous to the use of the Receiver Operating Characteristic (ROC) to compare different detectors.

We again consider two different wireless scenarios: measured outdoor, and measured indoor (i.i.d. Rayleigh simulations were also conducted, yielding consistent results, but these are omitted for brevity). In addition to instantaneous CSI-T, we also considered long-term CSI-T. For the latter, we estimated channel correlation matrices by averaging over 30 temporal snapshots; i.e., with $\mathbf{h}_{i,n}$ denoting the channel from the transmit antenna array to receiver i at time $n \in \{1, 2, \cdots, 30\}$, we used $\hat{\mathbf{H}}_i := \frac{1}{30} \sum_{n=1}^{30} \mathbf{h}_{i,n} \mathbf{h}_{i,n}^H$ in place of $\mathbf{H}_i$ for all algorithms.

In all experiments with a single multicast group, ENUM yielded the optimum rank-one solution of the original NP-hard problem in all cases except for those that correspond to full coverage (i.e., the received signal power requirements are low enough to ensure that everyone can be served - there is no need for admission control). This is interesting in itself, and it also suggests that ENUM is a tight upper bound in all cases where admission control is active.

The parameters used in the experiments were as follows: $N = 4$; $K = 10$ in experiments $3, 4$ and $K = 12$ for experiments $5, 6$; $P = 30$; $c_i = c$ and $\sigma_i^2 = \sigma^2 = 1$, $\forall i$; for MDR, $\epsilon = 10^{-10}$, $\delta < \min_{i \in \mathcal{U}} 4c^{-1}/\sigma_i^2$. For Lozano's and LLI algorithm, $\mu = 10^{-3}$. For dLLI, $\mu_0 = 1$ and back-off schedule number 6 is used. For all three adaptive algorithms, convergence is declared when the change in minimum SNR drops below 10^{-3}. These parameters were empirically tuned to optimize the performance of each algorithm.

Experiments 3 and 4 concern the 'Quad' measured outdoor scenario. Measurements corresponding to positions $1, 3, 4, 6, 7, 9, 12, 13, 15, 17$ in Figure 3.2 were selected for the respective ten users. Results for instantaneous CSI-T (experiment 3) are summarized in Figure 3.6, while those for long-term CSI-T (experiment 4) in Figure 3.7.

In Figure 3.6 (instantaneous CSI-T), MDR and dLLI perform very close to the optimum, while Lozano's algorithm is far behind. Specifically, the average coverage gap of Lozano's compared to MDR and dLLI is up to 5 users (50%) for a given average minimum SNR, while the average minimum SNR gap is up to 5 dB for a given average coverage. LLI's curve falls between Lozano's and MDR; dLLI significantly improves the performance of LLI. Unlike MDR and dLLI, the other two adaptive algorithms (Lozano's and LLI) do a poor job for full coverage. MDR and dLLI are on par performance-wise (MDR is somewhat better at lower coverage, dLLI at higher coverage); but dLLI is faster.

The situation is different for long-term CSI-T, as illustrated in Figure 3.7. Here dLLI is close to optimal throughout the coverage range *and* clearly outperforms the rest, including MDR.

Indoor measurements ('2nd Floor ECERF') were used for the next two experiments. Measurements corresponding to positions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 in Figure 3.3 were selected for the respective twelve users. Instantaneous CSI-T is used in experiment 5, whereas long-term CSI-T is used in experiment 6. As can be seen in Figure 3.8, MDR almost coincides performance-wise with ENUM, dLLI performs very close to MDR, while LLI keeps trailing both MDR and dLLI, by a significant margin. This picture changes (again) when long-term CSI-T is considered, in Figure 3.9: dLLI clearly outperforms all other algorithms, including MDR; but this time, dLLI is not as close to optimal as in the outdoors case.

Summarizing the insights obtained from single-group experiments, MDR and dLLI emerge as the clear winners (in light of the fact that ENUM is prohibitively complex for realistic values of K). Performance-wise, MDR is somewhat better for instantaneous CSI-T (rank-one channel correlations), especially in the higher SNR / lower coverage regime, whereas dLLI is clearly preferable for long-term CSI-T (higher rank channel correlations), and is generally close to MDR even for instantaneous CSI-T. The proposed modifications of Lozano's algorithm (LLI, dLLI) are simple, yet significantly boost performance. Complexity-wise, Lozano's algorithm together with LLI are the fastest ones, requiring typically 10^{-3} to 10^{-2} seconds to terminate, per problem instance. LLI is slightly faster than Lozano due to its better initialization, but

their run-times remain in the same order of magnitude. dLLI is somewhat slower, requiring on average 10^{-2} seconds. MDR follows next, requiring from 10^{-1} up to 1.5 seconds to run. dLLI is more than an order of magnitude faster than MDR in all cases considered. ENUM takes from 2 to 7 minutes per problem instance.

