The Linear Ordering Problem with Cumulative Costs

Livio Bertacco, Lorenzo Brunetta, Matteo Fischetti
Department of Information Engineering, University of Padova
via Gradenigo 6A - 35131 Padova - Italy
e-mail: {livio.bertacco,lorenzo.brunetta,matteo.fischetti}@unipd.it

October 13, 2004

Abstract

Several optimization problems require finding a permutation of a given set of items that minimizes a certain cost function. These problems are naturally modelled in graph-theory terms by introducing a complete digraph $G = (V, A)$ whose vertices $v \in V := \{1, \cdots, n\}$ correspond to the n items to be sorted. Depending on the cost function to be be used, different optimization problems can be defined on G. The most familiar one is the min-cost Hamiltonian path problem (or its closed-path version, the Travelling Salesman Problem), arising when the cost of a given permutation only depends on consecutive node pairs. A more complex situation arises when a given cost has to be paid whenever an item is ranked before another one in the final permutation. In this case, a feasible solution is associated with an acyclic tournament (the transitive closure of an Hamiltonian path), and the resulting problem is known as the Linear Ordering Problem.

In this paper we introduce and study, for the first time, a relevant case arising when the overall permutation cost can be expressed as the sum of terms α_u associated with each item u, each defined as a linear combination of the values α_v of all items v that follow u in the permutation. This setting implies a cumulative (non-linear) propagation of the value of variables α_v along the node permutation, hence the name Linear Ordering Problem with Cumulative Costs.

We illustrate the practical application that motivated the present study, namely the optimization (through the so-called Successive Interference Cancellation method) of UMTS mobile-phone telecommunication system. We prove complexity results, and propose a Mixed-Integer Linear Programming model as well as an ad-hoc enumerative algorithm for the exact solution of the problem. Extensive computational results on large sets of instances are presented, showing that the proposed techniques are capable of solving, in reasonable computing times, all the instances coming from our application.

Key words: Liner Ordering Problems, MIP models, enumerative search, Computational analysis, Telecommunication systems.

1 Introduction

Several optimization problems require finding a permutation of a given set of items that minimizes a certain cost function. These problems are naturally modelled in graph-theory terms by introducing a complete (loopless) digraph $G = (V, A)$ whose vertices $v \in V := \{1, \cdots, n\}$ correspond to the n items to be sorted. By construction, there is a 1-1 correspondence between the Hamiltonian paths $P = \{(k_1, k_2), \cdots, (k_{n-1}, k_n)\}$ in G (viewed as arc sets) and the item permutations $\mathcal{K} = \langle k_1, \cdots, k_n \rangle$.

Depending on the cost function to be be used, different optimization problems can be defined on G. The most familiar one arises when the cost of a given permutation $\mathcal{K}$ only depends on the consecutive pairs (k_i, k_{i+1}), $i = 1, \cdots, n-1$. In this case, one can typically associate a cost c_{uv} with each arc $(u, v) \in A$, and the problem reduces to finding a min-cost *Hamiltonian Path* (HP) in G, a relative of the famous Travelling Salesman Problem (TSP) [10, 6]. Note however that this model is only appropriate when the overall cost is simply the sum of the "direct costs" of putting an item right after another in the final permutation. A more complex situation arises when a given cost g_{uv} has to be paid whenever item u is ranked before item v in the final permutation. In this case, a feasible solution can be more conveniently associated with an *acyclic tournament*, defined as the transitive closure of an Hamiltonian path $P = \{(k_1, k_2), \cdots, (k_{n-1}, k_n)\}$:

$$[P] := \{(k_i, k_j) \in A : i = 1, \cdots, n-1, j = i + 1, \cdots, n\}$$

see Figure for an illustration. The resulting problem then calls for a min-cost acyclic tournament in G, and is known as the *Linear Ordering Problem* (LOP) [3, 4, 5, 13]. Both HP and LOP are known to be $\mathcal{NP}$-hard problems.

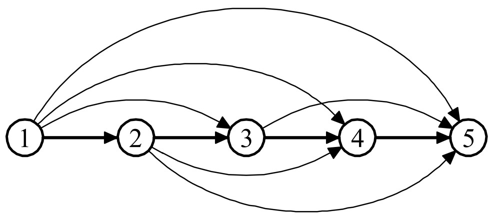


Figure 1: Acyclic tournaments are made by an Hamiltonian path (thick arcs) plus its transitive closure (thin arcs)

In some applications, both the HP and the LOP frameworks are unappropriate to describe the cost function. In this paper we introduce and study, for the first time, a relevant case arising when the overall permutation cost can be expressed as the sum of terms α_u associated with each item u, each defined as a linear combination of the values α_v of all items v that follow u in the permutation. To be more specific, we address the following problem:

Definition 1.1 (LOP-CC). *Given a complete digraph $G = (V, A)$ with nonnegative node weights p_v and nonnegative arc costs c_{uv}, the* Linear Ordering Problem with Cumulative Costs *(LOP-CC) is to find an Hamiltonian path $P = \{(k_1, k_2), \cdots, (k_{n-1}, k_n)\}$ and the corresponding node values α_v that minimize the total cost*

$$\pi(P) = \sum_{v=1}^{n} \alpha_v$$

under the constraints

$$\alpha_{k_i} = p_{k_i} + \sum_{j=i+1}^{n} c_{k_i k_j} \alpha_{k_j}, \quad \text{for } i = n, n-1, \cdots, 1 \tag{1}$$

Constraints (1) imply a *cumulative* "backward propagation" of the value of variables α_v for $v = n, n-1, \cdots, 1$, hence the name of the problem. We will also address a constrained version of the same problem, namely:

Definition 1.2 (BLOP-CC). *The* Bounded Linear Ordering Problem with Cumulative Costs *(BLOP-CC) is defined as the problem LOP-CC above, plus the additional constraints:*

$$\alpha_i \leq U \quad \forall i \in V \tag{2}$$

where U is a given nonnegative bound.

Notice that BLOP-CC can be infeasible. As shown in the next section, BLOP-CC finds important practical applications, in particular, in the optimization of mobile telecommunication systems.

In this paper we introduce and study both problems LOP-CC and BLOP-CC. In Section 2, we give the practical application that motivated the present study and leaded to the patented new methodology for cellular phone management described in [2]. In Section 3, we show that both LOP-CC and BLOP-CC are $\mathcal{NP}$-hard. A Mixed-Integer linear Programming (MIP) model is presented in Section 4, whereas an ad-hoc enumerative method is introduced in Section 5. Extensive computational results on a large set of instances are presented in Section 6, whereas some conclusions are drawn in Section 7.

