Signal Processing 104 (2014) 437–449

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

Comparative simulation study of fast heuristics for power control in copper broadband networks

Martin Wolkerstorfer [a,*], Tomas Nordström [a,b]

[a] FTW Telecommunications Research Center Vienna, Donau-City-Straße 1, A-1220 Vienna, Austria
[b] Centre for Research on Embedded Systems (CERES), Halmstad University, Box 823, SE-30118 Halmstad, Sweden

ARTICLE INFO

Article history:
Received 11 November 2013
Received in revised form
28 April 2014
Accepted 1 May 2014
Available online 10 May 2014

Keywords:
Digital subscriber lines
Power control
Meta-heuristics

ABSTRACT

The data-rate in currently deployed multi-carrier digital subscriber line (DSL) communication systems is limited by the interference among copper lines. This interference can be alleviated by multi-user transmit power allocation. Problem decomposition results in a large number of per-subcarrier problems. Our objective is to solve these nonconvex integer per-subcarrier power control problems at low complexity. For this purpose we develop ten combinatorial heuristics and test them by simulation under a small complexity budget in scenarios with tens of DSL users, where optimal solutions are currently intractable. Simulation results lead us to the conclusion that simple randomized greedy heuristics extended by a specific local search perform well despite the stringent complexity restriction. This has implications on multi-user discrete resource allocation algorithms, as these can be designed to *jointly* optimize transmit power among users even in large-scale scenarios.

1. Introduction

In 2012 over 360 million customers world-wide utilized digital subscriber lines (DSL), making it the most widely deployed fixed broadband access technology [1]. While each user has at least one dedicated copper line, multi-carrier DSL systems still suffer from the electromagnetic coupling (or "interference") among the twisted copper pairs. The effect is noise at the receiver, which limits the achievable data-rate and increases the energy consumption per transmitted data-bit. We study the nonconvex integer multi-user problem of controlling the power levels transmitted by all users on the subcarriers in order to lower interference, increase the number of transmitted data-bits, and reduce the transmit-power consumption (implicitly lowering the system power consumption [2]). The objective is formulated as a weighted sum of users' transmit powers and bits, constrained by technology and regulatory restrictions. This fundamental problem also finds applications in wireless networks [3–6].

The main contribution of this study lies in the comparative simulation experiments including ten combinatorial heuristics for large-scale discrete single-subcarrier power control with a *low complexity budget*. We propose a mixture of deterministic and stochastic heuristics, the latter comprising more direct applications of meta-heuristics as well as greedy schemes modified based on problem insights. These insights are derived from an analysis of suboptimality of these deterministic greedy algorithms, and motivate for example randomization-based modifications. Preliminary simulation results have appeared in [7,8]. However, the present study specifically focusses on the single-subcarrier problem and additionally provides full descriptions of all proposed heuristics, their parameter settings, as well as the detailed simulation

* Corresponding author. Tel.: +43 1 5052830x27,22; fax: +43 1 505283099.
E-mail addresses: wolkerstorfer@ftw.at (M. Wolkerstorfer), nordstrom@ftw.at, Tomas.Nordstrom@hh.se (T. Nordström).

http://dx.doi.org/10.1016/j.sigpro.2014.05.004
0165-1684/

results on an extensive set of 6 and 30-user DSL scenarios. While previously stochastic search heuristics may have been considered too complex for power control on a large number of subcarriers, our results show that simple randomized greedy heuristics enhanced by first-improving local search perform well even in case of a very stringent complexity budget, and that the heuristics' average suboptimality is implicitly dependent on the targeted data-rate.

We begin in Sections 2, 3.1, and 3.2 by reviewing some relevant literature, stating our notation, and reviewing the common nonconvex integer optimization model for power control in DSL, respectively. The "no free lunch" theorem states that, under specific assumptions on the problem domain and applied algorithms, any two algorithms have the same mean performance when compared over all possible problems [9] – we refer to [9] for a precise statement of this result. Hence, optimization heuristics cannot be regarded as superior in general, but should rather be evaluated for the specific problem domain based on a representative problem set. Generic optimization heuristics evaluated on an insufficient problem set may hence perform very differently in practice than what the simulation of the problem set would have suggested. Therefore problem-specific insights shall be exploited in the design of optimization heuristics, and a representative sampling of problem instances is needed in order to judge the performance of heuristics in a specific problem domain. Consequently, we introduce the three generic search principles applied throughout the paper in Section 4, review three basic search schemes from [10] following these search principles in Section 4.1, and in Section 4.2 analyze their performance on a selection of 84 medium-scale problems where optimal solutions are tractable. A branch-and-bound algorithm and a general-purpose mixed-integer nonconvex problem solver [11,12] are applied to generate the optimal reference solutions. It turns out that a specific problem feature (the well-known "near–far problem") leads to a high suboptimality of two greedy base-heuristics. The insights gained thereby as well as general meta-heuristics are applied in Section 5 to complement the set of, in total, ten combinatorial search heuristics for power control. For example, in Section 5.1 the greedy decisions of loading bits are randomized, while in Section 5.2.1 the *sequence* of greedy per-user decisions is randomized. The simulation setup and the methodology for the parameterization and performance evaluation of the heuristics in medium-scale and large-scale problems are described in Sections 6.1–6.4, respectively. The discussion of results in Section 6.5 is followed by our conclusions in Section 7.

2. Background on power control in DSL

Multi-carrier power control can be modeled as a multi-dimensional nonlinear Knapsack problem [13] which has motivated the application of Lagrange decomposition [14,15]. Thereby one obtains a large number of integer per-subcarrier subproblems which have to be solved numerous times for updating the Lagrange multipliers, and re-optimized when the DSL network changes. This motivates the tight complexity budget we impose upon their solution, where for reproducibility we opt for the number of function evaluations as our complexity metric, cf. Section 6.1 for its relation to simulation time. We explicitly focus on these subproblems only, where the objective is a weighted sum of data-bits and transmit power, cf. the large applicability of this problem [10,15]. Optimal algorithms are only tractable for medium-scale networks with few-users [11,16,17]. However, the low interference coupling compared to the direct coupling over a subscriber line that is typical for DSL [18] makes us believe that simple stochastic search heuristics may perform near-optimal even when constrained to a low number of function evaluations. Efficient state-of-the-art methods for power control in large-scale DSL networks solve a continuous relaxation only or violate the integer bit-loading restriction by iterating over users [19,20]. The nonlinear Dantzig–Wolfe decomposition approach in [10] allows for suboptimal solutions of the per-subcarrier problems, making the application of combinatorial search schemes as dual heuristics attractive. Furthermore, combinatorial heuristics could be applied on top of any previously proposed (e.g., continuous) power allocation algorithm in order to improve a rounded preliminary solution.

3. System and optimization model

3.1. System model and notation

We consider a communication network with U DSL users, and refer to [18] for a more detailed description of the technical background on DSL. By adequate modulation techniques the frequency bandwidth is subdivided into C regularly spaced and mutually exclusive frequency subcarriers. The index sets of users and subcarriers are $\mathcal{U}=\{1,\ldots,U\}$ and $\mathcal{C}=\{1,\ldots,C\}$, respectively. Our optimization variables are the power levels p_u^c applied by user u and subcarrier c. The achievable number of data-bits per channel-access of user u on subcarrier c is modeled as a, in general, nonconvex function $r_u^c(\mathbf{p}^c)$ [bits] [18] that depends on the transmit powers of all users. The transmit powers and data-bits over all users on subcarrier c are more compactly written as $\mathbf{p}^c\in\mathcal{R}_+^U$ and $\mathbf{r}^c(\mathbf{p}^c)\in\mathcal{R}_+^U$, respectively. The inverse function $\mathbf{p}^c(\mathbf{r}^c)$ for a bit allocation $\mathbf{r}^c$ is uniquely computable [21] by solving a system of linear equations of size $U\times U$. A publicly available tool for DSL performance evaluation can be found in [22]. We refer to Section 6.1 for a description of the network scenarios considered in this work.

Transmit power levels are constrained by a regulatory power mask constraint $p_u^c\le\hat{p}_u^c,\ \forall u\in\mathcal{U},\ c\in\mathcal{C}$, preventing excessive disturbance among competing DSL operators. Furthermore, the implicit constraint $r_u^c(\mathbf{p}^c)\in\mathcal{B},\forall u\in\mathcal{U}$, $c\in\mathcal{C}$, is motivated by practical modulation schemes which only support an integer number of data-bits in the set $\mathcal{B}=\{1,\ldots,\hat{B}\}$, with a technology-dependent maximum number of data-bits $\hat{B}$ per subcarrier. The feasible set of transmit powers is summarized as $\mathcal{Q}^c=\{\mathbf{p}^c|r_u^c(\mathbf{p}^c)\in\mathcal{B},\ 0\le p_u^c\le\hat{p}_u^c,\forall u\in\mathcal{U}\},\ c\in\mathcal{C}$. Our objective is defined as the weighted sum of transmit powers and data-bits

over all users

$$\bar{f}_c^{\mathbf{w}}(\mathbf{p}^c) = \sum_{u \in \mathcal{U}} \hat{w}_u p_u^c - \check{w}_u r_u^c(\mathbf{p}^c), \quad c \in \mathcal{C}, \tag{1}$$

where the weights $\mathbf{w}^T = [\hat{w}_1, \ldots, \hat{w}_U, \check{w}_1, \ldots, \check{w}_U] \in \mathcal{R}_+^U$ allow us to trade-off between data-rate and energy optimization, and also capture Lagrange dual variables in large-scale power control frameworks [10,15].