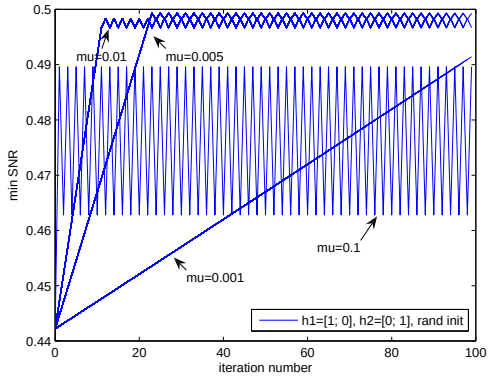

Figure 3.1: Illustration of the effect of μ on Lozano's algorithm for a contrived but instructive two-user scenario.

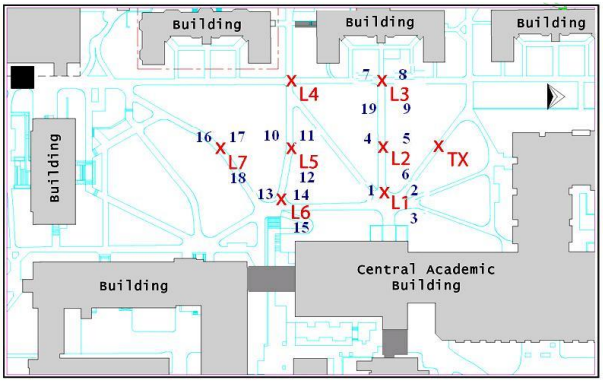

Figure 3.2: Quad: Outdoor measurement scenario from http://www.ece.ualberta.ca/∼mimo/

Figure 3.3: 2nd Floor of ECERF: Indoor office measurement scenario from http://www.ece.ualberta.ca/∼mimo/

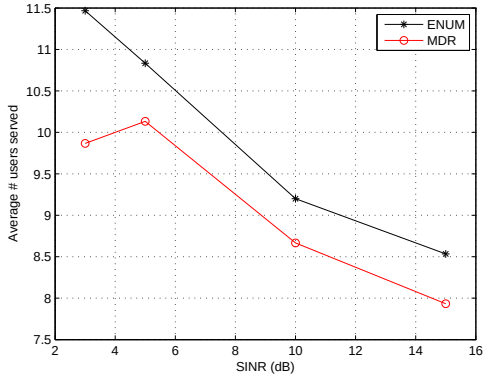

Figure 3.4: Experiment 1 (Outdoor): Average number of users served versus target SINR for 30 measured channel snapshots.

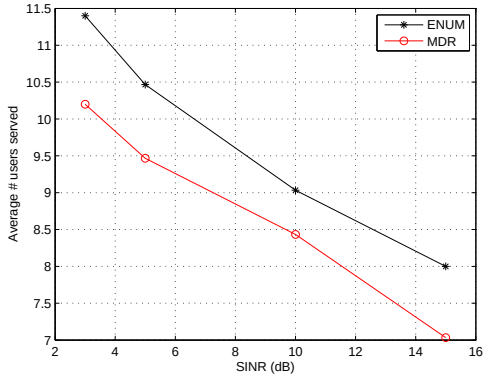

Figure 3.5: Experiment 2 (Indoor): Average number of users served versus target SINR for 30 measured channel snapshots.

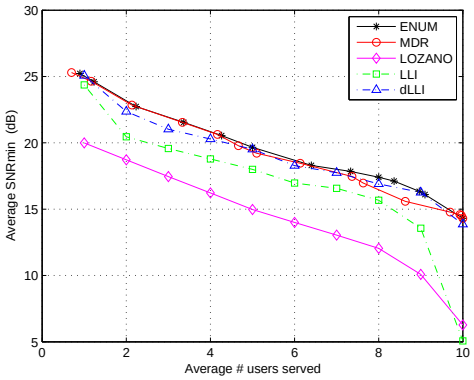

Figure 3.6: Experiment 3 (Outdoor, single multicast, instantaneous CSI-T): Average minimum SINR versus average number of users served over 30 measured channel snapshots.

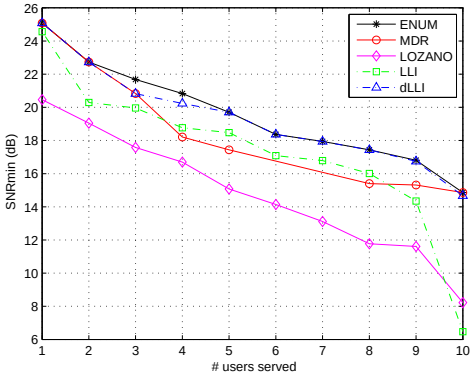

Figure 3.7: Experiment 4 (Outdoor, single multicast, long-term CSI-T): Minimum SINR versus number of users served.

