EWMA-Triggered Waterfilling for Reduced-Complexity Resource Management in ad-hoc Connections

Johannes Gonter, Norbert Goertz, Markus Rupp and Wolfgang Gartner

Institute of Telecommunications
Vienna University of Technology
Gußhausstr. 25-29 / 389, 1040 Vienna, Austria
Email: {johannes.gonter, norbert.goertz, markus.rupp, wolfgang.gartner}@nt.tuwien.ac.at

***Abstract*—This paper introduces a highly efficient waterfilling-based strategy for optimal use of the channel in vehicular or personal ad-hoc communications. The approach provides near optimum allocation of resources and enables small communications devices to establish connections when the energy efficiency is at its best. Instead of calculating the transmit power through conventional waterfilling, the proposed algorithm calculates the waterlevel, thus providing a decision threshold and a strategy for optimum use of the time-variant channel at the same time. The algorithm adapts to the changing average channel quality by applying an exponentially-weighted moving-average (EWMA) trigger to re-calculate the waterlevel. The new algorithm is compared to an efficient non-iterative algorithm that directly calculates the transmit powers in every time-slot. It is shown that the new strategy reduces computation time by approximately 90% compared to the classic approach without compromising performance measures such as transmitted information or energy. Practical implementation is briefly discussed to demonstrate suitability of the algorithm for integration into tomorrow's communication devices.**

***Index Terms*—Efficient Waterfilling, Fixed-Point Algorithm,**

I. Introduction

In emerging communication technologies designed for high-mobility applications, there is a strong demand not only for optimum use of valuable transmit time, but also for reduced computational load on usually small communications devices. The currently intensively researched field of vehicular communication concepts is just the first step towards self-organizing networks of small devices that are designed to communicate over briefly-available wireless links, thus acting as relays, mitigating shadowing effects and increasing network QoS for all users. Vehicular communications and wearable communication technologies share one common problem: the lifetime of an available wireless link may be at best in the order of a few seconds, and it is inherently non-stationary. There is, however, a lack of strategies that deal with the optimum use of these finite lifetime wireless links. In recent publications [1], [2], we have proposed a predictive strategy to efficiently exploit the finite lifetime channel through application of a newly-developed waterfilling algorithm. We assumed the channel quality to be symmetrically increasing and decreasing over time, as can be expected in vehicular communication scenarios, e.g. along a highway. The waterfilling approach was chosen because it guarantees a maximization of transmitted information, while at the same time satisfying a sum-energy constraint. Therefore, waterfilling can also be regarded as the most energy-efficient scheme for transmitting information. This, in turn, leads to reduced consumption of scarce wireless resources, i.e. reduces the time of transmission by identifying and selecting the most efficient time-slots for transmission. This will also reduce interference with other traffic. The downside, however, is the algorithmic time that has to be spent in every time-slot: conventional waterfilling algorithms (for a comprehensive overview, see [3]) have to be run in every time-slot, so that a decision on the use of a time-slot requires costly calculations and takes precious time. In this paper, we suggest the use of our previously-suggested [2] waterfilling algorithm to calculate the waterlevel λ rather than the channel-powers, and update it only if the channel changes enough to justify the computational cost. By focussing on a simple algorithmical decision structure, we can greatly reduce the average number of required calculations per channel use compared to conventional waterfilling schemes, while still delivering virtually the same performance. This paper is organized as follows: Section II discusses the channel model and states the coefficients' statistical properties for the generation of all this paper's examples. Section III presents the waterfilling algorithm developed in [1], [2], as well as the benchmark non-iterative waterfilling-algorithm. Section IV discusses the proposed approach of waterfilling-like performance at greatly reduced computational time, Section V discusses simulation results, and section VI concludes the paper.

II. Channel Model

This paper applies the common block-fading model with additional slow fading and additional i.i.d. Gaussian noise. It is therefore assumed that the channel will remain constant within a block that contains M channel uses (transmit symbols). For simplicity reasons, and because this paper's purpose is the comparison of two different waterfilling approaches, it

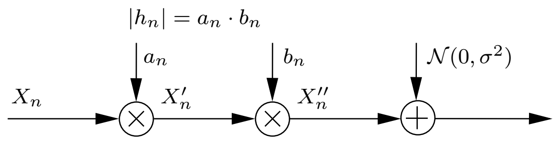

Fig. 1. Channel model used for simulations; depicted is one of two independent, statistically identical parallel transmit channels ("I" or "Q"-path).