3.2. Problem description

The multi-user, non-convex and integer power control problem in DSL on subcarrier $c \in \mathcal{C}$ is that of maximizing the number of users' data-bits and minimizing their transmit powers, constrained by the technology and regulatory restrictions $\mathcal{Q}^c$, formulated as [10]

$$\underset{\{\mathbf{r}^c | \mathbf{p}^c(\mathbf{r}^c) \in \mathcal{Q}^c\}}{\text{minimize}} \; f_c^{\mathbf{w}}(\mathbf{r}^c) = \bar{f}_c^{\mathbf{w}}(\mathbf{p}^c(\mathbf{r}^c)), \tag{2}$$

which is combinatorial only in a wide sense [23, Section 4.4] due to the existence of a continuous (yet nonconvex) relaxation. Furthermore, this problem is always feasible due to the feasibility of $\mathbf{r} = \mathbf{0}$. We select the integer bit allocation $\mathbf{r}^c$ as our variables instead of the uniquely coupled power allocation $\mathbf{p}^c$ used above, which simplifies the later algorithm design and reflects the bit-loading [18] process applied in practice. Another alternative choice would have been to consider both, a discrete number of data-bits *and* continuous power variables, as required by the general-purpose mixed-integer solver in [11] applied in Section 6. In the rest of the paper we study the approximate solution of the problem in (2) and therefore omit subcarrier indices c for the ease of notation, based on the understanding that different problem instances may reflect different subcarriers. The considered search space is $\times_{u \in \mathcal{U}} \mathcal{B} = \{\mathbf{r} | r_u \in \mathcal{B}, \forall u \in \mathcal{U}\}$, that is, the constraint $\mathbf{p}^c(\mathbf{r}) \in \mathcal{Q}^c$ is not explicitly taken into account. The objective value of an allocation $\mathbf{r}$ violating this constraint is by definition infinite, which prevents our later proposed algorithms from traversing infeasible search regions. As previously mentioned, objective evaluation and feasibility checking necessitate the solution of a linear system of equations [18]. Hence, as our focus is on *computational* complexity and reproducibility we will use the number of objective evaluations as our complexity metric/stopping criterion for the comparison of search heuristics instead of runtime or memory complexity.

The optimal solution of the problem in (2) was shown to have polynomial complexity in [10]. However, an integer solution for over ten users was found intractable for problem-specific as well as general-purpose branch-and-bound-based schemes [16,17]. Furthermore, the number of these per-subcarrier problems is in the order of thousands in the newest generations of DSL technology, and is expected to be re-optimized frequently upon changes in the DSL network (e.g., when DSL users turn their modems off). Altogether this motivates the following work on fast heuristics.

4. Search principles, base heuristics, and analysis

In order to provide a self-contained description and motivation of the later proposed stochastic heuristics we will first review the base-heuristics suggested in [10], classify them into three categories as illustrated in Fig. 1, and analyze their performance on a representative set of DSL power control problems. The first type in Fig. 1(a) represents constructive schemes where a feasible solution $\mathbf{r}$ to a subproblem in (2) is built up by iteratively adding one bit to the already allocated data-bits r_u of one of the users $u \in \mathcal{U}$. Constructive sequential decision algorithms as depicted in Fig. 1(b) build up a complete solution $\mathbf{r}$ by sequentially allocating r_1 data-bits to user $u = 1$, r_2 data-bits to user $u = 2$, and so forth. The third type of heuristics represents local search heuristics where an allocation $\mathbf{r}$ is improved by searching for an allocation with better objective value in the local "environment" of $\mathbf{r}$, where the exact meaning of this environment will be introduced shortly, cf. Fig. 1(c).

4.1. Base heuristics

4.1.1. Joint Greedy Optimization (JOGO)

The first base-heuristic, inspired by the Clarke–Wright heuristic for the traveling salesman problem (TSP) [24], is an iterative, joint greedy bit allocation scheme which will be used throughout as our reference heuristic, cf. the illustration in Fig. 1(a) and Algorithm 1. It starts the search from $\mathbf{r} = \mathbf{0}$ and in each iteration calculates the extra cost $f(\tilde{\mathbf{r}})^{\mathbf{w}} - f(\mathbf{r})^{\mathbf{w}}$ for increasing one user's number of data-bits by one, i.e., $\tilde{r}_u = r_u + 1, \tilde{r}_i = r_i, \forall i \in \mathcal{U} \backslash \{u\}$, for all users $u \in \mathcal{U}$,

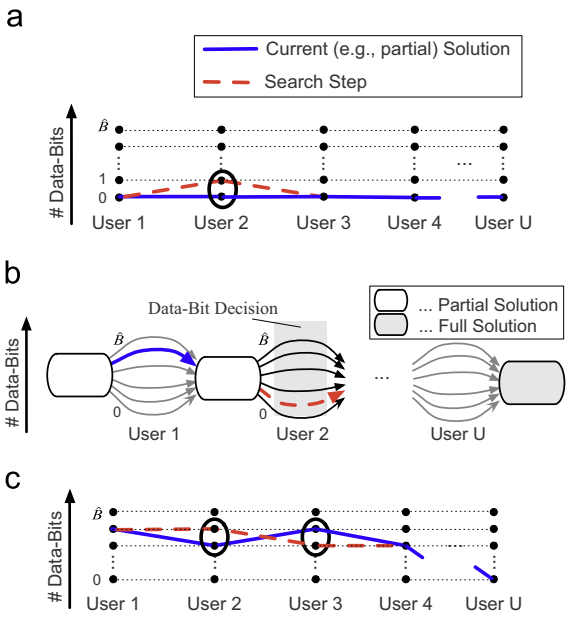

Fig. 1. Illustration of the three search principles underlying our base-heuristics; (a) constructive bit allocation (shown: based on a solution $\mathbf{r} = \mathbf{0}$, the number of data-bits of user 2 is increased by 1); (b) constructive sequential decision making (shown: the data-bits of user 2 are decided after the decision for user 1 has been made); (c) improving local search (shown: two data-bit decisions for users 2 and 3 are changed simultaneously).

cf. Lines 3–6. The allocation $\mathbf{r}$ of the previous iteration is updated by increasing the number of data-bits of the user which leads to the lowest extra cost, cf. Line 7. Ties are broken by picking the allocation with the lower sum-power. These steps are repeated until no update as defined above with negative extra cost exists.

Algorithm 1. Joint Greedy Optimization (JOGO), adapted from [10].

1: Initialize $\mathbf{r}$, $\delta^* = 0$, $f^{\text{prev}} = f^{\mathbf{w}}(\mathbf{r})$
2: **while** $\delta^* \leq 0$ **do**
3: **for** $u = 1, \ldots, U$ **do**
4: **if** $\exists \mathbf{p} \in \mathcal{Q} | r_u(\mathbf{p}) = r_u + 1, r_i(\mathbf{p}) = r_i,\ \forall i \in \mathcal{U} \backslash \{u\}$,
5: **then** $\delta_u = f^{\mathbf{w}}(\mathbf{p}) - f^{\text{prev}}$
6: **else** $\delta_u = \infty$
7: $u^* = \text{argmin}_{u=1\ldots U} \delta_u$, $\delta^* = \delta_{u^*}$
8: **if** $\delta^* \leq 0$ **then** $r_{u^*} = r_{u^*} + 1, f^{\text{prev}} = f^{\text{prev}} + \delta_{u^*}$

4.1.2. Sequential Greedy Optimization (SEGO)

The second, less complex base-heuristic is the Sequential Greedy Optimization (SEGO) scheme where users pick their data-bits r_u in sequence, cf. Fig. 1(b) and Algorithm 2. The user-sequence is determined by the ratio of the weights associated with the data-bits and transmit power respectively, cf. Line 2. Each user u then picks its data-bits r_u and joint power allocation $\mathbf{p}$ that minimizes the joint objective $f^{\mathbf{w}}(\mathbf{r})$, cf. Line 4. While JOGO has a complexity per bit-step of $O(U^3)$ due to the solution of a linear system of equations (assuming Gaussian elimination is used), the complete complexity of SEGO is only $O(U^2)$ as each user only evaluates up to $\hat{B}$ bit allocation possibilities. The main feature in such a sequential algorithm is the sequence in which the users commit to their data-bits. In Section 5.2.1 we will present an extension of SEGO where the user-sequence is randomized and selected together with the bit allocation.

Algorithm 2. Sequential Greedy Optimization (SEGO), adapted from [10].

1: Initialize $\mathbf{r}$
2: Determine sequence $\mathbf{s} \in \mathcal{R}^U$ by ordering users in descending order of $\check{w}_u / \hat{w}_u$
3: **for** $u = s_1, \ldots, s_U$ **do**
4: $[r_u, \mathbf{p}] = \text{argmin}_{[r_u \in \mathcal{B}, \mathbf{p} \in \mathcal{Q} | \mathbf{r}(\mathbf{p}) \geq \mathbf{r}]} \{f^{\mathbf{w}}(\mathbf{p})\}$

4.1.3. Local search (LS)

The third base-heuristic is a simple local search (LS), which aims to iteratively *improve* a given solution $\mathbf{r}$, cf. Fig. 1(c) and Algorithm 3. A specific search scheme is determined by the definition of a neighborhood set $\mathcal{N}(\mathbf{r}) \subseteq \mathcal{B}$ around $\mathbf{r}$ from which the solution evaluated next is taken, cf. [25]. A local optimum $\mathbf{r}^*$ can then be defined as an allocation with $f^{\mathbf{w}}(\tilde{\mathbf{r}}) \geq f^{\mathbf{w}}(\mathbf{r}^*)$, $\forall \tilde{\mathbf{r}} \in \mathcal{N}(\mathbf{r}^*)$, i.e., there is no allocation with strictly better objective value in the neighborhood of $\mathbf{r}^*$. Here we restrict ourselves to two possible neighborhood definitions [10]: (a) a one-step neighborhood $\mathcal{N}^{(1)}(\mathbf{r})$ where the data-bits r_u of exactly one user u are increased or decreased by one; and (b) a two-step neighborhood $\mathcal{N}^{(2)}(\mathbf{r}) \supseteq \mathcal{N}^{(1)}(\mathbf{r})$ where additionally two distinct users can independently increase or decrease their data-bits $r_u, r_{\tilde{u}}$, $u, \tilde{u} \in \mathcal{U}$, by one. Note that when we search through $\mathcal{N}^{(2)}(\mathbf{r})$ we start searching through $\mathcal{N}^{(1)}(\mathbf{r})$ first. This has an impact on the obtained solution depending on the search strategy, where we restrict ourselves to two possible strategies as described in Lines 4 and 5 of Algorithm 3, respectively. In the first option an allocation $\mathbf{r}$ is updated by the improving allocation $\tilde{\mathbf{r}} \in \mathcal{N}(\mathbf{r})$ which is encountered first, while in the second option the whole neighborhood $\mathcal{N}(\mathbf{r})$ is searched and the best allocation chosen for the update of $\mathbf{r}$. We will refer to these two LS strategies as "first-improving" and "best-improving" strategies [25, Chapter 8], respectively. In [10] the size $|\mathcal{N}(\mathbf{r})|$ of the neighborhood under the two described neighborhood definitions is shown to grow as $O(U)$. An empirical discussion about the impact of the number of users U and the initialization point $\mathbf{r}$ on the search complexity will be given in Section 6.

Algorithm 3. Local search (LS) based bit allocation.