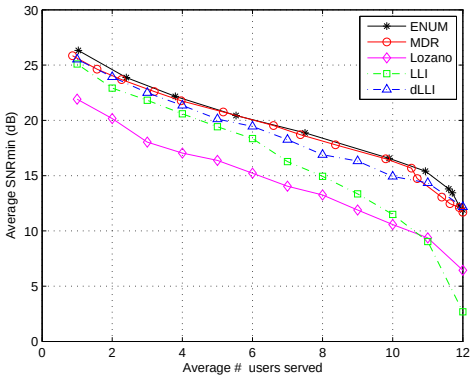

Figure 3.8: Experiment 5 (Indoor, single multicast, instantaneous CSI-T): Average minimum SINR versus average number of users served over 30 measured channel snapshots.

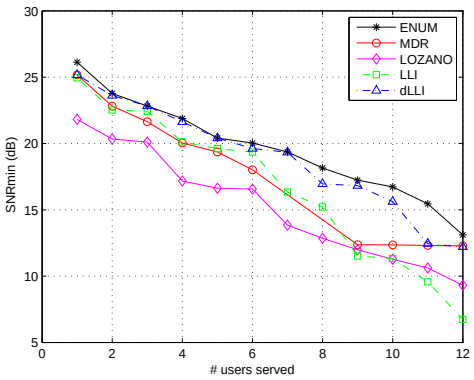

Figure 3.9: Experiment 6 (Indoor, single multicast, long-term CSI-T): Minimum SINR versus number of users served.

Table 3.1: Experiment 1: Stationary Outdoor, three multicast groups; Monte-Carlo results (30 measured channel snapshots): $N = 4$ Tx Ant., $K = 12$ users (all depicted in Fig. 3.2), $P = 1000$; $\sigma_k^2 = \sigma^2 = 1$, $c_k = c$, $\lambda_k = 1$, $\forall k$; $e = 10^{-9} < \frac{1}{P/4+1}$, $\delta = \frac{1/24c^{-1}}{P \max_m \|\mathbf{h}_m\|_2^2 + \sigma^2}$. Entries formatted as n @ $p/30$ mean that n users (or groups) are served in p out of 30 cases. All powers reported in linear scale.

QoS target	Alg.	♯ users served	♯ Groups	Avg. Min Tx Power	Max Min Tx Power	Avg. Time
3	ENUM	12 @ 10/30 11 @ 19/30 10 @ 1/30	3 @ 30/30	70	224	2960 s
3	D-SDR	12 @ 2/30 11 @ 6/30 10 @ 13/30 9 @ 4/30 8 @ 5/30	3 @ 29/30 2 @ 1/30	26	88	0.95 s
5	ENUM	11 @ 24/30 10 @ 6/30	3 @ 30/30	185	749	1064 s
5	D-SDR	11 @6/30 10 @ 22/30 9 @ 2/30	3 @ 30/30	122	846	0.88 s
10	ENUM	10 @ 6/30 9 @ 24/30	3 @ 30/30	146	774	535 s
10	D-SDR	10 @ 3/30 9 @ 14/30 8 @ 13/30	3 @ 24/30 2 @ 6/30	123	803	0.99 s
15	ENUM	9 @ 16/30 8 @ 14/30	3 @ 16/30 2@ 14/30	414	950	352 s
15	D-SDR	9 @1/30 8 @26/30 7 @ 3/30	3 @ 3/30 2 @ 27/30	153	447	0.79 s

Table 3.2: Experiment 2: Stationary Indoor, three multicast groups; Monte-Carlo results (30 measured channel snapshots): $N = 4$ Tx Ant., $K = 12$ users (all depicted in Fig. 3.3), $P = 1000$; $\sigma_k^2 = \sigma^2 = 1$, $c_k = c$, $\lambda_k = 1$, $\forall k$; $e = 10^{-9} < \frac{1}{P/4+1}$, $\delta = \frac{1/24c^{-1}}{P \max_m \|\mathbf{h}_m\|_2^2 + \sigma^2}$. Entries formatted as n @ $p/30$ mean that n users (or groups) are served in p out of 30 cases. All powers reported in linear scale.

QoS target	Alg.	♯ users served	♯ Groups	Avg. Min Tx Power	Max Min Tx Power	Avg. Time
3	ENUM	12 @ 12/30 11 @ 18/30	3 @ 30/30	76	432	479 s
3	MDR	12 @ 4/30 11 @ 7/30 10 @ 11/30 9 @ 7/30 8 @ 1/30	3 @ 30/30	51	422	0.40 s
5	ENUM	11 @ 14/30 10 @ 16/30	3 @ 30/30	101	388	308 s
5	MDR	11 @ 1/30 10 @ 15/30 9 @ 11/30 8 @ 3/30	3 @ 30/30	145	997	0.43 s
10	ENUM	10 @ 1/30 9 @ 29/30	3 @ 30/30	194	983	383 s
10	MDR	10 @ 15/30 8 @ 14/30 6 @ 1/30	3 @ 21/30 2 @ 9/30	338	981	0.45 s
15	ENUM	8 @ 30/30	3 @ 29/30 2 @ 1/30	296	692	330 s
1 15	MDR	8 @ 7/30 7 @ 17/30 6 @ 6/30	3 @ 2/30 2 @ 28/30	226	752	0.53 s

3.8 Conclusions

We have considered the problem of joint multicast beamforming and admission control for cellular and indoor/outdoor wireless networks. The objective is to serve as many potential subscribers as possible at or above their prescribed SINR, and minimize the power required to serve them. The problem is unfortunately NP-hard, but we have shown that it is possible to design approximate solutions of acceptable complexity.