As G is assumed to be complete, in the sequel we will not distinguish between an Hamiltonian path $P = \{(k_1, k_2), \cdots, (k_{n-1}, k_n)\}$ and the associated node permutation $\mathcal{K} = \langle k_1, \cdots, k_n \rangle$. Moreover, given any Hamiltonian path $P = \{(k_1, k_2), \cdots, (k_{n-1}, k_n)\}$, we call *direct* all arcs $(k_i, k_{i+1}) \in P$ (the thick ones in Figure 1), whereas the arcs (k_i, k_j) for $j \geq i+1$ are called *transitive* (these are precisely the arcs in $[P] \setminus P$, depicted in thin line in Figure 1). Finally, we use notation $\pi(P)$ to denote the *cumulative cost* of an Hamiltonian path P, defined as the LOP-CC cost $\pi = \sum_{v=1}^{n} \alpha_v$ of the corresponding permutation.

2 Motivation

In this section we outline the practical problem that motivated the present paper; the interested reader is referred to [1], [9] and [12] for more details.

In wireless cellular communications, mobile terminals (MTs) communicate simultaneously with a common Base Station (BS). In order to distinguish among the signals of different MTs, the Universal Mobile Telecommunication Standard (UMTS) [15] adopts the so-called code division multiple access technique, where each terminal is identified by a specific code. Due to the distortions introduced by radio propagation, the MTs partially interfere with each other, hence the need to keep the multiuser access interference below an acceptable level. A very effective technique for interference reduction has been proposed [11], and is called Successive Interference Cancellation (SIC). According to this method, MT signals are detected sequentially from the received signal, according to a predetermined order. After each detection, interference is removed from the received signal, thus allowing for improved detection for the next users.

A crucial problem in the design of the SIC system is therefore the choice of the detection order. Usually, users are ordered by decreasing received power [11], although a better performance can be obtained by considering also the level of mutual interference among users. A second issue is the choice of the power level α_i at which the i-th user has to transmit its data. Indeed, a large power level typically allows for an improved signal detection, whereas the minimization of the transmission power yields a longer duration of the batteries of the MT.[1] Moreover, physical and regulatory constraints impose an upper bound, U, on the transmission power of the mobile terminals.

Both the choice of the cancellation order and of the transmission power levels must ensure a reliable detection of the signals coming from all MTs. A proper reception is ensured when the average Signal-to-Noise (plus Interference) power ratio (SNIR) is equal to a target level Γ. For a SIC receiver, the SNIR is related to the power of the interference generated from user i on user j, denoted by ρ_{ij}. In particular, upon detection of user k_p the SNIR is

$$SNIR^{(p)} = \frac{\alpha_{k_p} \rho_{k_p k_p}}{N_0 \sqrt{\rho_{k_p k_p}} + \sum_{i \in \mathcal{U}_p} \alpha_i N_{\mathcal{S}} \rho_{i k_p}} \tag{3}$$

[1] Battery lifetime is one of the main limiting factors for mobile communication systems

where N_0 (noise power) and $N_{\mathcal{S}}$ (spreading factor) are given parameters, and

$$\mathcal{U}_p = \{k_{p+1}, k_{p+2}, \cdots, k_n\} \tag{4}$$

is the set of undetected user at stage p.

One then faces the problem of jointly optimizing the SIC detection order and the transmission power levels, with the aim of minimizing the overall transmission power while ensuring a proper reception for all users. This problem, called joint power-control and receiver optimization (JOPCO), has been introduced in [1], and can be formalized as follows: given a set of users $\{1, 2, \ldots, n\}$, the interference factors ρ_{ij} $(i, j = 1, \ldots, n)$, the noise power N_0, the spreading factor $N_{\mathcal{S}}$, the target ratio Γ, and the maximum allowed power level U, find the transmission power levels α_i $(i = 1, \cdots, n)$ and the detection permutation $\mathcal{K} = \langle k_1, \ldots, k_n \rangle$ that minimize the total transmission power $\pi = \sum_{i=1}^{n} \alpha_{k_i}$ under the following constraints:

$$\Gamma = \frac{\alpha_{k_i}\rho_{k_i k_i}}{N_0\sqrt{\rho_{k_i k_i}} + \sum_{l \in \mathcal{U}_i} \alpha_l N_{\mathcal{S}} \rho_{l k_i}}, \quad \text{for } i = 1, \cdots, n \tag{5}$$

$$\alpha_i \leq U \tag{6}$$

In [1] a simple GRASP heuristic is proposed for JOPCO with the aim of minimizing the system transmit power under the constraint of ensuring the same quality of the transmission (measured by the average raw Bit Error Rate, BER) to all users. In particular, the requirement on the BER is translated into a constraint on the SNIR at the detection point of each user, as discussed before. Extensive experiments are reported, showing that the JOPCO technique performs much better than the usual Average Power (AP) approach in all the four scenarios simulated, both in terms of quality of the transmission (BER) and of allocated transmission power. In particular, Figure 2 (taken from [1]) illustrates the average BER vs. the number n of active users for so-called *synchronous* and *asynchronous* transmission systems, and compares the JOPCO and AP methodologies. Thin and bold lines correspond to the case with and without the so-called *scrambling* operation on transmitted data, respectively.

It can be seen that, both with and without scrambling, JOPCO ensures approximately a constant average raw BER of 10^{-3} up to 10 active users with respect to the classical AP technique. In the case of syncronous transmission without scrambling, AP gives a BER that is even lower than the target, just because it allocates much more transmission power than necessary to guarantee the target quality, as can be seen in Figure 3 (top). When a larger number of active users is present, instead, JOPCO has slight performance degradation due to errors in the interference cancellation.

The JOPCO performance is indeed very good up to 10 active users, as the average BER is below 10^{-3}. When a larger number of active users is present, instead, we have a performance degradation due to interference effects.

Figure 3 (also taken from [1]) gives the expected total transmission power ratio expressed in dB, η, as a function of the number of active users (with and without scrambling), i.e.,

$$\eta = 10 \log_{10} \mathrm{E}\left[\frac{P_{\mathrm{tot}}^{(AP)}}{P_{\mathrm{tot}}^{(JOPCO)}} \right] . \tag{7}$$

where $P_{\mathrm{tot}}^{(AP)}$ and $P_{\mathrm{tot}}^{(JOPCO)}$ represent the total transmission power allocated by AP and JOPCO, respectively.