1: Initialize $\mathbf{r}$
2: **repeat**
3: Update $\mathbf{r}$ by
4: (**a**) the first-found $\tilde{\mathbf{r}} \in \mathcal{N}(\mathbf{r})$ with $f^{\mathbf{w}}(\tilde{\mathbf{r}}) < f^{\mathbf{w}}(\mathbf{r})$, or
5: (**b**) any $\tilde{\mathbf{r}} \in \mathcal{N}(\mathbf{r})$ with $f^{\mathbf{w}}(\tilde{\mathbf{r}}) < f^{\mathbf{w}}(\mathbf{r})$, $f^{\mathbf{w}}(\tilde{\mathbf{r}}) \leq f^{\mathbf{w}}(\bar{\mathbf{r}})$, $\forall \bar{\mathbf{r}} \in \mathcal{N}(\mathbf{r})$
6: **until** Convergence

4.2. Analysis of base-heuristics

We analyze our base heuristics on a set of 84 6-user very high speed DSL (VDSL) scenarios with subscriber line lengths of 200 m, 400 m, 600 m, and 800 m, respectively, and a subset of subcarriers $\tilde{\mathcal{C}} \subset \mathcal{C}$, cf. Section 6 for details on the simulation setup. We compute the suboptimality of an allocation $\mathbf{r}$ compared to the optimal solution $\mathbf{r}^*$ as[1] $\text{suboptimality} = ((f^{\mathbf{w}}(\mathbf{r}) - f^{\mathbf{w}}(\mathbf{r}^*))/-f^{\mathbf{w}}(\mathbf{r}^*)) \cdot 100\ [\%]$. Note that all base-heuristics are guaranteed to give a solution with negative objective value if initialized by $\mathbf{r} = \mathbf{0}$, cf. the stopping criterion in Line 2 of Algorithm 1 and the initial objective value $f^{\mathbf{w}}(\mathbf{0}) = 0$. This leads to a guaranteed suboptimality between 0 and 100%. The average suboptimality of the base heuristic over subcarriers $\tilde{\mathcal{C}}$ and the 84 scenarios is 3.7%, with a maximum suboptimality of 39.3%. Table 1 shows the average suboptimality over chosen subcarriers $\tilde{\mathcal{C}}$ for scenarios that are worst in this respect. We find that the base-heuristic JOGO gives optimal results in all scenarios where the line-lengths are equal. As seen in Table 1 the highest suboptimality occurs in classical "near–far" type of scenarios, that is, scenarios where some line-lengths are substantially shorter than others.[2] This type of scenarios is widely accepted as the main application focus for power control in DSL.

As an example we regard the worst scenario (the first line in Table 1) and a specific subcarrier at approx. 9.26 MHz, i.e., one user is at 200 m, one at 400 m, and 4 lines at 800 m distance from the central telephone office, respectively. The constructive greedy algorithm JOGO only assigns data-bits to the nearest users, more precisely

[1] The case $\mathbf{r}^* = \mathbf{0}$ where the suboptimality is undefined was not encountered in our simulations and is therefore neglected.

[2] In this work we are not considering another network topology prone to the near–far problem, that is a mixed central office and cabinet DSL deployment.

Table 1
Worst-case scenarios for greedy Algorithm 1 (JOGO) out of any 6 user VDSL scenarios using given line-lengths.

# Users with line-lengths of …				Suboptimality of sum-objective over subcarriers $\check{\mathcal{C}}$ w/o (w/) local search (%)
200 m	400 m	600 m	800 m	
1	1	0	4	13.1 (1.0)
2	0	0	4	10.0 (2.1)
0	1	1	4	9.3 (1.9)
1	0	5	0	9.0 (8.4)
0	2	0	4	8.9 (1.1)

$\mathbf{r} = [9, 4, 0, 0, 0, 0]$. This has to be compared to the optimal allocation $\mathbf{r}^* = [6, 4, 3, 2, 2, 2]$, and implies a suboptimality of JOGO of more than 31%. In this example SEGO only assigns non-zero data-bits to the nearest user and incurs a suboptimality of over 15% when the user-sequence starts with this nearest user. Having the greedy base-heuristics followed by LS does not improve the suboptimality beyond 15%. For instance, initializing $r_3 = \cdots = r_6 = 0$ we need to assign r_2 (the second-nearest user) a value between 5 and 12 in order to be in the "basin of attraction" for local search to the optimum, independent of the (feasible) number of data-bits r_1 (assigned to the nearest user). Clearly, some non-greedy steps are necessary to bring us into this region, motivating the GRASP-like scheme [25, Chapter 8] in Section 5.1. Furthermore, we note that any user-sequence in SEGO which starts with any of the most distant users would have lead to an optimal result after subsequently applying LS in this example. This worst-case example (among the selected DSL scenarios) motivates a randomization of the user-sequence as applied in one of our heuristics, proposed in Section 5.2.1.

Basically one of the three approaches will be applied to overcome the shortcomings of the two purely greedy schemes, namely (a) the extension of JOGO through *purely* random decisions (a special case of GRASP [25, Chapter 8]), (b) the randomization of the decision-sequence in SEGO (using an ant-colony system [26]), and (c) randomized local search (e.g., iterated local search [25, Chapter 11]).

5. Advanced search algorithms

In the following we present detailed descriptions of heuristics implementing the extensions suggested in Section 4.2 for the power control problem in (2) which are partly inspired by well-known meta-heuristics, cf. the overview of all studied algorithms in Table 2. The presentation of the proposed algorithms follows the three search principles introduced in Section 4: GRASP will be introduced in Section 5.1 as an extension of the greedy base heuristic JOGO using randomization (cf. Fig. 1(a)). Rollout algorithms (RA), rSEGO, and ant colony system algorithms are deterministic and randomized sequential decision making algorithms (cf. Fig. 1(b)), and described in Section 5.2. Randomized local search, simulated annealing, as well as iterated local search can be classified as randomized variants of the local search scheme in Algorithm 3 (cf. Fig. 1(c)), and are presented in Section 5.3.

Table 2
Summary of algorithms applied to the problem in (2).

Algorithm	Abbr.	Reference
Deterministic		
Solver "Couenne"	COU	[11,12]
Branch-and-bound	OPT	[16, Algorithm 1]
Joint Greedy Optimization	JOGO	[10], Section 4.1
Sequential Greedy Optimization	SEGO	[10], Section 4.1
Local Search	LS	[10], Section 4.1
Rollout Algorithm	RA	[27], Section 5.2.2
Stochastic		
Greedy Rand. Adapt. Search Proc.	GRASP	[25, Chapter 8], Section 5.1
Iterated Local Search	ILS	[25, Chapter 11], Section 5.3.2
Simulated Annealing	SA	[25, Chapter 10], Section 5.3.1
Ant Colony System	ACS	[26], Section 5.2.3
Randomized SEGO	rSEGO	[7], Section 5.2.1
Randomized LS	rLS	Section 6.1

Note that for simplicity a parameter K will denote the number of iterations in all these algorithms, although the already mentioned stopping criterion based on the number of objective evaluations is used in all our simulations. Furthermore, the best found solution (the "incumbent") is initialized at $\mathbf{r}^* = \mathbf{0}$ or by the reference algorithm JOGO.

5.1. Greedy Randomized Adaptive Search Procedure (GRASP)

GRASP [25, Chapter 8] is a modification of greedy search through randomizing the data-bit decisions, cf. Algorithm 4. In Lines 4–12 a greedy bit allocation similarly as in JOGO is performed. However, differently to JOGO, in Line 10 a random non-greedy selection is made. An additional parameter β determines the trade-off between greedy ($\beta = 0$) and purely random search ($\beta = 1$). This parameter is randomly sampled in each iteration based on the average objective value g_i experienced so far under parameter i from the set $\{\beta_j\}, j \in \mathcal{M} = \{1, \ldots, M\}$, M meaning the number of values among which we select β, cf. Lines 3 and 15 of Algorithm 4, respectively. Note that a set of pre-selected parameter values makes the algorithm self-adaptive to the actual problem instance. For example, in a crosstalk-free scenario pure greedy decisions will experience the best average objective, and the strategy $\beta = 0$ will hence be selected more frequently. Differently, near–far scenarios require less greedy decisions, meaning that higher β values will be selected most of the time. The experienced objective values $g_i, i \in \mathcal{M}$, are initialized by the lowest (i.e., best) possible objective value $\check{f} = -\hat{B} \sum_{u \in \mathcal{U}} \check{w}_u$, which comes from the fact that the transmit power is lower-bounded by zero and the number of data-bits upper-bounded by $\hat{B}$, cf. the definition of the objective in (1). A single run of allocating data-bits to users (Lines 4–12) terminates when a purely greedy decision would increase the objective value, i.e., $\delta^{\min} > 0$, cf. Lines 4, 7, and 9. The free parameters to be chosen are the LS strategy, the definition of the neighborhood

set (cf. Line 13), and the randomization parameters β_i, $i \in \{1,\dots,M\}$.

Algorithm 4. GRASP, adapted from [7].

1: Initialize $\mathbf{r}=\mathbf{0}$, $\mathbf{r}^*$, $\delta^{\min}=0$, $f^{\text{prev}}=f^{\mathbf{w}}(\mathbf{r})$, $\boldsymbol{\beta}\in\mathcal{R}^M$, $\beta_i\in[0,1]$, $g_i=\check{f}$, $\forall i\in\mathcal{M}=\{1,\dots,M\}$, K
2: **for** $k=1,\dots,K$ **do**
3: Set $\beta=\beta_{i^*}$, where $i^*\in\{1,\dots,M\}$ is sampled from the distribution

$$P_i=\frac{\tilde{g}_i}{\sum_{j\in\{1,\dots,M\}}\tilde{g}_j},\quad \forall i\in\mathcal{M}, \tag{3}$$

$$\tilde{g}_i=\max_{i\in\{1,\dots,M\}}\{g_i\}-g_i,\quad \forall i\in\mathcal{M}, \tag{4}$$

4: **while** $\delta^{\min}\le 0$ **do**
5: **for** $u=1,\dots,U$ **do**
6: **if** $\exists\mathbf{p}\in\mathcal{Q}|r_u(\mathbf{p})=r_u+1, r_i(\mathbf{p})=r_i, i\in\mathcal{U}\backslash\{u\}$,
7: **then** $\delta_u=f^{(\mathbf{w})}(\mathbf{r})-f^{\text{prev}}$
8: **else** $\delta_u=\infty$
9: $\delta^{\min}=\min_{u\in\mathcal{U}}\{\delta_u\}$, $\delta^{\max}=\max_{u\in\mathcal{U}}\{\delta_u\}$
10: Uniformly sample a user u^* from the set $\{u\in\mathcal{U}|$
11: $\delta_u\le(1-\beta)\delta^{\min}+\beta\delta^{\max}\}$
12: **if** $\delta^{\min}\le 0$ **then** $r_{u^*}=r_{u^*}+1, f^{\text{prev}}=f^{\text{prev}}+\delta_{u^*}$
13: Local search starting at $\mathbf{r}$ and ending in $\tilde{\mathbf{r}}$, cf. Algorithm 3
14: Update incumbent $\mathbf{r}^*=\tilde{\mathbf{r}}$ if $f^{\mathbf{w}}(\tilde{\mathbf{r}})<f^{\mathbf{w}}(\mathbf{r}^*)$
15: Update the average g_{i^*} with the newest value f^{prev}

5.2. Advanced sequential decision making algorithms

In Section 4.2 we identified the potential of changing the sequence of users in making data-bit decisions, cf. Fig. 1(b) for an illustration of the general search principle. This idea will be turned into an algorithm in Section 5.2.1. Differently, although users make bit-decisions sequentially, they may still take the influence their decision has on subsequent users into account. This idea is followed in Section 5.2.2. Another approach applied in Section 5.2.3 is to repeatedly run through the sequential decision making process, thereby learning from beneficial data-bit decisions.