Two distinct approaches have been developed: one extending our earlier work for the multiuser SDMA downlink; the other building upon Lozano's alternating gradient iteration. The former (MDR) is a batch algorithm based on convex approximation, and handles multiple co-channel multicast groups. The latter (dLLI) is an adaptive filtering-type algorithm that is restricted to a single multicast group. These algorithms were thoroughly tested in experiments using measured indoor and outdoor wireless channel data. These experiments indicate that MDR and dLLI are generally good and sometimes remarkably good low-complexity approximations for the problem at hand.

For a single multicast group, dLLI has better or comparable performance relative to MDR, in all cases considered. For long-term CSI-T, dLLI is the clear winner. Long-term CSI-T is more realistic in a cellular context, due to mobility and the desire to limit signaling overhead. The simplicity of dLLI is also very appealing. Taken together, these factors swing the verdict in favor of dLLI, at least for cellular applications and a single multicast group (multiple multicasts can be served via frequency- or time-division multiplexing, but this is generally not spectrally efficient). When multiple co-channel multicasts are considered, and/or in fixed wireless applications where instantaneous CSI-T is available, MDR is the method of choice.

3.9 Appendix

3.9.1 Proof of Claim 3

Feasibility: Using the Cauchy-Schwartz inequality, it is easy to show that $\mathbf{w}_m = \mathbf{0}$, $\forall m$, $s_i = 1$, $\forall i$ is always feasible for (3.9)-(3.11), provided

$$\delta \leq \min_{m\in\{1,\cdots,G\}} \min_{i\in\mathcal{G}_m} \frac{4c_i^{-1}}{P \max_{n\in\{1,\cdots,K\}} \|\mathbf{h}_n\|_2^2 + \sigma_i^2}. \tag{3.35}$$

Optimality: We next show that under the additional condition

$$\epsilon < \frac{1}{P/4+1}, \tag{3.36}$$

the single-stage reformulation in (3.9)–(3.11) is equivalent to the two-stage problem in (3.4)-(3.8). The proof is by contradiction. Let $\left\{\check{\mathbf{w}}_m, \{\check{s}_i \in \{-1,+1\}\}_{i\in\mathcal{G}_m}\right\}_{m=1}^G$ be a solution of (3.9)-(3.11), and let $\left\{\tilde{\mathbf{w}}, \{\tilde{s}_i \in \{-1,+1\}\}_{i\in\mathcal{G}_m}\right\}_{m=1}^G$ denote a feasible alternative [that satisfies (3.10)-(3.11)] for which

$$\sum_{m=1}^G \sum_{i\in\mathcal{G}_m} 1(\tilde{s}_i = -1) > \sum_{m=1}^G \sum_{i\in\mathcal{G}_m} 1(\check{s}_i = -1),$$

where $1(\cdot)$ stands for the indicator function. It follows that

$$\sum_{m=1}^G \sum_{i\in\mathcal{G}_m} (\tilde{s}_i + 1)^2 \leq \sum_{m=1}^G \sum_{i\in\mathcal{G}_m} (\check{s}_i + 1)^2 - 4,$$

so

$$\epsilon \sum_{m=1}^G \|\tilde{\mathbf{w}}_m\|_2^2 + (1-\epsilon) \sum_{m=1}^G \sum_{i\in\mathcal{G}_m} (\tilde{s}_i + 1)^2 \leq$$

$$\epsilon P + (1-\epsilon) \sum_{m=1}^G \sum_{i\in\mathcal{G}_m} (\check{s}_i + 1)^2 - (1-\epsilon)4.$$

Now, $\epsilon < \frac{1}{P/4+1} \Leftrightarrow \epsilon P - (1-\epsilon)4 < 0$, therefore

$$\epsilon \sum_{m=1}^G \|\tilde{\mathbf{w}}_m\|_2^2 + (1-\epsilon) \sum_{m=1}^G \sum_{i\in\mathcal{G}_m} (\tilde{s}_i + 1)^2 <$$

$$(1-\epsilon) \sum_{m=1}^G \sum_{i\in\mathcal{G}_m} (\check{s}_i + 1)^2 \leq$$

$$\epsilon \sum_{m=1}^G \|\check{\mathbf{w}}_m\|_2^2 + (1-\epsilon) \sum_{m=1}^G \sum_{i\in\mathcal{G}_m} (\check{s}_i + 1)^2,$$

which is a contradiction.