In all cases, we observe that the JOPCO approach requires a reduced transmission power with respect to the AP approach. In particular, for a full loaded synchronous (resp., asynchronous) system without scrambling, on average the system power requirement using the JOPCO technique is 7 dB (resp., 3 dB) lower than that for AP. When scrambling is considered, instead, the JOPCO power requirement is 3 dB (resp., 2dB) lower than that for AP.

We next show how JOPCO can be formulated as a BLOP-CC. Clearly, for any given user permutation $\mathcal{K}$ the power levels α_i are univocally determined by the SNIR constraints (5). Indeed, rewriting (5) as

$$\alpha_{k_i} = \frac{\Gamma N_0 \sqrt{\rho_{k_i k_i}} + \Gamma N_{\mathcal{S}} \sum_{l \in \mathcal{U}_i} \alpha_l \rho_{l k_i}}{\rho_{k_i k_i}}$$

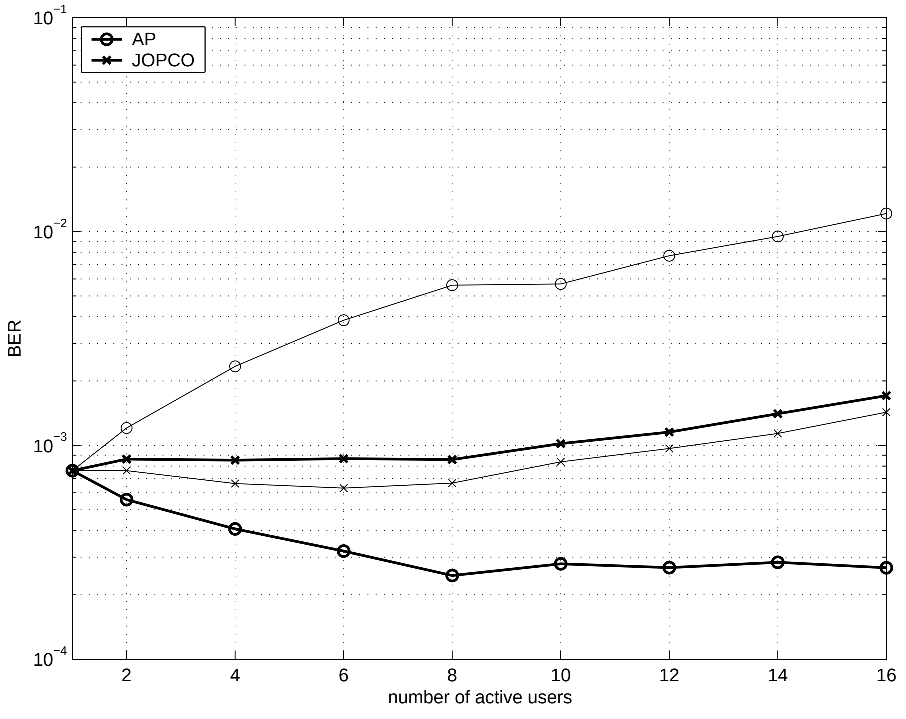
one has that values α_{k_i} can easily be computed in the reverse order $i = n, n-1, \cdots, 1$. Defining the weights $p_i = \Gamma N_0 / \sqrt{\rho_{ii}}$ and the costs $c_{ij} = \Gamma N_{\mathcal{S}} \rho_{ji} / \rho_{ii}$ one then obtains precisely the BLOP-CC formulation introduced in the previous section.

3 Complexity of the LOP-CC

We start proving that the LOP-CC problem is $\mathcal{NP}$-hard. We first give a simple outline of the proof, and then address in a formal proof the details required.

Our proof is by reduction from the following Hamiltonian Path problem, HP, which is known to be $\mathcal{NP}$-complete.

Definition 3.1 (HP). *Given a digraph $G_{HP} = (V_{HP}, A_{HP})$, decide whether G contains any directed Hamiltonian path.*



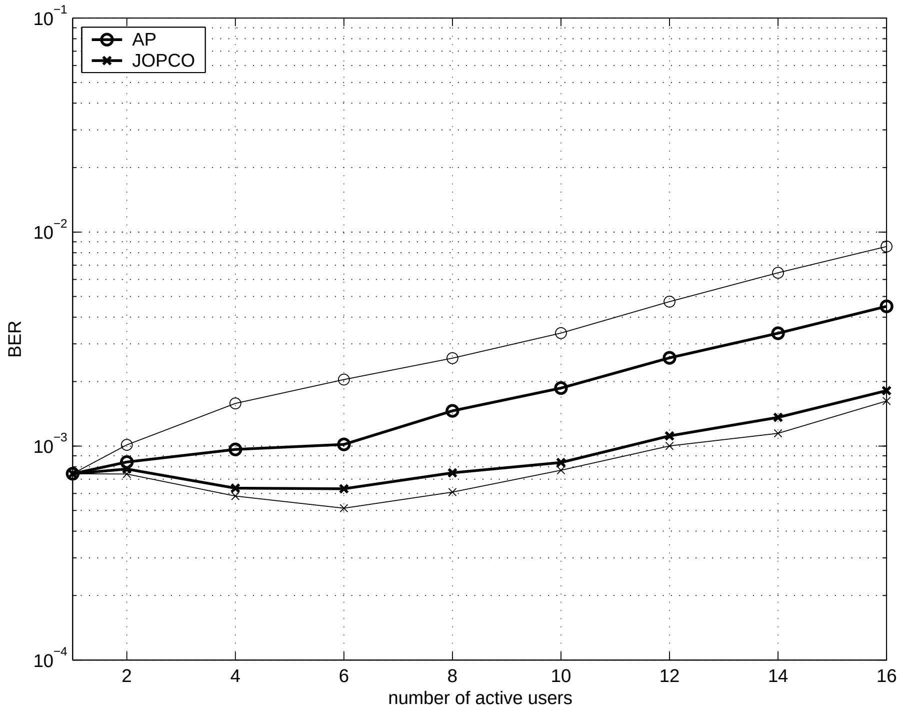
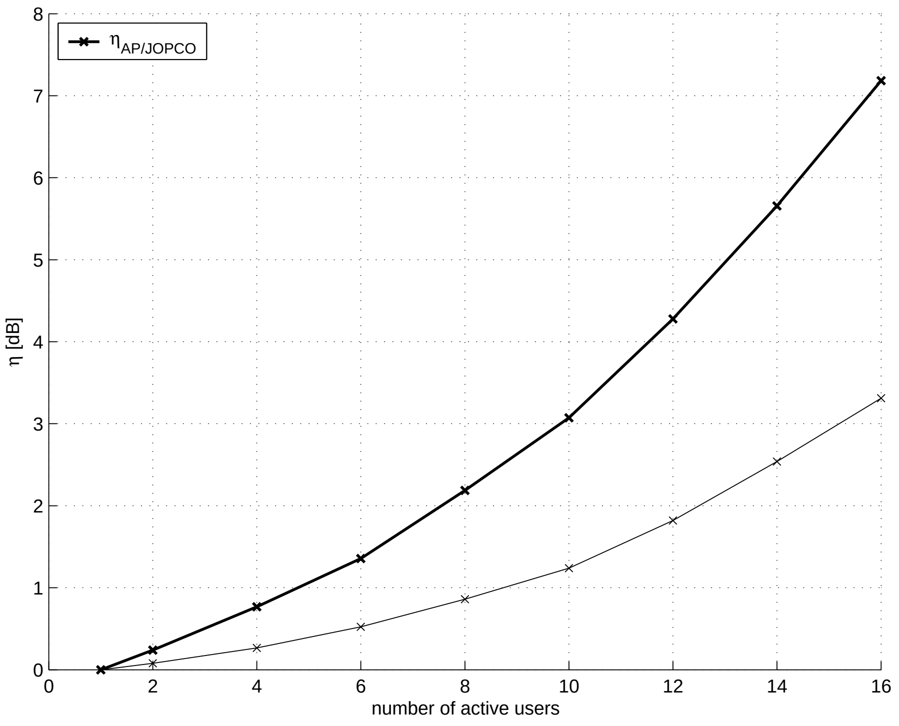