5.2.1. Randomized SEGO (rSEGO)

The heuristic rSEGO is a modification of SEGO in Section 4.1 through the randomization of user-sequence as well as data-bit decisions, cf. Algorithm 5. More precisely, the bit allocation is sequential as in SEGO (cf. Line 5), the user-sequence is adaptive and based on an ant colony system [26] (cf. Lines 9–15), and also the per-user bit allocation is randomized similarly as in GRASP above (cf. Line 7). Note that this approach differs from the version of ant colony systems in proposed Section 5.2.3, where data-bit decisions are learned directly.

Algorithm 5. Randomized SEGO (rSEGO), adapted from [7].

1: Initialize $\mathbf{r}^*$, K, $\overline{K}$, values $\tilde{\tau}_u(i)$, $\forall u,i\in\mathcal{U}$, q_0, β, ρ, user-order $\boldsymbol{s}^{(\overline{k})}\in\mathcal{R}^U$
2: **for** $k=1,\dots,K$ **do** {/*iterations*/}
3: **for** $\overline{k}=1,\dots,\overline{K}$ **do** {/*ant runs*/}
4: Set $\mathbf{r}^{(\overline{k})}=\mathbf{0}$, $\tilde{\mathcal{U}}=\mathcal{U}$, Uniformly sample $s_1^{(\overline{k})}\in\mathcal{U}$
5: **for** $i=1,\dots,U$ **do** {/*sequential decisions*/}
6: Set $u=s_i^{(\overline{k})}$ and compute

$$f^{\min}=\min_{\{r_u^{(\overline{k})}\in\mathcal{B}|\mathbf{p}(\mathbf{r}^{(\overline{k})})\in\mathcal{Q}\}}\{f^{\mathbf{w}}(\mathbf{r}^{(\overline{k})})\},$$

$$f^{\max}=\max_{\{r_u^{(\overline{k})}\in\mathcal{B}|\mathbf{p}(\mathbf{r}^{(\overline{k})})\in\mathcal{Q}\}}\{f^{\mathbf{w}}(\mathbf{r}^{(\overline{k})})\}$$

7: Uniformly sample $r_u^{(\overline{k})}$ from the set $\{r_u^{(\overline{k})}\in\mathcal{B}|\ \mathbf{p}(\mathbf{r}^{(\overline{k})})\in\mathcal{Q}, f^{\mathbf{w}}(\mathbf{r}^{(\overline{k})})\le(1-\beta)f^{\min}+\beta f^{\max}\}$
8: **if** $i<U$ **then**
9: Remove $u=s_i^{(\overline{k})}$ from the set of possibilities $\tilde{\mathcal{U}}$
10: Uniformly sample $q\in[0,1]$
11: **if** $q<q_0$ **then**
12: Set $s_{i+1}^{(\overline{k})}=\text{argmax}_{j\in\tilde{\mathcal{U}}}\{\tilde{\tau}_u(j)\}$
13: **else**
14: Sample $s_{i+1}^{(\overline{k})}=j\in\mathcal{U}$ from the distribution

$$P_u(j|\tilde{\mathcal{U}})=\begin{cases}\dfrac{\tilde{\tau}_u(j)}{\sum_{l\in\tilde{\mathcal{U}}}\tilde{\tau}_u(l)} & \text{if } j\in\tilde{\mathcal{U}},\\ \text{and } 0 & \text{otherwise,}\end{cases}$$

15: Update $\tilde{\tau}_u(s_{i+1}^{(\overline{k})})=(1-\rho)\cdot\tilde{\tau}_u(s_{i+1}^{(\overline{k})})$
16: Update $\mathbf{r}^{(\overline{k})}$ by a local optimum $\tilde{\mathbf{r}}$, through LS in Algorithm 3 starting at $\mathbf{r}^{(\overline{k})}$ using neighborhood $\mathcal{N}(\mathbf{r})$
17: $\overline{k}^*=\text{argmin}_{\overline{k}=1,\dots,\overline{K}}\{f^{\mathbf{w}}(\mathbf{r}^{(\overline{k})})\}$
18: Update the incumbent $\mathbf{r}^*=\mathbf{r}^{(\overline{k}^*)}$ if $f^{\mathbf{w}}(\mathbf{r}^{(\overline{k}^*)})<f^{\mathbf{w}}(\mathbf{r}^*)$
19: Update $\tilde{\tau}_{s_i^{(\overline{k}^*)}}(s_{i+1}^{(\overline{k}^*)})$, $i=1,\dots,U-1$, as $\tilde{\tau}_u(j)=(1-\rho)\cdot\tilde{\tau}_u(j)+\rho\cdot(\hat{f}-f^{\mathbf{w}}(\mathbf{r}^{(k^*)}))$.

In the following we describe the process of allocating data-bits more closely. We define a set $\tilde{\mathcal{U}}$ that includes the indices of those users that have not yet made a bit allocation. A user $i\in\tilde{\mathcal{U}}$ is assigned a corresponding value $\tilde{\tau}_u(i)$ that automatically decreases over time (cf. Line 15) and is updated over iterations (cf. Line 19) based on the best solutions found (cf. Line 17). More precisely, defining the highest (i.e., worst) objective value as $\hat{f}=\sum_{u\in\mathcal{U}}\hat{w}_u\hat{p}_u$ and a coefficient $\rho\in[0,1]$ for calculating an exponential moving average, the values $\tilde{\tau}_u(j)$ are updated in Line 19. The decision on the subsequent user is made deterministically (cf. Line 12) or by sampling according to these values, cf. Line 14, based on the threshold q_0. Differently to the original ACS the decision on the data-bits r_u is made greedily, cf. Line 7.

5.2.2. Rollout Algorithm (RA)

RAs [27] fall into the category of sequential decision making schemes, cf. Fig. 1(b) and Algorithm 6. For each user u we test any possible bit allocation $b\in\mathcal{B}$, set $r_u=b$, and run our base heuristic in Algorithm 1 to determine the allocations $r_i, i=u+1,\dots,U$, and the heuristic cost $\delta(b)$ of user u's allocation, cf. Line 5. User u then chooses the allocation r_u which minimizes $\delta(r_u)$. An attractive feature of RA is that there are no parameters to be tuned and the only choice is that of the base heuristic to be used. Furthermore, it necessarily dominates the base heuristic in performance as the base heuristic is run at iteration $u=1$ using all possible bit allocations $r_1\in\mathcal{B}$, hence also including the one the base heuristic would give as a result for user 1.

Algorithm 6. Rollout Algorithm (RA).

1: Initialize $\mathbf{r}=\mathbf{0}$, $\mathbf{r}^*$, decision value $\Delta(b)$, $\forall b\in\mathcal{B}$
2: **for** $u=1,\dots,U-1$ **do**
3: Set $\mathbf{r}^0=\mathbf{r}$
4: **for** all $b\in\mathcal{B}$ **do**

5: Obtain $\tilde{\mathbf{r}}$ and $\Delta(b)=f^{\mathbf{w}}(\tilde{\mathbf{r}})$ by Algorithm 1 with $U^{\text{start}}=u+1$, starting at $\mathbf{r}=\mathbf{r}^0$, where $r_u^0=b$
6: Update incumbent $\mathbf{r}^*=\tilde{\mathbf{r}}$ if $f^{\mathbf{w}}(\tilde{\mathbf{r}})<f^{\mathbf{w}}(\mathbf{r}^*)$
7: Set $r_u=\text{argmin}_{b\in\mathcal{B}}\{\Delta(b)\}$

5.2.3. Ant Colony System (ACS)

ACS is a nature-inspired meta-heuristic originally applied to solve the TSP [26]. We now describe a modified ACS algorithm for the discrete bit allocation problem in (2). A number of $\overline{K}$ tentative solutions are built sequentially by deciding how many data-bits to assign to users $u=1,\ldots,U$, cf. Fig. 1(b) and Lines 3–12 of Algorithm 7. These are referred to as "ant runs" in ACS terminology. The user-sequence in our simulations is generated by sorting users according to their distance (subscriber line length). A key distinction to other heuristics is the reinforcement of the decisions by assigning a quality-value $\tau_u(b)$ to each of the $|\mathcal{B}|\cdot U$ possible decisions, $b\in\mathcal{B}, u\in\mathcal{U}$. The sequential decisions are made by sampling according to a given probability distribution in Line 10 of Algorithm 7. These probabilities depend on the one hand on the decision values $\tau_u(b)$ for user u (see Line 11 for the corresponding update procedure), and on the other hand on the heuristic information $\eta_u(b,\mathbf{r})$ (defined in Line 10), which represents the modified cost of a bit allocation decision to user u. Note that the subset $\hat{\mathcal{B}}$ of available bit-allocation decisions in Line 10 can either be chosen according to the maximum number of bits as $\mathcal{B}$, or more restrictively from the subset $\{b|\mathbf{r}\in\times_{u\in\mathcal{U}}\mathcal{B}, r_u=b, \exists\mathbf{p}(\mathbf{r})\in\mathcal{Q}\}$. The latter set simply consists of all data-bit decisions b for user u that lead to feasible power allocations. However, as power increases with the number of data-bits, the calculation of this subset necessitates to search the largest feasible number of data-bits for user u, which is no extra work when the heuristic info in Line 10 is used. In our simulations in Section 6 we will compare the two definitions of $\hat{\mathcal{B}}$ to each other in terms of the resulting performance and complexity, with or without the usage of heuristic information $\eta_u(b,\mathbf{r})$.