This shows that serving more than $\sum_{m=1}^{G}\sum_{i\in\mathcal{G}_m} 1(\check{s}_i=-1)$ users is impossible. It remains to show that users whose $\check{s}_i=-1$ are served by $\left\{\check{\mathbf{w}}_m\in\mathbb{C}^N\right\}_{m=1}^{G}$ at minimum power.

For this, notice that joint optimality of $\left\{\check{\mathbf{w}}_m,\{\check{s}_i\in\{-1,+1\}\}_{i\in\mathcal{G}_m}\right\}_{m=1}^{G}$ implies conditional optimality of $\{\check{\mathbf{w}}_m\}_{m=1}^{G}$ given $\{\check{s}_i\}$ - for otherwise we would again have a contradiction. For given $\{\check{s}_i\}$, and denoting $S_o:=\cup_{i\ |\ \check{s}_i=-1}$, note that $\mathcal{G}_m\cap S_o=\emptyset\Rightarrow\check{\mathbf{w}}_m=\mathbf{0}_{N\times 1}$. This comes from the cost function and constraints in (3.9)–(3.11): if the constraints for a certain subset of users are satisfied via the slack variables, allocating power to these users is wasteful (increases the cost function) and simply adds interference to other users. It follows that, conditioned on $\{\check{s}_i\}$, the $\{\check{\mathbf{w}}_m\}_{m=1}^{G}$ should minimize $\epsilon\sum_{m\ |\ \mathcal{G}_m\cap S_o\neq\emptyset}\|\mathbf{w}_m\|_2^2+\text{constant}$ under $\sum_{m\ |\ \mathcal{G}_m\cap S_o\neq\emptyset}\|\mathbf{w}_m\|_2^2\leq P$, and

$$\frac{|\mathbf{w}_m^H\mathbf{h}_i|^2}{\sum_{\ell\ |\ \mathcal{G}_\ell\cap S_o\neq\emptyset,\ell\neq m}|\mathbf{w}_\ell^H\mathbf{h}_i|^2+\sigma_i^2}\geq c_i,\ \forall i\in\mathcal{G}_m\cap S_o,\ \forall m. \tag{3.37}$$

Finally, note that if there are multiple solutions of (3.4)-(3.6), i.e., if the maximal subset of users that can be served is not unique, then (3.9)–(3.11) will automatically pick a maximal subset requiring minimal total power - for otherwise a contradiction would emerge: cf. (3.9), and note that the second term is only a function of the number of users served. This completes the proof.

■

3.9.2 Further simplifications

The reformulated problem in (3.9)-(3.11) can be further simplified as follows

$$\min_{\left\{\mathbf{w}_m\in\mathbb{C}^N,\{s_i\in\{-1,+1\}\}_{i\in\mathcal{G}_m}\right\}_{m=1}^{G}}\mathcal{J}\left(\left\{\mathbf{w}_m,\{s_i\}_{i\in\mathcal{G}_m}\right\}_{m=1}^{G}\right):=$$

$$\epsilon\sum_{m=1}^{G}\|\mathbf{w}_m\|_2^2+(1-\epsilon)2\sum_{m=1}^{G}\sum_{i\in\mathcal{G}_m}(s_i+1) \tag{3.38}$$

$$\text{subject to}:\ \sum_{m=1}^{G}\|\mathbf{w}_m\|_2^2\leq P, \tag{3.39}$$

$$\frac{|\mathbf{w}_m^H\mathbf{h}_i|^2+\delta^{-1}2(s_i+1)}{\sum_{\ell\neq m}|\mathbf{w}_\ell^H\mathbf{h}_i|^2+\sigma_i^2}\geq c_i,\ \forall i\in\mathcal{G}_m,\ \forall m,\ell\in\{1,\cdots,G\}, \tag{3.40}$$

since for $s_i\in\{-1,+1\}$ it holds that $(s_i+1)^2=s_i^2+2s_i+1=2(s_i+1)$. This way, the quadratic penalty term in the objective and the constraints can be equivalently replaced by the linear term $2(s_i+1)$. In a similar vain, the semidefinite relaxation in (3.17)-(3.21) can be equivalently

rewritten as

$$\min_{\left\{\mathbf{W}_m\in\mathbb{C}^{N\times N},\{s_i\in\mathbb{R}\}_{i\in\mathcal{G}_m}\right\}_{m=1}^{G}} \phi\left(\left\{\mathbf{W}_m,\{s_i\}_{i\in\mathcal{G}_m}\right\}_{m=1}^{G}\right) =$$

$$\epsilon\sum_{m=1}^{G}\mathrm{Tr}(\mathbf{W}_m)+(1-\epsilon)2\sum_{m=1}^{G}\sum_{i\in\mathcal{G}_m}(s_i+1) \tag{3.41}$$