is furthermore assumed that the classical Gaussian channel capacity is a valid measure for the performance of the schemes. This approach is acceptable since, as will be shown later, the differences in power allocation between the two schemes are so small that the channel capacity is a valid comparison metric, even for a small number of bits transmitted per channel use. Figure 1 illustrates the generation of channel coefficients by taking into account both slow fading (a_i), and block fading (b_i). This particular approach was chosen to test the behavior of the proposed algorithm in the presence of block fading and slowly changing average channel quality at the same time. In each block i the receive power $P_r(i)$ will equal the transmit power $P(i)$ scaled by the magnitude square $|h(i)|^2$ of the channel coefficient $h(i)$, i.e.,

$$P_r(i) = |h(i)|^2 \cdot P(i) \tag{1}$$

Coherent detection - i.e. full channel knowledge at the receiver - is assumed.

III. The Waterfilling Algorithms

Most commonly used waterfilling algorithms are directly calculating the transmit powers for the corresponding channel coefficients, without calculating the waterlevel λ at all [3]. This, in turn, leads to the necessity to rerun the algorithm on a sufficiently long ensemble of recent channel coefficients every time a transmit decision shall be made. Therefore, in practice, the conventional waterfilling algorithm has to be executed in every block if there is information to be transmitted, which is always assumed. The number N of channel coefficients, on which the waterfilling-algorithms are run (i.e. the recent history of the channel) is a crucial measure. If N is large, the solution is very similar to the theoretical (but non-causal) optimum solution of running the algorithm on the total lifetime of the channel, i.e. both slow and fast fading is exploited - the power assignment is near optimum within a long time-span. If N is short, on the other hand, the computational cost is greatly reduced, but only the fast fading can be exploited and the power assigned to each channel coefficient will deviate significantly from the theoretical (but non-causal) solution, especially if the channel suffers adverse slow fading conditions or temporarily sporadic availability of a potential communications partner. In a finite-lifetime channel scenario, a large N is generally preferred: by concatenating previously experienced channel profiles, a waterlevel can thus be found which solves the difficult problem of finding the optimum channel coefficients to use for an energy-efficient transmission. In what follows, the previously-developed [2] fixed-point algorithm and an efficient non-iterative algorithm are briefly discussed.

A. Fixed-Point Calculation of the Waterlevel

In [1], [2], we have presented an algorithm for finding the optimum waterlevel λ through fixed-point iterations according to

$$\varphi(\lambda) = \frac{J(\lambda)}{\frac{E_0/M}{2\sigma^2} + \sum_{j=1}^{J(\lambda)} \frac{1}{|\hat{h}(j)|^2}} \,. \tag{2}$$

This fixed-point function is subsequently called $\varphi(\lambda)$. In the fixed-point, $\varphi(\lambda) = \lambda$. In (2), the function $J(\lambda)$ in the numerator returns the number of channel-power coefficients which are larger than the waterlevel λ:

$$J(\lambda) = \max_{j:|\hat{h}(j)|^2 > \lambda} j \,, \tag{3}$$

where the channel coefficients are sorted according to magnitude in descending order. Furthermore, M is the number of channel uses in each block, and E_0 denotes the total amount of energy that can be spent on the whole sequence of channel coefficients that waterfilling is applied on. Although not an iteration-less algorithm, [1] has found convergence within at most 9 iterations in the case of $N = 1000$. The algorithm is shown to always converge (i.e., there is only one fixed-point), in [1]. The following implementation similar to Matlab-Syntax illustrates the approach (boldface characters denote vectors):

```
**Initialise:**
$|\mathbf{h}|^2_{sorted}$ = sort($|\mathbf{h}|^2$,'descend');
$\lambda$ = 0.3; $\lambda_{old}$ = 0.5; $\frac{E_0/M}{2\sigma^2}$ = 10;

**Fixed-Point Iterations:**
**while** $|\lambda - \lambda_{old}|$ > 0.00001
    $\lambda_{old}$ = $\lambda$
    J = length(find(($|\mathbf{h}|^2_{sorted} - \lambda_{old}$)>0));
    $\lambda$ = J/($\frac{E_0/M}{2\sigma^2}$ + sum(1./$|\mathbf{h}|^2_{sorted}$(1:J)));
**end**

**Normalize Result:**
$\lambda$ = $\frac{\lambda}{2\sigma^2}$
```

Please note that the fraction $\frac{E_0/M}{2\sigma^2}$ is an arbitrarily chosen simulation parameter that does not affect the algorithmic performance. The fixed-point algorithm can easily be implemented recursively (see [1]), however this does not produce any performance advantage in Matlab.