Algorithm 7. Ant Colony System (ACS) based bit allocation.

1: Initialize $\mathbf{r}^*$, K, $\overline{K}$, values $\tau_u(b), \forall b\in\mathcal{B}, u\in\mathcal{U}$, q_0, ρ
2: **for** $k=1,\ldots,K$ **do** {/*iterations*/}
3: **for** $\overline{k}=1,\ldots,\overline{K}$ **do** {/*ant runs*/}
4: Set $\mathbf{r}^{(\overline{k})}=\mathbf{0}$
5: **for** $u=1,\ldots,U$ **do** {/*sequential decisions*/}
6: Uniformly sample $q\in[0,1]$
7: **if** $q<q_0$ **then**
8: Pick $r_u^{(\overline{k})}=b$ based on $b=\text{argmax}_{b\in\hat{\mathcal{B}}}\{\tau_u(b)\eta_u(b,\mathbf{r})\}$
9: **else**
10: Sample $r_u^{(\overline{k})}=b$ from $P_u(b|\mathbf{r}^{(\overline{k})})$ given as

$$P_u(b|\mathbf{r})=\begin{cases}\dfrac{\tau_u(b)\cdot\eta_u(b,\mathbf{r})}{\sum_{\tilde{b}\in\hat{\mathcal{B}}}\tau_u(\tilde{b})\cdot\eta_u(\tilde{b},\mathbf{r})} & \text{if } b\in\hat{\mathcal{B}},\\ 0 & \text{otherwise,}\end{cases}$$

where $\eta_u(b,\mathbf{r})=\hat{f}-f^{\mathbf{w}}(\tilde{\mathbf{r}})|_{\tilde{r}_i}=r_i, \forall i\in\mathcal{U}\backslash\{u\}, \tilde{r}_u=b$
11: Update $\tau_u(r_u^{(\overline{k})})$ as $\tau_u(b)=(1-\rho)\cdot\tau_u(b)$
12: *Optional:* If $\mathbf{p}(\mathbf{r}^{(\overline{k})})\in\mathcal{Q}$ then update $\mathbf{r}^{(\overline{k})}$ by a local optimum $\tilde{\mathbf{r}}$ starting Algorithm 3 at $\mathbf{r}^{(\overline{k})}$ using $\mathcal{N}(\mathbf{r})$
13: $\overline{k}^*=\text{argmin}_{\overline{k}=1,\ldots,\overline{K}}\{f^{\mathbf{w}}(\mathbf{r}^{(\overline{k})})\}$
14: Update the incumbent $\mathbf{r}^*=\mathbf{r}^{(\overline{k}^*)}$ if $f^{\mathbf{w}}(\mathbf{r}^{(\overline{k}^*)})<f^{\mathbf{w}}(\mathbf{r}^*)$
15: Update $\tau_u(b), \forall u\in\mathcal{U}$, as

$$\tau_u(b)=\begin{cases}(1-\rho)\cdot\tau_u(b)+\rho\cdot(\hat{f}-f^{\mathbf{w}}(\mathbf{r}^{\text{best}})) & \text{if } r_u^{\text{best}}=b,\\ \tau_u(b) & \text{otherwise,}\end{cases}$$

either using (**a**) $\mathbf{r}^{\text{best}}=\mathbf{r}^{(\overline{k}^*)}$, or (**b**) $\mathbf{r}^{\text{best}}=\mathbf{r}^*$

After a data-bit decision $r_u=b$ for user u, the value $\tau_u(b)$ is updated in Line 11. After all $\overline{K}$ runs of making the sequential data-bit decisions, a best allocation $\mathbf{r}^{\text{best}}$ is found either as the allocation giving the best objective value in the current iteration or in all iterations so far. Based thereupon, a global value update is performed in Line 15 in Algorithm 7, using a coefficient ρ in the exponential moving average calculation. The objective value of a solution constructed by the sequential decisions is not restricted in sign. Therefore we shift the objective values in Lines 10 and 15 by $\hat{f}=\sum_{u\in\mathcal{U}}\hat{w}_u\hat{p}^u$ in order to be able to use them as a scaled probability mass function in Line 10. Furthermore, we propose to initialize the decision values by large values $\tau_u(b)=\hat{f}-\check{f}$, which gives meaningful values also for the special case where $\hat{w}_u=0, \forall u\in\mathcal{U}$ (the pure maximization of data-bits), and where $\check{f}$ is defined in Section 5.1. In order to obtain a trade-off between exploration and exploitation a random value $q\in[0,1]$ is sampled in Line 6 of Algorithm 7, where the decision b is based on a greedy decision rule in Line 8 when $q<q_0$, and based on the probabilistic sampling in Line 10 otherwise. Note that the value $q_0=1$ selected for performance reasons in Section 6.2 means that only greedy decisions are taken.

In the definition of Algorithm 7 we have mainly followed the meta-heuristic description as outlined in [26]. Problem-specific adaptions were made for determining the feasible neighborhood $\hat{\mathcal{B}}$, for defining the decision value updates based on a shifted objective function, and for specific parameter settings. The free parameters to be defined are the decay coefficient ρ in the local and global pheromone updates, the threshold q_0 for determining the decision strategy, the number of virtual ants $\overline{K}$, the local search parameters (neighborhood and local search strategy) if Line 12 in Algorithm 7 is used, the update method in Line 15, and the set $\hat{\mathcal{B}}$ in Line 10, cf. the discussion above. Note that the local value updates diversify the solutions produced by the virtual ants [26], cf. the "temperature value" described in the next section.

5.3. Advanced local search schemes

The local search scheme in Section 4.1.3 may be trapped in a local optimum depending for instance on the initialization, cf. the search principle in Fig. 1(c). One such example has already been given in Section 4.2 where LS is initialized by a greedy base heuristic and eventually converges to a local optimum only. Similar to the principle of allowing non-greedy steps used in Section 5.1, in Section 5.3.1 we will allow LS to take non-improving steps in a simulated annealing framework. Differently, the idea of repeatedly restarting the LS at different starting points is followed in Section 5.3.2. As a simple base-line for our stochastic heuristics we consider a randomized local search (rLS) scheme where the LS algorithm is reinitialized at random starting points $\mathbf{r}$ uniformly drawn from the set $\times_{u\in\mathcal{U}}\mathcal{B}$.

5.3.1. Simulated annealing (SA)

SA [28, 25, Chapter 10] is a local search method in the spirit of Fig. 1(c) with the potential of escaping local optima by probabilistically allowing for "uphill" moves, cf. Algorithm 8 for a basic description. The neighborhood $\mathcal{N}(\mathbf{r})$ at an allocation $\mathbf{r}$ is sampled uniformly in Line 4. The probability of accepting the sampled allocation and moving from $\mathbf{r}$ to $\tilde{\mathbf{r}}$ is given in Line 5. Note that the neighborhood definitions in Section 4.1 ensure that $\tilde{\mathbf{r}} \in \times_{u \in \mathcal{U}} \mathcal{B}$ whenever $\mathbf{r} \in \times_{u \in \mathcal{U}} \mathcal{B}$, but $\mathbf{p}(\tilde{\mathbf{r}})$ might not be in the feasible set $\mathcal{Q}$. We therefore use the common understanding that $f^{\mathbf{w}}(\tilde{\mathbf{r}}) = \infty$ if $\tilde{\mathbf{r}} \notin \mathcal{Q}$. Parameter T is the "temperature" which regulates the probability of uphill moves. In the basic Algorithm 8 we use a simple geometric update schedule for T, cf. Line 7. The free parameters are the definition of the neighborhood $\mathcal{N}(\mathbf{r})$, the initial temperature T, and the parameter γ for updating the temperature in Line 7 of Algorithm 8.

Algorithm 8. Simulated annealing (SA) based bit allocation.

1: $\mathbf{r} = \mathbf{0}$, *Optional:* Use Algorithm 1to initialize $\mathbf{r}$
2: Initialize, $\mathbf{r}^*$, T, γ, K
3: **for** $k = 1, \ldots, K$ **do**
4: Uniformly sample $\tilde{\mathbf{r}}$ from the neighborhood $\mathcal{N}(\mathbf{r})$
5: Assign $\mathbf{r} = \tilde{\mathbf{r}}$ with probability $P_{\mathbf{r}}(\tilde{\mathbf{r}})$ defined as

$$P_{\mathbf{r}}(\tilde{\mathbf{r}}) = \begin{cases} \exp\left(-\frac{1}{T}\left(f^{\mathbf{w}}(\tilde{\mathbf{r}}) - f^{\mathbf{w}}(\mathbf{r})\right)\right) & \text{if } f^{\mathbf{w}}(\tilde{\mathbf{r}}) - f^{\mathbf{w}}(\mathbf{r}) > 0, \\ 1 & \text{otherwise.} \end{cases}$$

6: Update the incumbent $\mathbf{r}^* = \mathbf{r}$ if $f^{\mathbf{w}}(\mathbf{r}) < f^{\mathbf{w}}(\mathbf{r}^*)$
7: Update the temperature $T = T \cdot \gamma$

5.3.2. Iterated local search (ILS)

ILS [25, Chapter 11] is a scheme which iteratively runs local searches initialized at different starting points and thereby performs a random walk in the space of local optimal solutions, cf. Algorithm 9. More precisely, the basic algorithm repeatedly performs a local search and accepts the new local optimum based on a criterion identical to that in SA, cf. Lines 4 and 5 in Algorithm 9, respectively. The starting point for the next local search is determined by a perturbation of the current local optimum, cf. Line 7. This perturbation is one of the key points of the algorithm and performed by uniformly sampling one of the two alternative sets: the set $\overline{\mathcal{N}}^{(2)}(\mathbf{r})$ or the set $\overline{\mathcal{N}}^{(U\text{steps})}(\mathbf{r})$ where the elements of $\mathbf{r}$ are reduced by up to U data-bits, cf. Line 7. This perturbation set has the advantage that the perturbed allocation produced by sampling $\overline{\mathcal{N}}^{(U\text{steps})}(\mathbf{r})$ remains feasible if $\mathbf{r}$ was feasible.