$$\text{subject to}: \sum_{m=1}^{G}\mathrm{Tr}(\mathbf{W}_m)\leq P, \tag{3.42}$$

$$\frac{\mathrm{Tr}(\mathbf{H}_i\mathbf{W}_m)+\delta^{-1}2(s_i+1)}{\sum_{\ell\neq m}\mathrm{Tr}(\mathbf{H}_i\mathbf{W}_\ell)+\sigma_i^2}\geq c_i,\ \forall i\in\mathcal{G}_m,\ \forall m,\ell\in\{1,\cdots,G\} \tag{3.43}$$

$$\mathbf{W}_m\geq 0,\ \forall m, \tag{3.44}$$

$$-1\leq s_i\leq 1,\ \forall i\in\mathcal{G}_m,\ \forall m. \tag{3.45}$$

which avoids the introduction of 2×2 positive semidefinite matrix variables $\mathbf{S}_i$, using instead scalar variables. This helps speed up computations, by roughly a factor of two.

Bibliography

[1] M. Bengtsson and B. Ottersten, "Optimal and Suboptimal Transmit Beamforming," ch. 18 in *Handbook of Antennas in Wireless Communications*, L. C. Godara, Ed., CRC Press, Aug. 2001.

[2] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*, 2nd ed. Belmont, MA: Athena Scientific, 1996.

[3] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, 2nd ed. Belmont, MA: Athena Scientific, 1999.

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Foundations & Trends in Machine Learning*, Vol. 3, No 1, pp.1-122, 2011.

[5] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
See also http://www.stanford.edu/∼boyd/cvxbook.html.

[6] R. Bro, N.D. Sidiropoulos, and A.K. Smilde, "Maximum Likelihood Fitting Using Ordinary Least Squares Algorithms," *J. of Chemometrics*, vol. 16, pp. 387-400, Sep. 2002.

[7] R. Cendrillon, W. Yu, M. Moonen, J. Verliden, and T. Bostoen, "Optimal multi-user spectrum management for digital subscriber lines," *IEEE Trans. on Communications*, vol. 54, no. 5, pp. 922-933, May 2006.

[8] P. Chaporkar, K. Kar, and S. Sarkar, "Throughput Guarantees in Maximal Scheduling in Wireless Networks," in *Proc. 43rd Annual Allerton Conference on Communication, Control and Computing*, pp. 1557-1567, Allerton, IL, Sep. 2005.

[9] M. Chiang, C. W. Tan, D. P. Palomar, D. O'Neill, and D. Julian, "Power Control By Geometric Programming," *IEEE Trans. on Wireless Communications*, vol. 6, no. 7, pp. 2640-2651, 2007.

[10] J. Cortes, "Distributed algorithms for reaching consensus on general functions," *Automatica*, vol. 44, no. 3, pp. 726-737, 2008.

[11] G. Dimic, N.D. Sidiropoulos, "On Downlink Beamforming with Greedy User Selection: Performance Analysis and a Simple New Algorithm," *IEEE Trans. on Signal Processing*, vol. 53, no. 10, pp. 3857-3868, Oct. 2005.

[12] D. Dolev, A. Zymnis, S.P. Boyd, D. Bickson, and Y. Tock, "Distributed large scale network utility maximization," in *Proc. IEEE International Symposium on Information Theory (ISIT) 2009*, pp. 829-833, Seoul, Korea, June 28 - July 3, 2009.

[13] F.-R. Farrokhi, K.R. Liu, and L. Tassiulas, "Transmit Beamforming and Power Control for Cellular Wireless Systems," *IEEE J. Select. Areas Commun.*, vol. 16, p. 1437, October 1998.

[14] L. Georgiadis, M. Neely, L. Tassiulas, "Resource Allocation and Cross-Layer Control in Wireless Networks," *Foundations & Trends in Networking*, Vol. 1, pp. 1-147, 2006.

[15] A. Giannoulis, K. Tsoukatos, and L. Tassiulas, "Maximum throughput power control in CDMA wireless networks," in *Proc. IEEE International Conference on Communications*, pp. 4457-4462, Instabul, Turkey, June 2006.

[16] P. Goud Jr., R. Hang, D. Truhachev, and C. Schlegel, "A Portable MIMO Testbed and Selected Channel Measurements," *EURASIP Journal on Applied Signal Processing*, vol. 2006, Article ID 51490, 10 pages, 2006. doi:10.1155/ASP/2006/51490

[17] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming," web page and software: http://stanford.edu/ boyd/cvx

[18] B. S. He, H. Yang, and S. L. Wang, "Alternating Direction Method with Self-Adaptive Penalty Parameters for Monotone Variational Inequalities," in *Journal of Optimization Theory and Applications*, vol. 106, no. 2, pp. 337-356, Aug. 2000.

[19] A. Jadbabaie, A. Ozdaglar, and M. Zargham, "A Distributed Newton Method for Network Optimization," in *Proc. IEEE Conference on Decision and Control*, pp. 2736-2741, Shanghai, China, Dec. 2009.