B. Direct Calculation of the Transmit Powers

An alternative, non-iterating (in the sense that the result does not have to be refined after running the algorithm once) algorithm, presented in [3], is used for efficiently and precisely calculating the transmit powers associated to every channel coefficient:

$$C_P = M \sum_{n=1}^{N} \log_2 \Big(1 + \frac{|h(n)|^2 \cdot P(n)}{2\sigma^2}\Big) . \qquad (4)$$

The limitation of the total energy according to

$$\sum_{n=1}^{N} P(n) = E_0 , \qquad (5)$$

defines a constrained optimization problem. The corresponding functional $L\left(P(n), \lambda\right)$ with the Lagrange multiplier $\lambda > 0$ is:

$$L \doteq M \sum_{n=1}^{N} \log_2 \Big(1+\frac{|h(n)|^2 \cdot P(n)}{2\sigma^2}\Big)+\lambda\big(E_0 - M\sum_{n=1}^{N} P(n)\big) . \qquad (6)$$

Therefore, the waterfilling solution determining the transmit powers for the individual channel coefficients reads

$$P^*(n) = \Big(\frac{1}{\lambda \log(2)} - \frac{2\sigma^2}{|h(n)|^2}\Big)^+ . \qquad (7)$$

which is a "waterfilling" solution (e.g. [4]). Equation (7) introduced a "max"-operation according to $(x)^+ \doteq \max(0, x)$ to ensure that the solutions for the powers do not take negative values; the Karush-Kuhn-Tucker [5] conditions guarantee that (7) is still an optimal solution to the problem. Together with (5), the following statement is therefore true:

$$E_0 = \frac{J}{\lambda \log(2)} - \sum_{n=1}^{J} \frac{2\sigma^2}{|h(n)|^2} , \qquad (8)$$

where it is important to note that only those channel-coefficients are taken into account that are contributing a transmission power $P^*(n)$ in (7) that is greater than zero. Reformulating (8), an expression for the waterlevel can be found:

$$\lambda = \frac{1}{\log(2)} \frac{J}{E_0 + \sum_{n=1}^{J} \frac{2\sigma^2}{|h(n)|^2}} . \qquad (9)$$

Therefore, the transmit powers are, according to (7),

$$P^*(n) = \frac{E_0 + \sum_{\nu=1}^{J} \frac{2\sigma^2}{|h(\nu)|^2}}{J} - \frac{2\sigma^2}{|h(n)|^2} . \qquad (10)$$

Please note the substitution of $n \to \nu$ in order to avoid naming conflicts.

This algorithm, however, is not practical in this specific form, since, as stated above, there will be some channel coefficients contributing positive, and some will be contributing negative transmit power. There is a simple solution: In a waterfilling solution, and assuming that at least one channel coefficient is greater than zero, there will always be energy assigned to at least one - the biggest - channel coefficient. Furthermore, there will be a smallest channel coefficient that will be used for transmission, and all the coefficients that are still smaller will be contributing "negative transmit Power" according to (10). Therefore, the channel coefficients (or channel SNRs, if the noise power is not constant) will be sorted from large to small, and the stopping criterion for the algorithm is reached as soon as (10) is negative. The following schematic illustrates the implementation of the algorithm. Again, boldface characters denote vectors.

```
Initialise:
$|\mathbf{h}|^2_{sorted}$ = sort($|\mathbf{h}|^2$,'descend');
CSNR = 1./($|\mathbf{h}|^2_{sorted}$./$2\sigma^2$);
k = 1;
SUM = CSNR(k);

Determine Stopping Criterion and Calculate SUM
while ($E_0$+SUM+CSNR(k+1))/k - CSNR(k+1) > 0
     SUM = SUM + CSNR(k+1);
   k=k+1;
end

Calculate Transmit Powers:
for i = 1:k
P(i) = ($E_0$+SUM)/k - CSNR(i);
end
```

C. Comparison of the Algorithms

A comparison of the algorithms shows some fundamental differences. Above all, the fixed-point algorithm is of the iterative kind with a notably fast convergence. We have shown in [1] that the algorithm converges within at most 9 iterations for 1000 channel coefficients. The second algorithm is non-iterative, does therefore not require refinement of the result and does not provide the waterlevel λ in its original form, whereas the fixed-point algorithm calculates *only* λ, which has to be used to calculate the actual transmit power as in (7), if required.