Algorithm 9. Iterated local search (ILS) based bit allocation.

1: $\mathbf{r}^{(1)} = \mathbf{0}$, *Optional:* Use Algorithm 1 to initialize $\mathbf{r}^{(1)}$
2: Initialize $\mathbf{r} = \mathbf{r}^{(1)}$, $\mathbf{r}^*$, T, γ, K
3: **for** $k = 1, \ldots, K$ **do**
4: Search local optimum $\tilde{\mathbf{r}}$ starting at $\mathbf{r}$ as in Algorithm 3
5: Acceptance: Assign $\mathbf{r}^{(k+1)} = \tilde{\mathbf{r}}$ with probability $P_{\mathbf{r}^{(k)}}(\tilde{\mathbf{r}})$ defined in Line 5 of Algorithm 8, otherwise assign $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)}$
6: Update incumbent $\mathbf{r}^* = \mathbf{r}^{(k+1)}$ if $f^{\mathbf{w}}(\mathbf{r}^{(k+1)}) < f^{\mathbf{w}}(\mathbf{r}^*)$
7: Perturbation: Obtain $\mathbf{r}$ by uniformly sampling

$$\overline{\mathcal{N}}^{(2)}(\mathbf{r}) = \left\{\tilde{\mathbf{r}} \in \prod_{u \in \mathcal{U}} \mathcal{B} | \tilde{r}_u = r_u \pm 1, \tilde{r}_{\overline{u}} = r_{\overline{u}} \pm 1, \tilde{r}_i = r_i, \forall i \in \mathcal{U} \setminus \{u, \overline{u}\}, u \neq \overline{u}, u, \overline{u} \in \mathcal{U}\right\},$$

or

$$\overline{\mathcal{N}}^{(U\text{steps})}(\mathbf{r}) = \left\{\tilde{\mathbf{r}} \in \mathcal{B} | \tilde{r}_u = r_u - k_u, k_u \geq 0, \forall u \in \mathcal{U}, \sum_{u \in \mathcal{U}} k_u = \min\left\{U, \sum_{u \in \mathcal{U}} r_u\right\}\right\},$$

8: Update the temperature $T = T \cdot \gamma$

The free parameters in the presented basic implementation of ILS are the initialization method in Line 1 of Algorithm 9, the neighborhood $\mathcal{N}(\mathbf{r})$ used for local search, the perturbation set ($\overline{\mathcal{N}}^{(2)}(\mathbf{r})$ or $\overline{\mathcal{N}}^{(U\text{steps})}(\mathbf{r})$), the local search strategy, the initial temperature T, and the parameter γ for updating the temperature. We refer to [25, Chapter 11] for suggestions of possible modifications of the described ILS scheme.

6. Simulations and discussion

In Section 6.1 we describe the methodology and simulation parameters for the following selection of algorithmic parameters in Section 6.2. There we also compare the exact suboptimality of all heuristics on medium-scale problems, that is, with 6 DSL users only. In Section 6.4 we will then investigate the average performance of all heuristics in more realistic scenarios with 30 users, before discussing the obtained results in Section 6.5.

6.1. Simulation setup and methodology

The parameters for our xDSL simulator available in [22] were selected according to the ETSI very high speed DSL (VDSL) standard [29], with upstream band plan 997-M1x-M (containing 1635 subcarriers), background noise comprising ETSI VDSL noise A added to a flat background noise at -140 dBm/Hz, and hypothetical parameters as an SNR-gap [18] of 12.8 dB and $\hat{B} = 16$ based on our later comparison to the optimal branch-and-bound results in [16]. For comparing our single-carrier heuristics we choose a subset of subcarriers $\tilde{\mathcal{C}} = 1, 51, \ldots, 1601$, and construct our network scenarios using a set of specified subscriber line lengths $\mathcal{L} = \{200, 400, 600, 800\}$ m, considering all $U-$combinations with repetitions. In order to still be able to compare to optimal schemes as in [16], for the parameter setting and initial simulations in Sections 6.2 and 6.3 we restrict ourselves to $U = 6$ users, which results in $(|\mathcal{L}| + U - 1)$ over U generated network scenarios. Note that this allows us to identify scenarios where the given algorithms perform badly. Such scenarios were used to initially set the algorithmic parameters, cf. Section 6.2. Based on these settings various parameter changes were selected and the impact on the average performance studied by Monte-Carlo simulation. For analyzing randomized algorithms we make 100 repetitions and present mean results as well as 95% confidence intervals according to a t-test. As a benchmark for all our algorithms we use "Couenne" [11,12], a free branch-and-bound based solver for nonconvex mixed-integer problems.[3] In order to study networks with a

[3] We used release 0.2.2 and default parameters except the time limit as specified in the simulation results, the priority level of continuous

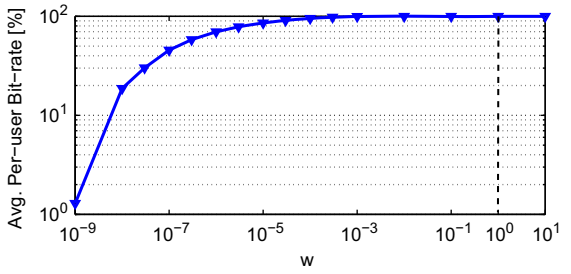

Fig. 2. Average data-bits for various 6 user VDSL scenarios under identical weights $\check{w}_u = w, \forall u \in \mathcal{U}$.

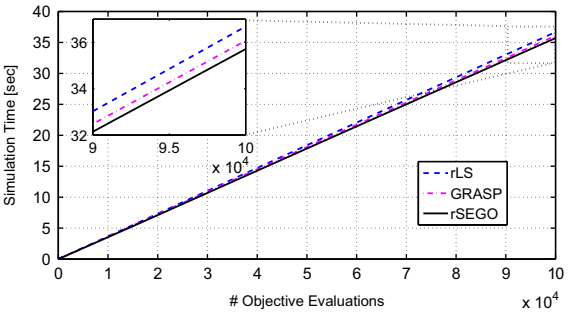

Fig. 3. Comparison of complexity metrics.

larger number of users, in Section 6.4 we consider 30-user scenarios. The number of combinations of assigning the four line-lengths to each user is in the thousands. Therefore we randomly assign the four line-lengths to the 30 users. While for 6 users we are able to compute the optimum of the problem in (2), for 30 users we state the performance improvements compared to the base heuristic JOGO. As in [16] we use equal weights for all users, set to $\hat{w}_u = 1/U, \check{w}_u = 1, \forall u \in \mathcal{U}$, which leads to a maximum sum of data-bits in the 6-user scenarios, cf. Fig. 2.

We make the practical assumption that there is a restriction in simulation time for solving the subproblems in (2). Furthermore, in order to make our results reproducible for future research we use the number of power evaluations $\mathbf{p}(\mathbf{r})$ by solving a linear matrix equation [15] as the stopping criterion for all considered stochastic search heuristics. In Fig. 3 we relate the number of evaluations to the simulation times of a selection of our algorithm implementations[4] for an example with $U = 60$ users, uniformly distributed over the four subscriber line lengths, and regarding the lowest subcarrier at approx. 3 MHz. The metric can be seen to preserve the comparability among different heuristics. For the above described parameter setting we assume a limit of 10^3 power evaluations for all stochastic algorithms. This low value is justified by the large number of subproblems that has to be solved in practice (e.g., in the order of 10^5 in decomposition-based large-scale multi-carrier power control [10]) and the sufficient performance certain algorithms already show for small complexity budgets in networks with few users.

We initialize the incumbent solution (but not the initial starting-point $\mathbf{r} = \mathbf{0}$) of all methods by the result of the base heuristic JOGO in Algorithm 1. As this heuristic is guaranteed to give a solution with negative objective value, we have that also all other heuristics will produce a solution $\mathbf{r}^*$ with negative objective value $f^{\mathbf{w}}(\mathbf{r}^*)$ and we can compute the suboptimality of all algorithms as described in Section 4.2.

Table 3
Parameter settings.

Algorithm	Selected parameter settings
Deterministic	
JOGO	With/without LS, $\mathcal{N}(\mathbf{r}) = \mathcal{N}^{(2)}(\mathbf{r})$, first-improving strategy
SEGO	With/without LS, $\mathcal{N}(\mathbf{r}) = \mathcal{N}^{(2)}(\mathbf{r})$, first-improving strategy
LS	$\mathcal{N}(\mathbf{r}) = \mathcal{N}^{(2)}(\mathbf{r})$, first-/best-improving strategy
RA	With/without LS, $\mathcal{N}(\mathbf{r}) = \mathcal{N}^{(2)}(\mathbf{r})$, first-improving strategy
Stochastic	
rLS	$\mathcal{N}(\mathbf{r}) = \mathcal{N}^{(2)}(\mathbf{r})$, first-improving strategy
rSEGO	$\beta = 0.75$, uses local search with $\mathcal{N}(\mathbf{r}) = \mathcal{N}^{(2)}(\mathbf{r})$ and first-improving strategy, $\overline{K} = 5$, $\rho = 0.99$, $q_0 = 0.2$, iteration-best global value update
GRASP	$\boldsymbol{\beta} = [0.75, 1]$, $\mathcal{N}(\mathbf{r}) = \mathcal{N}^{(2)}(\mathbf{r})$, first-improving strategy
ILS	Initialization of $\mathbf{r}^{(1)}$ using Algorithm 1, $\mathcal{N}(\mathbf{r}) = \mathcal{N}^{(2)}(\mathbf{r})$, first- improving strategy, $T = 10$, $\gamma = 0.99$, perturbation set $\overline{\mathcal{N}}^{(2)}(\mathbf{r})$
SA	$\mathcal{N}(\mathbf{r}) = \mathcal{N}^{(2)}(\mathbf{r})$, $T = 10$, $\gamma = 0.99$
ACS	$\rho = 0.99$, $\hat{\mathcal{B}} = \mathcal{B}$, $q_0 = 1$, $\overline{K} = 20$, no heuristic info, using local search with $\mathcal{N}(\mathbf{r}) = \mathcal{N}^{(2)}(\mathbf{r})$ and first-improving strategy, iteration-best global value update

6.2. Parameter selection

In the following we describe the chosen parameters, some insights, and the performance results for the 6-user VFSL scenarios, cf. Tables 3 and 4, respectively. The values reported in Table 4 marked by a footnote 'a' reflect the applied stopping criterion (complexity budget) and exclude the incumbent initialization using Algorithm 1. As previously mentioned, the choice of parameters is based on the Monte-Carlo simulation in 6-user networks where optimal solutions are computable.