[20] C. Joo, X. Lin, and N. B. Shroff, "Performance Limits of Greedy Maximal Matching in Multi-hop Wireless Networks," in *Proc. IEEE Conference on Decision and Control*, pp. 1128-1133, December 12-14, New Orleans, LA, 2007.

[21] E. Karipidis, "Transmit Beamforming to Multiple Cochannel Multicast Groups," Ph.D. dissertation [chapter 4], Dept. of ECE, Technical University of Crete, Aug. 2008.

[22] E. Karipidis, N.D. Sidiropoulos, and Z.-Q. Luo, "Quality of Service and Max-min-fair Transmit Beamforming to Multiple Co-channel Multicast Groups," *IEEE Trans. on Signal Processing*, vol. 56, no. 3, pp. 1268-1279, Mar. 2008.

[23] E. Karipidis, N.D. Sidiropoulos, Z.-Q. Luo, "Far-field Multicast Beamforming for Uniform Linear Antenna Arrays," *IEEE Trans. on Signal Processing*, vol. 55, no. 10, pp. 4916-4927, Oct. 2007.

[24] X. Lin and N. B. Shroff, "The Impact of Imperfect Scheduling on Cross-Layer Congestion Control in Wireless Networks," *IEEE/ACM Transactions on Networking*, vol. 14, no. 2, pp. 302315, April 2006.

[25] M. J. Lopez, "Multiplexing, scheduling, and multicasting strategies for antenna arrays in wireless networks," Ph.D. thesis, Dept. of Elect. Eng. and Comp. Sci., MIT, Cambridge, MA, 2002.

[26] A. Lozano, "Long-Term Transmit Beamforming for Wireless Multicasting," in *Proc. ICASSP 2007*, Apr. 15-20, Honolulu, Hawaii.

[27] Z.-Q. Luo, N.D. Sidiropoulos, P. Tseng, P., and S. Zhang, "Approximation Bounds for Quadratic Optimization with Homogeneous Quadratic Constraints," *SIAM Journal on Optimization*, vol. 18, no. 1, pp. 1-28, 2007.

[28] Z.-Q. Luo and W. Yu, "An introduction to convex optimization for communications and signal processing," *IEEE JSAC*, vol. 24, no. 8, pp. 1426-1438, August 2006.

[29] Z.-Q. Luo and S. Zhang, "Dynamic Spectrum Management: Complexity and Duality," *IEEE J. Selected Topics in Signal Processing*, vol. 2, no. 1, pp. 57-73, Feb. 2008.

[30] Z.-Q. Luo, private communication with N.D. Sidiropoulos, University of Minnesota, July 2009.

[31] W.-K. Ma, T.N. Davidson, K.M. Wong, Z-Q. Luo, and P.-C. Ching, "Quasi-ML Multiuser Detection Using Semi-Definite Relaxation with Application to Synchronous CDMA," *IEEE Transactions on Signal Processing*, vol. 50, no. 4, pp. 912-922, Apr. 2002.

[32] E. Matskani, N.D. Sidiropoulos, and L. Tassiulas, "Convex Approximation Algorithms for Back-pressure Power Control of Wireless Multi-hop Networks," in *Proc. IEEE ICASSP 2011*, pp. 3032-3035, Prague, Czech Republic, May 22-27, 2011.

[33] E. Matskani, N.D. Sidiropoulos, and L. Tassiulas, "Distributed Back-pressure Power Control for Wireless Multi-hop Networks," in *Proc. IEEE ICASSP 2012*, Kyoto, Japan, March 25-30, 2012.

[34] E. Matskani, N.D. Sidiropoulos, and L. Tassiulas, "Convex Approximation Algorithms for Back-pressure Power Control," *IEEE Transactions on Signal Processing*, vol. 60, no. 4, pp. 1957-1970, April 2012.

[35] E. Matskani, N.D. Sidiropoulos, Z.-Q. Luo, and L. Tassiulas, "A Second-order Cone Deflation Approach to Joint Multiuser Downlink Beamforming and Admission Control," in *Proc. SPAWC 2007*, June 17-20, Helsinki, Finland.

[36] E. Matskani, N.D. Sidiropoulos, Z.-Q. Luo, and L. Tassiulas, "Convex Approximation Techniques for Joint Multiuser Downlink Beamforming and Admission Control," *IEEE Trans. on Wireless Communications*, vol. 7, no. 7, pp. 2682-2693, July 2008.

[37] E. Matskani, N.D. Sidiropoulos, Z.-Q. Luo, and L. Tassiulas, "Efficient Batch and Adaptive Approximation Algorithms for Joint Multicast Beamforming and Admission Control," *IEEE Trans. on Signal Processing*, vol. 57, no. 12, pp. 2682-2693, Dec. 2009.

[38] E. Matskani, N.D. Sidiropoulos, Z.-Q. Luo, and L. Tassiulas, "Joint Multicast Beamforming and Admission Control," in *Proc. CAMSAP 2007*, December 12-14, St. Thomas, U.S. Virgin Islands.