Figure 2: The average raw bit error rate BER vs. the number of active users for synchronous (top) and asynchronous (bottom) transmissions, with scrambling (thin line) and without scrambling (bold line)



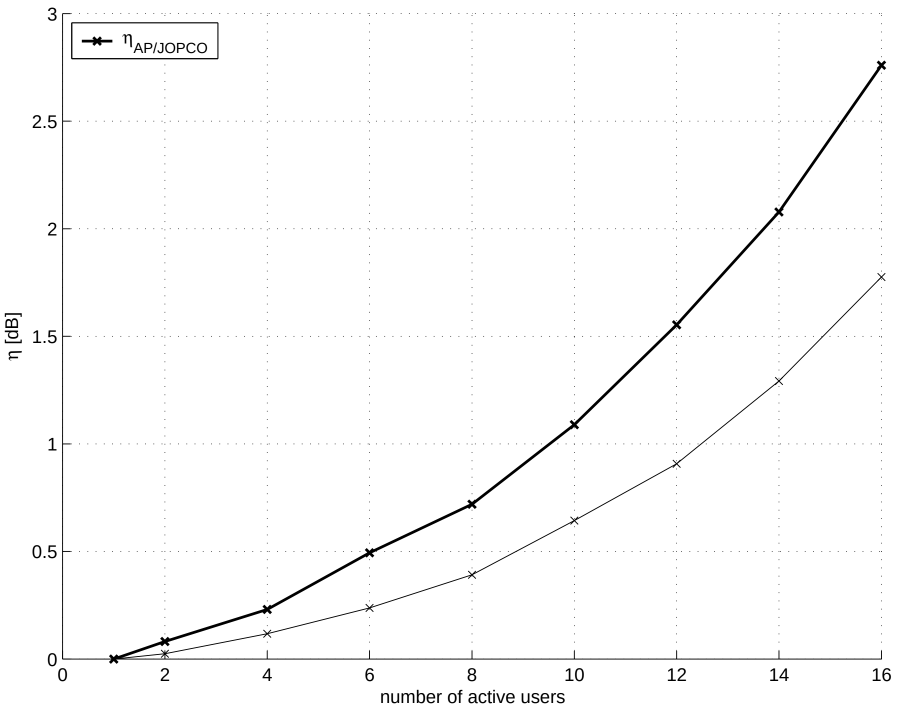


Figure 3: The expected total transmission power ratio η vs. the number of active users for synchronous (top) and asynchronous (bottom) transmissions, with scrambling (thin line) and without scrambling (bold line)

Our reduction takes any HP instance $G_{HP} = (V_{HP}, A_{HP})$ and computes the following LOP-CC instance:

$$
\begin{aligned}
V &:= V_{HP} = \{1, \cdots, n\} && (8)\\
p_i &:= 1 \quad \forall i \in V && (9)\\
c_{ij} &= \begin{cases} M & \text{if } (i,j) \in A_{HP} \\ 2M & \text{otherwise} \end{cases} \quad \forall i, j \in V,\, i \neq j && (10)
\end{aligned}
$$

where M is a sufficiently large positive value (to be defined later). The construction can clearly be carried out in polynomial time, provided that value M can be stored in a polynomial number of bits a property that will be asserted in the formal proof.

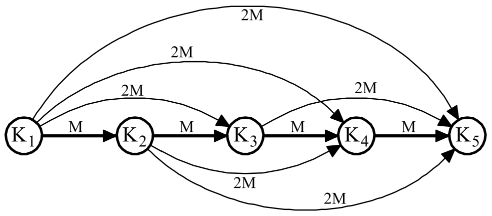


Figure 4: The worst-case "good" path used in the complexity proof

"Good" Hamiltonian paths of G (i.e., those corresponding to Hamiltonian paths in G_{HP}) only involve direct arcs of cost equal to M, while all the transitive arcs have a cost not larger than $2M$; see Figure 4 for an illustration. It is therefore not difficult to show that, for sufficiently large M, the overall cumulative cost associated with such a path is $\pi(P) = M^{n-1} + O(M^{n-2})$. On the other hand, if P does not correspond to a Hamiltonian path in G_{HP}, then it has to involve at least one direct arc of cost $2M$, hence its cumulative cost $\pi(P)$ cannot be smaller than $2M^{n-1}$. For a large M, our construction then ensures that $\pi(P) < \pi(P')$ for any "good" Hamiltonian path P and for any "non-good" Hamiltonian path P', which implies that G_{HP} contains an Hamiltonian path if and only if any arbitrary optimal LOP-CC solution corresponds to a "good" Hamiltonian path (or, equivalently, if the optimal LOP-CC value is strictly less than $2M^{n-1}$).