Comparing the two greedy randomized schemes rSEGO and GRASP, both, their performance metrics as well as the scenarios where their performance is suboptimal are found to be similar. GRASP is based on JOGO and we found that it takes a parameter $\beta = 1$ to remedy the shortcomings of JOGO in the example made in Section 4.2.[5] Hence, we used parameters $\boldsymbol{\beta} \in \mathcal{R}^2, \beta_2 = 1$ as we found that selecting a

(*footnote continued*)
variables (1001), thereby enforcing branching on the integer data-bit variables, the number of convexification points (10), and the artificial cutoff parameter (0).

[4] The algorithms were coded in Matlab and run on a 2.4 GHz quad-core Windows XP system with 3.5 GB RAM.

[5] The users' subscriber lines s are certainly not identical in their transmission parameters in practice. Still, the example suggests β to be set to a large value in near–far scenarios in order to overcome the shortcomings of a greedy search.

Table 4
Simulation results on suboptimality and complexity of various heuristics on the 84 6-user VDSL scenarios with a subset of 33 subcarriers.

Algorithm	Mean subopt. [%] (per subcarr.)	Max. subopt. [%] of sum-obj. over subcarr. $\tilde{c}$	Mean op.-count [$\times 10^3$] or sim.-time per subcarr.
Deterministic			
OPT	0.0	0.0	73.28
COU	43.216	87.751	1 s
	3.243	7.286	10 s
	2.437	6.124	20 s
JOGO + COU	2.819	8.661	1 s
	0.800	2.294	10 s
	0.650	1.857	20 s
JOGO	3.703	13.099	0.15
JOGO + LS	1.322	8.407	0.22
SEGO	32.450	(51.003)	0.03
	(53.630)	(58.884)	(0.02)
SEGO + LS	17.728	27.758	0.14
	(2.422)	(27.758)	(0.28)
LS^{first}	3.043	9.044	0.27
LS^{best}	1.371	8.407	1.16
RA	0.530	2.874	2.17
RA + LS	0.058	0.500	3.33
Stochastic			
rLS	1.364 ± 0.012	7.770	1[a]
	0.044 ± 0.001	0.820	10[a]
rSEGO	0.035 ± 0.001	0.779	1[a]
	0.0006 ± 0.0003	0.260	10[a]
GRASP	0.038 ± 0.001	0.649	1[a]
	0.015 ± 0.0003	0.384	10[a]
ILS	0.547 ± 0.006	6.182	1[a]
	0.149 ± 0.002	0.920	10[a]
SA	0.161 ± 0.003	1.298	1[a]
	0.057 ± 0.001	0.780	10[a]
ACS	$0.001 \pm 5 \times 10^{-19}$	0.124	1[a]
	0 ± 0	0	10[a]

[a] The numbers reflect the set complexity budget per subcarrier problem.

second randomization parameter besides $\beta_2 = 1$ reduces suboptimality. However, the performance is not sensitive to the exact choice of β_1 in our selected scenarios, cf. also the parameter selection in Line 3 of Algorithm 4.

6.3. Medium-scale problem results

6.3.1. Greedy deterministic algorithms

We start the discussion of the results on medium-scale problems with the deterministic algorithms in Table 4. Regarding the results for the solver Couenne we see that even with a larger complexity budget of 10 s it is not possible to come close to the performance of any shown heuristic. This is another motivation for using heuristics. Fig. 3 where our complexity metric is related to the simulation time of our algorithms shows that this complexity comparison is in fact favorable for Couenne. The complexity of the greedy schemes JOGO, SEGO and RA depends on the number of data-bits that are feasible, and therefore on the scenario and subcarrier (frequency bin). The values for SEGO in brackets are the result of starting the greedy scheme in the reverse order of users' line lengths. For the TSP, greedy heuristics have shown to provide good starting points for local search schemes, where a better performing greedy search gives better starting points [30]. A comparison of the results for JOGO and SEGO confirms this intuition. The rollout algorithm (RA) has on average a higher complexity metric than the lower limit (10^3 power evaluations) set for the randomized algorithms. However, combining RA with local search we find lower average suboptimality and (maximal) complexity compared to SA with 10^4 power evaluations.

6.3.2. Local search strategies

Next we discuss the selected local search strategies. While $|\mathcal{N}^{(2)}(\mathbf{r})| \geq |\mathcal{N}^{(1)}(\mathbf{r})|$ means that a two-step neighborhood potentially necessitates more objective evaluations, the performance of algorithms using $\mathcal{N}(\mathbf{r}) = \mathcal{N}^{(2)}(\mathbf{r})$ outperformed that using $\mathcal{N}(\mathbf{r}) = \mathcal{N}^{(1)}(\mathbf{r})$, cf. the selected

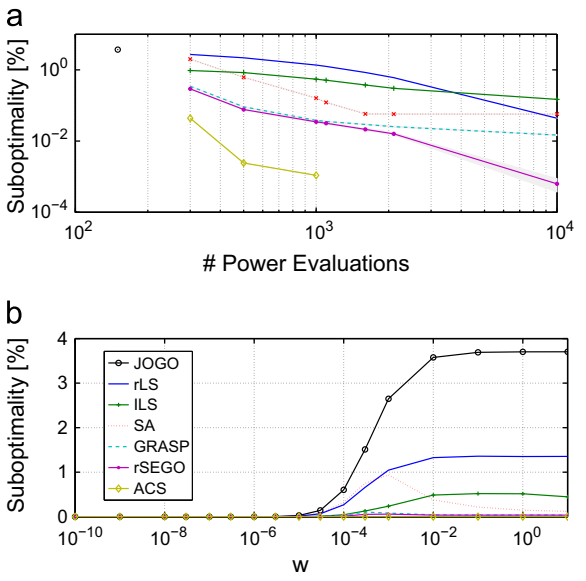

Fig. 4. Average suboptimality of various heuristics in 6-user VDSL scenarios; (a) dependency on the complexity budget; (b) dependency on the weight on data-bits $\check{w}_u = w, \forall u \in \mathcal{U}$.

parameters in Table 3 and the neighborhood definitions described in Section 4.1.3. Furthermore, the suboptimality in Table 4 of most stochastic algorithms is satisfying, which justifies setting the complexity budget to a fairly low value (10^3 power evaluations, i.e., less than 2% of the fastest optimal scheme's average complexity in [10]). Another selected parameter common to all local search schemes is the local search strategy, with the exception being the pure LS algorithm where we have tried both discussed strategies. We argue that a first-improving strategy saves operations compared to a best-improving (i.e., full neighborhood) search, which is especially critical when a tight complexity budget is used. This can also be seen from the results of "LS$^{\text{first}}$" using the first-improving strategy compared to that of "LS$^{\text{best}}$" using the best-improving strategy. Note however that the speed of a local search depends on its initialization and is typically faster when initialized at a solution different from $\mathbf{r} = \mathbf{0}$ as done for pure LS. With the exception of such improved initialization we have not explored techniques for reducing the complexity of local search, cf. [30] for references on such techniques proposed for the TSP.

6.3.3. Stochastic algorithms

A parameter setting with a similar impact as the local search strategy is the choice $\hat{\mathcal{B}} = \mathcal{B}$ for ACS, cf. its usage in Line 10 of Algorithms 7. Not restricting the bit allocation possibilities in a virtual ant's sampling process to values in $\{\mathbf{r} \in \times_{u \in \mathcal{U}} \mathcal{B} | p(\mathbf{r}) \in \mathcal{Q}\}$ opens the chance that the generated starting point is infeasible, but allows us to save objective evaluations and therefore to run more local searches from different starting points. For given scenarios and parameter settings ACS is the best performing algorithm out of the compared schemes, with optimal results at 10^4 power evaluations and suboptimal results only for a single subcarrier/scenario at 10^3 power evaluations.

Next we focus on the performance results of randomized heuristics in Table 4. The randomized base-line algorithm rLS gives results similar to JOGO with subsequent LS but at higher complexity, which shows that the naive rLS scheme is not a good choice for the constrained combinatorial problem in (2). Fig. 4(a) depicts the average suboptimality of all randomized heuristics as a function of the complexity budget in various 6-user VDSL network scenarios. Intuitively, allowing the algorithms to test more solutions leads on average to a better performance. The performance curve for SA in Fig. 4(a) flattens out at a higher number of power evaluations, indicating an insufficient mechanism for escaping local optima and/or a too small neighborhood size. ACS performs best in these test scenarios, where its curve stops at 10^3 as it is optimal on the simulated points beyond that. Note that rLS eventually performs better than ILS and SA, which hints at insufficient diversification capabilities of these two schemes. Fig. 4(b) similarly shows, for a fixed complexity budget of 10^3 evaluations, the dependency of the heuristics' average suboptimality on the weight on data-bits $\check{w}_u = \check{w}, \forall u \in \mathcal{U}$, and hence implicitly on the user's targeted transmission rate, as the average number of data-bits per user increases with these weights, cf. Fig. 2. JOGO, the solution of which is used as an initial incumbent for all schemes, was found to have a monotonously increasing suboptimality with $\check{w}$. Also, the optimal number of data-bits does not change in most scenarios above $\check{w} = 10^{-2}$. Differently to JOGO, all heuristics show a peak suboptimality for a medium weight value, however, at different values for different heuristics. Intuitively this can be explained by the fact that with increasing $\check{w}$ what matters most is the total number of data-bits achieved by all users. Then it matters less *how* the data-bits are distributed among the users as this distribution only influences the transmit power which has a low weight in the objective for high values of $\check{w}$. A similar behavior has been observed for branch-and-bound schemes in [16].

6.4. Large-scale problem results

Due to the large number of 30-user scenarios under combinations of the four selected subscriber line lengths we uniformly sample 200 and 100 scenarios (not cable lengths!) for the deterministic and randomized algorithms, respectively, make 50 repetitions for randomized algorithms, and present mean values with 95% confidence intervals according to a *t*-test. In order to reduce the variance of our results we run all heuristics on identical, sampled scenarios. The parameterization of all algorithms is the same as in the previous section. Furthermore, as we lack an optimal solution we use the greedy algorithm JOGO as our reference and show the improvements in objective value. The maximum complexity limit is now 2×10^4 objective evaluations. Table 5 reports our average simulation results, where the values marked by a footnote 'a' reflect the applied stopping criterion (complexity budget) and exclude the incumbent initialization using JOGO which would add 150 operations.