[39] E. Matskani, N.D. Sidiropoulos, and L. Tassiulas, "A Comparison of Multicast Beamforming Algorithms for UMTS-LTE," in *Proc. ICASSP 2008*, April 12-14, Las-Vegas, Nevada.

[40] Motorola Inc., "Long Term Evolution (LTE): A Technical Overview," Technical White Paper, http://business.motorola.com/experiencelte/pdf/LTE%20Technical%20Overview.pdf

[41] M. J. Neely, "Energy optimal control for time varying wireless networks," *IEEE Transactions on Information Theory*, vol. 52, no. 7, July 2006.

[42] M. J. Neely, E. Modiano, and C.-P. Li, "Fairness and optimal stochastic control for heterogeneous networks," in *Proc. IEEE INFOCOM*, pp. 1723-1734, Miami, Florida, USA, March 13-17, 2005.

[43] D. P. Palomar, and Mung Chiang, "A Tutorial on Decomposition Methods for Network Utility Maximization," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1439-1451, Aug. 2006.

[44] D. P. Palomar, and Mung Chiang, "Alternative Distributed Algorithms for Network Utility Maximization: Framework and Applications," *IEEE Transactions on Automatic Control*, vol. 52, no. 12, pp. 2254-2269, Dec. 2007.

[45] J. Papandriopoulos and J.S. Evans, "Low-Complexity Distributed Algorithms for Spectrum Balancing in Multi-User DSL Networks," in *Proc. IEEE International Conference on Communications*, pp. 3270-3275, Instabul, Turkey, June 2006.

[46] J. Papandriopoulos and J.S. Evans, "SCALE: A Low-Complexity Distributed Protocol for Spectrum Balancing in Multiuser DSL Networks," *IEEE Trans. Information Theory*, vol. 55, no. 8, pp. 3711-3724, Aug. 2009.

[47] L. P. Qian, Y. J. Zhang, and J. Huang, "Mapel: Achieving global optimality for a non-convex wireless power control problem," *IEEE Trans. on Wireless Communications*, vol. 8, no. 3, pp. 1553-1563, Mar. 2009.

[48] S. Sarkar and L. Tassiulas, "A framework for routing and congestion control for multicast information flows," *IEEE Transactions on Information Theory,* vol. 48, no. 10, pp. 2690-2708, October 2002.

[49] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, "Consensus in *Ad Hoc* WSNs With Noisy Links- Part I: Distributed Estimation of Deterministic Signals," *IEEE Trans. Signal Processing*, vol. 56, no.1, pp. 350-364, Jan. 2008.

[50] Q. Shi, M. Razaviyayn, Z.-Q. Luo, and C. He, "An Iteratively Weighted MMSE Approach to Distributed Sum-Utility Maximization for a MIMO Interfering Broadcast Channel," *IEEE Trans. on Signal Processing*, vol. 59, no. 9, pp. 4331-4340, Sep. 2011.

[51] N.D. Sidiropoulos, T.N. Davidson, and Z-Q. Luo, "Transmit Beamforming for Physical Layer Multicasting," *IEEE Trans. on Signal Processing*, vol. 54, no. 6, part 1, pp. 2239-2251, June 2006.

[52] A. L. Stolyar, "Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm," *Queueing Systems*, vol. 50, pp. 401-457, 2005.

[53] L. Tassiulas, "Scheduling and Performance Limits of Networks with Constantly Changing Topology," *IEEE Trans. Inf. Theory*, vol. 43, no. 3, pp. 1067–1073, 1997

[54] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Autom. Control*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.

[55] L. Tassiulas, "Adaptive back-pressure congestion control based on local information," *IEEE Trans. Autom. Control*, vol. 40, no. 2, pp. 236–250, Feb. 1995.

[56] L. Tassiulas, "Linear complexity algorithms for maximum throughput in radio networks and input queued switches," *Proceedings of IEEE INFOCOM*, vol. 2, pp. 533-539, San Francisco, CA, Mar. 29 - Apr. 2, 1998.

[57] S. L. Wang and L. Z. Liao, "Decomposition Method with a Variable Parameter for a Class of Monotone Variational Inequality Problems," in *Journal of Optimization Theory and Applications*, vol. 109, no. 2, pp. 415-429, May 2001.

[58] H. Wolkowicz, "Relaxations of Q2P," Chapter 13.4 in *Handbook of Semidefinite Programming: Theory, Algorithms, and Applications*, H. Wolkowicz, R. Saigal, L. Vandenberghe (Eds.), Kluwer Academic Publishers, 2000.

[59] Y. Ye, *Interior Point Algorithms: Theory and Analysis*, Wiley, 1997.

[60] W. Yu and R. Lui, "Dual Methods for Nonconvex Spectrum Optimization of Multicarrier Systems," *IEEE Trans. on Communications*, vol. 54, no. 7, pp. 1310-1322, July 2006.