Theorem 3.1. *LOP-CC is $\mathcal{NP}$-hard*

Proof. Given the transformation above, our formal proof amounts to establishing an upper bound $UB_{good}(n, M)$ on the cumulative cost $\pi(P)$ of any "good" Hamiltonian path P as well as a lower bound $LB_{nogood}(n, M)$ on the cumulative cost $\pi(P')$ of any "non-good" Hamiltonian path P', and to show that $UB_{good}(n, M) < LB_{nogood}(n, M)$ for all n and for a value of M such that $\log(M)$ is polynomial in n.

The lower bound $LB_{nogood}(n, M)$ corresponds to the case where only one direct arc in P' has cost $2M$, hence it can be computed in a straightforward way as

$$LB_{good}(n, M) = 2M^{n-1} \tag{11}$$

As to upper bound $UB_{good}(n, M)$, it is computed by considering the cumulative cost of a Hamiltonian path P where all direct arcs have cost M, whereas all transitive arcs have cost $2M$. This case is illustrated in Figure 4. To be more precise, we claim that

$$\pi(P_n) \le UB_{good}(n, M) := M^{n-1} + 4^n M^{n-2} \tag{12}$$

holds for any Hamiltonian path P_n in G whose direct arcs all have cost M, where $M > 1$ is assumed. The proof of this claim is by induction on n. The claim clearly holds in cases $n = 1$ and $n = 2$, where we have $\pi(P_1) = 1$ and $\pi(P_2) = M + 2$, respectively. We assume now that (12) holds for all $n \le h$ for a given $h \ge 2$, and we prove that it also holds for $n = h+1$. Let $P_{n=h+1} = \{(k_1, k_2), \cdots, (k_h, k_{h+1})\}$ be any Hamiltonian path whose direct arcs all have cost M, and let $P_h = \{(k_2, k_3), \cdots, (k_h, k_{h+1}\}$ and $P_{h-1} = \{(k_3, k_4), \cdots, (k_h, k_{h+1}\}$ be obtained from P_{h+1} by removing its first arc and its first two arcs, respectively. We have

$$\begin{aligned}
\pi(P_{h+1}) &= \sum_{i=1}^{h+1} \alpha_{k_i} = \sum_{i=2}^{h+1} \alpha_{k_i} + \alpha_{k_1} \\
&= \pi(P_h) + \alpha_{k_1} \\
&\le \pi(P_h) + M\alpha_{k_2} + 1 + 2M \sum_{i=3}^{h+1} \alpha_{k_i} \quad \text{(because of (1) and (10))} \\
&\le \pi(P_h) + M\pi(P_h) \quad (\text{since } \pi(P_h) \ge \alpha_{k_2} + 1) \\
&\qquad + 2M\,\pi(P_{h-1}) \quad (\text{since } \pi(P_{h-1}) = \textstyle\sum_{i=3}^{h+1} \alpha_{k_i}) \\
&\le (1 + M)\pi(P_h) + 2M\pi(P_{h-1})
\end{aligned}$$

The claim then follows from the induction hypothesis, as we have

$$\begin{aligned}
\pi(P_{h+1}) &\le (1 + M)(M^{h-1} + 4^h M^{h-2}) + 2M(M^{h-2} + 4^{h-1} M^{h-3}) \\
&\le M^h + (3 + 4^h)M^{h-1} + \frac{3}{2} 4^h M^{h-2} \qquad (13) \\
&\le M^h + (3 + \frac{5}{2} 4^h)M^{h-1} \quad (\text{since } M^{h-2} \le M^{h-1}) \qquad (14) \\
&\le M^h + 4^{h+1} M^{h-1} \quad (\text{since } h \ge 2) \qquad (15)
\end{aligned}$$

To complete the complexity proof, we have to choose a value for M that guarantees $UB_{good}(n, M) < LB_{nogood}(n, M)$, i.e.,

$$M^{n-1} + 4^n M^{n-2} < 2M^{n-1} \Longrightarrow 4^n M^{n-2} < M^{n-1} \Longrightarrow 4^n < M$$

We then set $M = 4^n + 1$, whose size $log(M) = O(n)$ is polynomial in n, as required. □

Corollary 3.1.1. *BLOP-CC is $\mathcal{NP}$-hard*

Proof. We use the same construction as in the proof of the previous theorem, with $U := UB_{good}(n, M) = O(M^n) = O(4^{n^2})$ large enough to make all "good" Hamiltonian paths in G feasible for the BLOP-CC instance, but still with size $log(U) = O(n^2)$. □

4 A MIP model

In this section we introduce a MIP model for BLOP-CC, derived from the LOP model of Grötschel, Jünger and Reinelt [3].

As already mentioned, in a standard linear ordering problem we have n items to be placed in a convenient order. If we place item i before item j, we pay a cost of g_{ij}. The objective is to choose the item order that minimizes the total cost. This problem can then be modelled as

$$\begin{array}{rl} \min & \sum_{(i,j)\in A} g_{ij}x_{ij} \\ \text{subject to} & \text{“}x\text{ is the incidence vector of an acyclic tournment”} \end{array}$$

where $x_{ij} = 1$ if item i is placed before j in the final order, $x_{ij} = 0$ otherwise.

In order to get an acyclic tournament, it is shown in [3] that, besides the obvious conditions

$$x_{ij} + x_{ji} = 1, \quad \forall (i,j) \in A, i < j \tag{16}$$

it is sufficient to prevent 3-node cycles of the form $x_{ij} = x_{jk} = x_{ki} = 1$, leading to the *triangle inequalities*

$$x_{ij} + x_{jk} + x_{ki} \leq 2 \tag{17}$$

Using the same set of variables, one can rewrite the LOP-CC constraints (1) as the nonlinear equalities:

$$\alpha_i = p_i + \sum_{j=1}^{n} c_{ij}\alpha_j x_{ij} \quad \forall i \in V \tag{18}$$

In order to get linear constraints, we introduce the following $n(n-1)$ new variables:

$$y_{ij} (= \alpha_j x_{ij}) = \begin{cases} \alpha_j & \text{if } x_{ij} = 1 \\ 0 & \text{otherwise} \end{cases} \quad \forall i, j \in V : i \neq j \tag{19}$$