Table 5
Simulation results comparing various heuristics to the greedy heuristic JOGO on various 30-user VDSL scenarios with a subset of 33 subcarriers.

Algorithm	Mean obj. improvement [%] (per subcarr.)	[Min.,Max.] improv. [%] of sum-obj. over subcarr. $\check{c}$	Mean op.-count [$\times 10^3$] per subcarr.
Deterministic			
JOGO	0	[0, 0]	1.79 ± 0.06
JOGO + LS	8.029 ± 0.781	[0.909, 32.129]	3.92 ± 0.07
SEGO	-63.996 ± 1.664	[−76.569, −14.716]	0.08 ± 0.00
SEGO + LS	-64.755 ± 0.928	[−74.966, −39.134]	2.34 ± 0.05
LS^{first}	0.357 ± 0.193	[0, 12.339]	4.06 ± 0.10
LS^{best}	8.022 ± 0.778	[0.811, 31.480]	40.70 ± 0.91
RA	6.494 ± 1.181	[0, 36.078]	121.77 ± 5.60
RA + LS	8.728 ± 0.837	[1.358, 32.971]	248.04 ± 6.04
Stochastic			
rLS	0 ± 0	[0, 0]	10[a]
	0 ± 0	[0, 0]	20[a]
rSEGO	8.897 ± 0.004	[0, 29.024]	10[a]
	9.756 ± 0.002	[1.360, 21.886]	20[a]
GRASP	9.047 ± 0.004	[0, 29.024]	10[a]
	9.860 ± 0.002	[1.484, 21.886]	20[a]
ILS	8.661 ± 0.011	[0, 28.469]	10[a]
	9.152 ± 0.007	[1.137, 21.886]	20[a]
SA	5.536 ± 0.023	[0, 29.024]	10[a]
	5.742 ± 0.025	[0, 20.029]	20[a]
ACS	6.965 ± 0.000	[0, 20.618]	10[a]
	7.244 ± 0.000	[1.618, 13.070]	20[a]

[a] The numbers reflect the set complexity budget per subcarrier problem.

The randomized heuristics GRASP, rSEGO and ILS perform now best, with an improvement upon the objective values achieved by the greedy base heuristic by on average 9.9%, 9.8%, and 9.2%, respectively, under the limit of 2×10^3 objective evaluations. Note however that the simple *deterministic* extension of the greedy constructive heuristic JOGO by a two-step local search (with a negligible difference whether we initialized LS at the solution of JOGO or at $\mathbf{r} = \mathbf{0}$, using JOGO solely to provide an incumbent solution) improves the greedy heuristic already by on average 8% while taking on average only 3.92×10^3 power evaluations. Notably, the maximum improvement in sum-objective over all 33 tested subcarriers encountered in any tested network scenario is as high as 32%.

6.5. Discussion

Considering the parameter settings in Table 3 and the results in Table 4 it becomes evident that those schemes work well which use some kind of a diversification method (a "guided" randomization) to obtain different starting points for a local search. These schemes can be summarized as "multi-point" or "multi-start methods" [25, Chapter 12] and categorized from this point of view: All presented multi-start algorithms have in common that they use a "guided" randomization for generating different starting points. However, while in ILS and SA solutions are improved iteratively, rSEGO, GRASP and ACS build the initial starting points for local search from scratch in a constructive way. Also, the results under ILS and SA are worse in comparison to the other stochastic methods (except rLS, which is inferior due to its evaluation of infeasible solutions), which hints that the diversification achieved in these schemes is insufficient. We note however that ILS offers a possibility to improve diversification by altering the perturbation method in Line 7 of Algorithm 9. Another criterion is the memory used in multi-start schemes [25, Chapter 12]. GRASP on the one hand is memory-less (with the exception of the choice of random parameters). On the other hand, rSEGO and ACS use memory in the form of value tables which allow us to reinforce decisions which repeatedly lead to good performance. Note that one possible way of extending our algorithms is by allowing moves through infeasible regions of the search space, e.g., by utilizing penalty functions.

7. Conclusions

We study the application of ten selected heuristics to a non-convex integer power control problem in digital subscriber lines (DSL). In various 6-user DSL scenarios provably near-optimal results are obtained by several of the proposed randomized heuristics. In 30-user scenarios optimal results are intractable and a deterministic greedy constructive heuristic "JOGO" is hence chosen as a reference. Extending this reference algorithm by a proposed local search scheme gives already average improvements of over 7% at low complexity. Randomized heuristics still show slight improvements beyond that for moderate complexity limits. Quantitatively, the proposed heuristics have shown an average gain in objective value compared to the greedy constructive heuristic of up to 10% for the larger 30-user scenarios.

Acknowledgments

The Competence Center FTW Forschungszentrum Telekommunikation Wien GmbH is funded within the program COMET - Competence Centers for Excellent Technologies by BMVIT, BMWA, and the City of Vienna. The COMET program is managed by the FFG.

References

[1] Point Topic, World broadband statistics: Short report Q3 2010, Technical Report, Point Topic Ltd, 2012.
[2] M. Guenach, C. Nuzman, K. Hooghe, J. Maes, M. Peeters, Reduced dimensional power optimization using class AB and G line drivers in DSL, in: International Workshop on Green Communications, IEEE GLOBECOM, Miami, USA, 2010, pp. 1443–1447.
[3] C. Wong, R. Cheng, K. Letaief, R. Murch, Multiuser OFDM with adaptive subcarrier, bit, and power allocation, IEEE J. Sel. Areas Commun. 17 (10) (1999) 1747–1758.
[4] K. Leung, L.-C. Wang, Integrated link adaptation and power control to improve error and throughput performance in broadband wireless packet networks, IEEE Trans. Wireless Commun. 1 (4) (2002) 619–629.
[5] M. Chatterjee, H. Lin, S. Das, Rate allocation and admission control for differentiated services in CDMA data networks, IEEE Trans. Mobile Comput. 6 (2) (2007) 179–191.
[6] M. Pischella, J.-C. Belfiore, Weighted Sm throughput maximization in multicell OFDMA networks, IEEE Trans. Veh. Technol. 59 (2) (2010) 896–905.
[7] M. Wolkerstorfer, T. Nordstrom, Heuristics for discrete power control: a case-study in multi-carrier DSL networks, in: ALIO/EURO Workshop on Applied Combinatorial Optimization 2011, Porto, Portugal, 2011, pp. 1–4.
[8] M. Wolkerstorfer, T. Nordstrom, Coverage optimization in DSL networks by low-complexity discrete spectrum balancing, in: IEEE Global Communications Conference (GLOBECOM), Houston, TX, USA, 2011, pp. 1–6.
[9] D. Wolpert, W. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82.
[10] M. Wolkerstorfer, J. Jaldén, T. Nordstrom, Column generation for discrete-rate multi-user and multi-carrier power control, IEEE Trans. Commun. 60 (9) (2012) 2712–2722.
[11] P. Belotti, Couenne: a user's manual, Technical Report, Lehigh University, 2009.
[12] P. Belotti, J. Lee, L. Liberti, F. Margot, A. Wächter, Branching and bounds tightening techniques for non-convex MINLP, Optim. Methods Softw. 24 (4–5) (2009) 597–634.
[13] T. Morin, R. Marsten, Branch-and-bound strategies for dynamic programming, Oper. Res. 24 (4) (1976) 611–627.
[14] M. Barton, M. Honig, Spectral optimization of discrete multitone system on twisted wire copper plants, in: IEEE International Conference on Communications (ICC), vol. 3, Seattle, USA, 1995, pp. 1668–1672.
[15] R. Cendrillon, W. Yu, M. Moonen, J. Verlinden, T. Bostoen, Optimal multiuser spectrum balancing for digital subscriber lines, IEEE Trans. Commun. 54 (5) (2006) 922–933.
[16] M. Wolkerstorfer, J. Jaldén, T. Nordstrom, Low-complexity optimal discrete-rate spectrum balancing in digital subscriber lines, Signal Process. 93 (1) (2013) 23–34.
[17] P. Tsiaflakis, J. Vangorp, M. Moonen, J. Verlinden, A low complexity optimal spectrum balancing algorithm for digital subscriber lines, Signal Process. 87 (7) (2007) 1735–1753.
[18] P. Golden, H. Dedieu, K. Jacobsen (Eds.), Fundamentals of DSL Technology, Auerbach Publications, Boca Raton, New York, 2006.
[19] J. Papandriopoulos, J.S. Evans, Low-complexity distributed algorithms for spectrum balancing in multi-user DSL networks, in: IEEE International Conference on Communications (ICC), vol. 7, Istanbul, Turkey, 2006, pp. 3270–3275.
[20] P. Tsiaflakis, M. Diehl, M. Moonen, Distributed spectrum management algorithms for multiuser DSL networks, IEEE Trans. Signal Process. 56 (10) (2008) 4825–4843.
[21] R. Yates, A framework for uplink power control in cellular radio systems, IEEE J. Sel. Areas Commun. 13 (7) (1995) 1341–1347.
[22] T. Nordstrom et al., xDSL simulator v3.1, xdsl.ftw.at, 2008.
[23] K. Price, R. Storn, J. Lampinen, Differential Evolution, Springer, Berlin, Heidelberg, 2005.
[24] G. Clarke, J. Wright, Scheduling of vehicles from a central depot to a number of delivery points, Oper. Res. 12 (4) (1964) 568–581.
[25] F. Glover, G. Kochenberger (Eds.), Handbook of Metaheuristics, Kluwer Academic Publishers, New York, 2003.
[26] M. Dorigo, L. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Trans. Evol. Comput. 1 (1) (1997) 53–66.
[27] D. Bertsekas, Rollout algorithms for discrete optimization: a survey, in: P. Pardalos, et al., (Eds.), Handbook of Combinatorial Optimization, Springer, New York, 2013, pp. 2989–3013.
[28] D. Bertsimas, J. Tsitsiklis, Simulated annealing, Stat. Sci. 8 (1) (1993) 10–15.
[29] ETSI, Transmission and Multiplexing (TM); Access transmission systems on metallic access cables; Very high speed Digital Subscriber Line (VDSL); Part 1: Functional requirements, Technical Report TM6 TS 101 270-1, Version 1.3.1, ETSI, 2003.
[30] D.S. Johnson, L.A. McGeoch, The traveling salesman problem: a case study in local optimization, in: E.H.L. Aarts, J.K. Lenstra (Eds.), Local Search in Combinatorial Optimization, John Wiley and Sons, Ltd., New York, 1997, pp. 215–310.