Thus (18) becomes linear in the y variables,

$$\alpha_i = p_i + \sum_{j=1}^{n} c_{ij} y_{ij}$$

and conditions (19) become

$$\begin{array}{lcl} x_{ij} = 0 \Rightarrow y_{ij} = 0 & \longrightarrow & y_{ij} \le M x_{ij} \\ x_{ij} = 1 \Rightarrow y_{ij} \ge \alpha_j & \longrightarrow & y_{ij} \ge \alpha_j - M(1 - x_{ij}) \\ x_{ij} = 1 \Rightarrow y_{ij} \le \alpha_j & \longrightarrow & y_{ij} \le \alpha_j + M(1 - x_{ij}) \end{array}$$

where M is a sufficiently large positive value. Notice that constraints $y_{ij} \le \alpha_j + M(1 - x_{ij})$ can be removed from the model, as the minimization of variables α_j implies that of variables y_{ij}. (As to constraints $y_{ij} \le M x_{ij}$, they are redundant as well; however, our computational experience showed that they improve the numerical stability of the MIP solver, so we keep them into our model.) Also, from (19), the y variables are bounded by the α variables, thus we can take $M = U$. Finally, $\alpha \ge 0$ can be assumed since $p, c \ge 0$).

As it is customary in LOP models, one can use equations (16) to eliminate all variables x_{ij} with $i > j$,[2] and modify the triangle inequalities into:

$$\begin{array}{ll} x_{ij} + x_{jk} - x_{ik} \le 1 & \text{for all } 1 \le i < j < k \le n \\ -x_{ij} - x_{jk} + x_{ik} \le 0 & \text{for all } 1 \le i < j < k \le n \end{array}$$

This leads to the following MIP model for BLOP-CC:

$$\begin{array}{rlr} \text{minimize} & \sum_{i=1}^{n} \alpha_i & \\ \text{subject to} & x_{ij} + x_{jk} - x_{ik} \le 1 & \text{for } 1 \le i < j < k \le n \\ & x_{ij} + x_{jk} - x_{ik} \ge 0 & \text{for } 1 \le i < j < k \le n \\ & \alpha_i = p_i + \sum_{j=1}^{n} c_{ij} y_{ij} & \text{for } 1 \le i \le n \\ & y_{ij} \le U x_{ij} & \text{for } 1 \le i < j \le n \\ & y_{ji} \le U(1 - x_{ij}) & \text{for } 1 \le i < j \le n \\ & y_{ij} \ge \alpha_j - U(1 - x_{ij}) & \text{for } 1 \le i < j \le n \\ & y_{ji} \ge \alpha_i - U x_{ij} & \text{for } 1 \le i < j \le n \\ & 0 \le \alpha_i \le U & \text{for } 1 \le i \le n \\ & y_{ij} \ge 0 & \text{for } 1 \le i \ne j \le n \\ & x_{ij} \in \{0, 1\} & \text{for } 1 \le i < j \le n \end{array}$$

5 An ad-hoc enumerative algorithm

Our enumerative algorithm is based on a standard backtracking technique (akin to those used in Constraint Programming) to generate all permutations and to find one with the lowest total cost. To limit the number of permutations actually evaluated, we use a pruning mechanism based on lower bounds.

Permutations are built progressively, and are extended backwards from the last node. At the root of the search tree (depth 0), none of the permutation elements in $\mathcal{K} = \langle k_1, \cdots, k_n \rangle$ is fixed. At depth 1, the last element

[2] The α variables could be removed as well, but this would have a marginal effect on the solution time of the model.

```
Permutation generation:
  TraverseSearchTree(n)
  1.   initialize set $S := \{1, 2, \cdots, n\}$
  2.   EnumeratePermutationElement($n$)

  EnumeratePermutationElement($i$)
  3.   for each element $e$ in $S$ do
  4.     remove $e$ from $S$
  5.     perm[$i$] $\leftarrow$ $e$
  6.     evaluate partial permutation
                 $\langle *, \cdots, *, \mathrm{perm}[i], \mathrm{perm}[i+1], \cdots, \mathrm{perm}[n] \rangle$
  7.     if $i > 1$ then EnumeratePermutationElement($i - 1$)
  8.     insert $e$ back into $S$
  9.   enddo
```

Figure 5: The basic method

of the permutation, k_n, is fixed to one of the n possible choices (thus, the root has n sons). At depth 2, the next to last item, k_{n-1}, is fixed to one of the remaining $n-1$ possible choices, and so on. The search tree is visited in depth-first manner. The only required data structures to implement this method are an array to store the current partial permutation, and another to keep track of the nodes that have not yet been inserted in the current partial permutation.

We chose this method to enumerate permutations, rather than more sophisticated ones, because we can compute very quickly a parametric lower bound for partial permutations (to be used for pruning purposes), thus enumerating very effectively a large number of nodes.

Our lower bound is computed as follows. Given a permutation $\mathcal{K} = \langle k_1, \cdots, k_n \rangle$, we can write the corresponding node values α_v as:

$$\begin{array}{llllllll}
\alpha_{k_n} & = & p_{k_n} & & & & & \\
\alpha_{k_{n-1}} & = & p_{k_{n-1}} & + & \alpha_{k_n} & c_{k_{n-1},k_n} & & \\
\alpha_{k_{n-2}} & = & p_{k_{n-2}} & + & \alpha_{k_n} & c_{k_{n-2},k_n} & + & \alpha_{k_{n-1}} \quad c_{k_{n-2},k_{n-1}} \\
\cdots & & & & & & &
\end{array}$$

and the total permutation cost π is the sum of all the α_v. Notice that all the node weights p_v contribute to the total cost, so their sum can be used as an initial lower bound for the cost of any permutation.

Assume now that nodes $k_{i+1}, k_{i+2}, \cdots, k_n$ have already been chosen. When node k_i is also chosen, one can easily compute the corresponding α_{k_i} by using equation (1). Furthermore, the contribution of this α_{k_i} to the final cost of any permutation of the type $\langle *, \cdots, *, k_i, k_{i+1}, \cdots, k_n \rangle$ is given by:

$$\alpha_{k_i} \sum_{u \notin \{k_i, \cdots, k_n\}} c_{uk_i} \tag{20}$$

regardless of the rank of the remaining nodes in the complete permutation. This property allows us to compute easily, in a parametric way, a valid lower bound on the cost of any such permutation.

To be more specific, our pruning mechanism works as follows. We start with lower bound $LB := \sum_v p_v$ and with an empty permutation. We then build-up partial permutations recursively. Every time a new node k_i is inserted in front of the current permutation, we compute the corresponding α_{k_i} and add (20) to the current lower bound LB. If the resulting LB is strictly smaller than the incumbent solution value, we proceed with the recursion; otherwise we backtrack, and update LB accordingly.

A small modification of this algorithm can be used to start with a given permutation (rather than from scratch), thus allowing the enumerative search to quickly produce improved permutations through successive modifications of the starting one. To this end, the only change required at Step 3 of the algorithm of Figure 5 is to implement the set S as a FIFO queue, to be initialized with the desired starting permutation (the last node in the sequence being the first to be extracted).

In our implementation we use a simple yet effective heuristic to find a "good" initial solution for the BLOP-CC instances arising in our telecommunication application. Namely, we sort the nodes by decreasing autocorrelation factors ρ_{ii}, and consider the associated permutation (the first and last node in the permutation being those with the largest and smallest ρ_{ii}, respectively) to initialize our incumbent solution. Alternative heuristics to find a good initial permutation are presented in [1].

Our complete algorithm is outlined in Figure 6.

6 Computational results

We have compared the performance of our enumerative algorithm with that of a state-of-the-art commercial MIP solver (`ILOG-Cplex` 9.0.2 [7]) applied to the MIP model of Section 4. The outcome of this experiment is reported in Table 1, where 4 large sets of BLOP-CC instances have been considered.

The instances used in our study have been provided by the telecommunications group of the Engineering School of the University of Padova, and are related to detection-order optimization in UMTS networks [1]. All the four communication scenarios considered in [1] have been addressed: synchronous and asynchronous transmission, with and without scrambling. For each scenario, 500 matrices (ρ_{ij}) have been randomly generated assuming the presence of 16 users[3] uniformly distributed in the cell (with a

[3]The UMTS technology considers up to 8-16 active users at a time; a larger number of

```
Optimization by backtracking:
  1.   find a starting heuristic solution $H$
  2.   initialize the fifo queue $Q$ with the elements in $H$
  3.   LB $\leftarrow \sum_{i=1}^{n}$ p[i]
  4.   UB $\leftarrow \infty$
  5.   EnumeratePermutationElement($n$, LB)
  6.   output bestperm

  EnumeratePermutationElement(i, LB)
  7.   repeat i times
  8.     q $\leftarrow$ pop $Q$
  9.     perm[$i$] $\leftarrow$ q
 10.     calculate alpha[q]
 11.     if i = 1 then
 12.       bestperm $\leftarrow$ perm
 13.       UB $\leftarrow$ LB
 14.     else
 15.       LB $\leftarrow$ LB +  alpha[q] * $\sum_{r\in Q}$c[r,q]
 16.       if LB<UB then EnumeratePermutationElement(i-1, LB)
 17.     endif
 18.     push q into $Q$
 19.   enddo
```

Figure 6: The overall enumerative algorithm

radius of 580 m), according to the so-called log-distance path loss model. The input values Γ, U, N_S, and N_0 have been set to 0.625, 10.0, 16, and 0.50476587558415, respectively. From each matrix ρ, we have derived 8 BLOP-CC instances of different sizes ($n = 2, 4, \ldots, 16$), using the equations given at the end of Section 2 to compute the weights p_v and costs c_{uv}. The full set of these matrices ρ is available for download at `http://www.math.unipd.it/~bertacco/LOPCC_instances.zip`

For each instance, the corresponding MIP model is built-up through a C++ code based on `ILOG Concert Technology 2.0` [8]. All the model constraints are statically incorporated in the initial formulation, and the outcome is solved through `ILOG-Cplex` 9.0.2. Since the default `ILOG-Cplex` tolerances are too large to solve correctly even small instances, we used the following parameter setting (all other parameters being left at default values):

- Absolute mipgap tolerance (CPX_PARAM_EPAGAP): 1e-13
- Relative mipgap tolerance (CPX_PARAM_EPGAP): 1e-11
- Integrality tolerance (CPX_PARAM_EPINT): 1e-9
- Optimality tolerance (CPX_PARAM_EPOPT): 1e-9
- Feasibility tolerance (CPX_PARAM_EPRHS): 1e-9

Unfortunately, in some cases these tolerances are not small enough to ensure a valid resolution, even when $n = 2$. The reason is that the ρ matrix contains coefficients that may vary by several orders of magnitude, hence even our feasibility tolerance of 1e-9 can be insufficient. On the other hand, `ILOG-Cplex` does not support smaller values for this parameter, so we could not fix this pathological situation, and we had to report in Table 1 the number of instances that `ILOG-Cplex` could not solve correctly.

Table 1 reports, for each group of instances and for each size n, the following information: the number of instances solved, the average solution times for both our enumerative code (*Enum*) and `ILOG-Cplex` (*MIP*), the speedup of the enumerative code with respect to `ILOG-Cplex`, the maximum solution times, and the number of instances not solved correctly by `ILOG-Cplex` (*# fails*). Computing times are expressed in CPU seconds, and refer to a Pentium M 1.4 Ghz notebook with 512 MBytes of main memory.

The computational results clearly show that our enumerative approach outperforms `ILOG-Cplex`, and is up to three orders of magnitude faster. As a matter of fact, in no instance `ILOG-Cplex` beated the enumerative code. As to scalability, the enumerative code proved capable of solving instances with $n = 20$ in about 4 hours.

active users is unrealistic for this application

Set 1: Synchronous communication without scrambling

	# instances		average time (sec.s)		Enum	max time (sec.s)		
size	Enum	MIP	Enum	MIP	speedup	Enum	MIP	# fails
2	500	500	-	0.0		-	0.0	2
4	500	500	-	0.0		0.0	0.0	2
6	500	500	-	0.0		0.0	0.0	7
8	500	500	-	0.0	497	0.0	0.6	7
10	500	500	0.0	0.5	186	0.0	5.3	6
12	500	500	0.0	9.6	787	0.2	455.5	9
14	500	50	0.2	395.0	1,906	12.5	2,537.6	1
16	500	5	4.8	22,233.0	4,574	381.9	60,096.2	1

Set 2: Synchronous communication with scrambling

	# instances		average time (sec.s)		Enum	max time (sec.s)		
size	Enum	MIP	Enum	MIP	speedup	Enum	MIP	# fails
2	500	500	-	0.0		-	0.0	3
4	500	500	-	0.0		-	0.0	5
6	500	500	-	0.0		-	0.0	8
8	500	500	-	0.0		0.0	1.0	7
10	500	500	0.0	0.6	237	0.0	10.0	7
12	500	500	0.0	8.8	441	0.9	319.8	10
14	500	50	0.5	267.4	503	43.1	2,256.9	1
16	500	4	15.0	14,473.2	964	794.6	49,424.0	1

Set 3: Asynchronous communication without scrambling

	# instances		average time (sec.s)		Enum	max time (sec.s)		
size	Enum	MIP	Enum	MIP	speedup	Enum	MIP	# fails
2	500	500	-	0.0		-	0.0	2
4	500	500	-	0.0		-	0.0	3
6	500	500	-	0.0		-	0.0	6
8	500	500	-	0.1		0.0	0.5	4
10	500	500	0.0	1.2	1,212	0.0	13.1	4
12	500	500	0.0	30.0	2,447	0.2	786.6	10
14	500	50	0.2	1,130.8	5,218	11.9	8,058.2	0
16	500	5	5.6	28,191.6	4,978	407.8	61,402.8	1

Set 4: Asynchronous communication with scrambling

	# instances		average time (sec.s)		Enum	max time (sec.s)		
size	Enum	MIP	Enum	MIP	speedup	Enum	MIP	# fails
2	500	500	-	0.0		0.0	0.0	3
4	500	500	-	0.0		0.0	0.0	2
6	500	500	-	0.0		0.0	0.0	3
8	500	500	-	0.1		0.0	0.7	4
10	500	500	0.0	1.3	1,096	0.0	15.8	4
12	500	500	0.0	31.3	2,310	0.2	473.4	8
14	500	50	0.2	1,801.5	6,880	6.6	35,469.5	1
16	500	2	6.3	8,362.0	1,316	204.8	9,821.5	1

Overall statistics

	# instances		average time (sec.s)		Enum	max time (sec.s)		
size	Enum	MIP	Enum	MIP	speedup	Enum	MIP	# fails
2	2000	2000	-	0.0		0.0	0.0	10
4	2000	2000	-	0.0		0.0	0.0	12
6	2000	2000	-	0.0		0.0	0.0	24
8	2000	2000	-	0.1	1,146	0.0	1.0	22
10	2000	2000	0.0	0.9	488	0.0	15.8	21
12	2000	2000	0.0	20.0	1,373	0.9	786.6	37
14	2000	200	0.3	898.7	2,952	43.1	35,469.5	3
16	2000	16	7.9	20,421.2	2,562	794.6	61,402.8	4

Table 1: Computational results

All the instances in our testbed were solved to proven optimality, thus enabling us to benchmark the GRASP heuristic proposed in [1] (evaluating the performance of this method was indeed our initial motivation in studying BLOP-CC).

7 Conclusions

We have introduced and studied, for the first time, a new optimization problem related to the well-known Linear Ordering Problem, in which the solution cost is non-linear due to a cumulative backwards propagation mechanism. This model was motivated by a practical application in UMTS mobile-phone telecommunication system.

We have formalized the problem, in two versions, and proved that they are both $\mathcal{NP}$-hard. We have proposed a Mixed-Integer Linear Programming model as well as an ad-hoc enumerative algorithm for the exact solution of the problem. Extensive computational results on large sets of instances have been presented, showing that the proposed techniques are capable of solving, in reasonable computing times, all the instances coming from our application. As a byproduct, our method allowed to benchmark the GRASP heuristic proposed in [1].

Future research should be devoted to enhancing the MIP formulation, and/or to embed more sophisticated pruning mechanisms in our enumerative scheme. Also worth studying are more complex (nonlinear) cost functions applied to the basic Linear Ordering model.

Acknowledgements

Work supported by MIUR and CNR, Italy. Thanks are due to Nevio Benvenuto, Giambattista Carnevale, and Stefano Tomasin for helpful discussions on the JOPCO methodology.

References

[1] N. Benvenuto, G. Carnevale and S. Tomasin. Joint Power Control and Receiver Optimization of CDMA Transceivers using Successive Interference Cancelation. Technical Report DEI, Department of Information Engineering, University of Padova (2004).

[2] N. Benvenuto and S. Tomasin. On the Comparison Between OFDM and Single Carrier Modulation With a DFE Using a Frequency-Domain Feedforward Filter. IEEE Transactions on Communications 50(6), 947-955, 2002.

[3] M. Grötschel, M. Jünger, and G. Reinelt. A Cutting Plane Algorithm for the Linear Ordering Problem. Operations Research, 32, 1195-1220, 1984.

[4] M. Grötschel, M. Jünger, and G. Reinelt. On the acyclic subgraph polytope. Mathematical Programming, 33, 28-42, 1985.

[5] M. Grötschel, M. Jünger, and G. Reinelt. Facets of the Linear Ordering Polytope. Mathematical Programming, 33, 43-60, 1985.

[6] G. Gutin and A. Punnen (editors) *The Traveling Salesman Problem and its Variations*, Kluwer, 2002.

[7] ILOG Cplex 9.0: User's Manual and Reference Manual, ILOG, S.A., `http://www.ilog.com/`, 2004.

[8] ILOG Concert Technology 2.0: User's Manual and Reference Manual, ILOG, S.A., `http://www.ilog.com/`, 2004.

[9] H. Holma and A. Toskala. *WCDMA for UMTS: Radio Access for Third Generation Mobile Communications*, Wiley, New York, 2000.

[10] J.K. Lenstra, A.H.G. Rinnooy Kan and D.Shmoys (editors). *The Traveling Salesman Problem: A Guided Tour to Combinatorial Optimization,* Wiley, 251-305, 1985.

[11] P. Patel and J. Holzman. Analysis of a Simple Successive Interference Cancellation Scheme in a DS/CDMA System. IEEE J. Select. Areas Commun., 12, 727–807, 1994.

[12] J. G. Proakis. *Digital Communications 4th edition*, McGraw Hill, New York, 2004

[13] G. Reinelt. *The Linear Ordering Problem: Algorithms and Applications.* Research and Exposition in Mathematics 8, Heldermann Verlag, Berlin, 1985

[14] D. Warrier and U. Madhow. On the Capacity of Cellular CDMA with Successive Decoding and Controlled Power Disparities. in Proc. Vehic. Tech. Conf. (VTC), vol. 3, 1873–1877, 1998.

[15] 3GPP, Technical Specification Group Radio Access Networks; Radio Transmission and Reception. 3G TS 25.102 version 3.6.0, 1999.