The following Figure (2) illustrates the runtime differences between the two algorithms: the average runtime in Matlab (as an average of 1000 runs) is plotted over the number of channel coefficients. In Matlab, the relative performance of the two algorithms changes as the number of channel coefficients increases: for a large number of channel coefficients, the ratio of the runtimes decreases roughly exponentially (3). In spite of the fixed-point algorithm being less efficient, the next section will introduce a new strategy for reduced average computation time by re-calculating the waterlevel λ *when it is required.*

IV. EWMA-Triggered Waterfilling

In section II we have introduced the channel model as one which takes into account two different types of fading: slow fading and block fading. Such a channel behavior may occur along the highway in vehicular communications scenarios, or, physically very similarly, for ad-hoc-communications of personal devices which may exchange data (special case: act as a relay) in a pedestrian scenario in a crowded place (such as a shopping mall). Here, it is an unsolved problem how to decide if a wireless link which is detected is good enough to establish a connection, and, if that is the case,

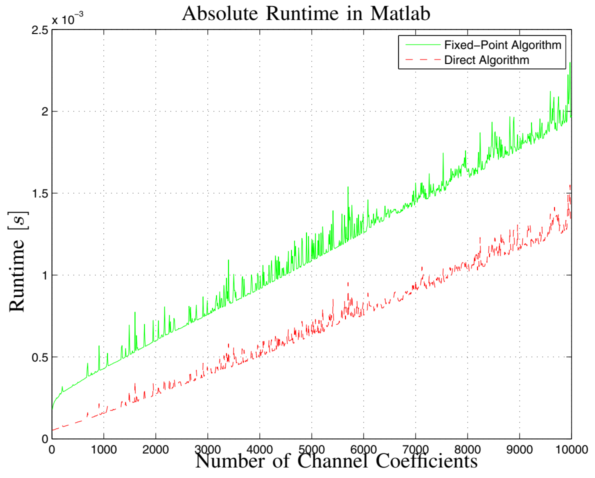

Fig. 2. Comparison of Algorithm Runtime in Matlab for Direct Calculation vs. Fixed-Point Algorithms. Lines smoothed by taking the average of 1000 algorithm runs for each data-point.

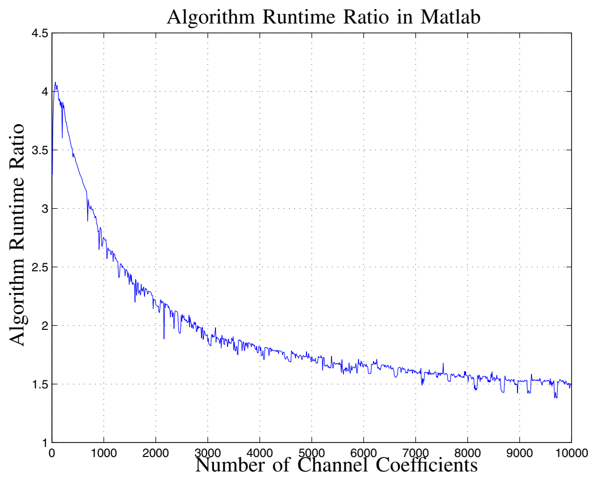

Fig. 3. Algorithm Runtime Ratio in Matlab for Direct Calculation vs. Fixed-Point Algorithms. The Fixed-Point Algorithm is slower. Lines smoothed by taking the average of 1000 algorithm runs for each data-point.

how to optimally exploit the channel and at the same time save resources and minimize interference with others. In [2], we have proposed an algorithm intended for vehicular communications that tries to forecast the channel quality and adapt the waterlevel accordingly. However, this approach, while efficient, does not solve the problem of the fundamental decision, if a connection shall be established at all.

This paper therefore suggests an alternative approach: if the waterlevel has been calculated based on the most recent N channel coefficients, and assuming that this time-span is long enough to contain, due to physical proximity of communication partners, a few "good" channel states, it will automatically provide a decision threshold when to establish a connection or to let the opportunity pass. In order to reduce accumulation of data (channel states), it is possible to sub-sample the channel state and consider only e.g. each 10^{th} channel coefficient for the calculation of the waterlevel. Traditional waterfilling approaches typically calculate the transmit power in every current transmit block. For comparison purposes, it is assumed that they do that for the same most recent N channel coefficients. This will, however, produce a huge processing demand. This paper therefore suggests the following approach which is *not* supported by algorithms that directly calculate the transmit power in the current block.

```
while 1
    Transmit Decision:
      if $P^*(n) = \left(\frac{1}{\lambda \log(2)} - \frac{2\sigma^2}{|h(n)|^2}\right) > 0$
         transmit with power $P^*(n)$;
      end

    Update EWMA Filter:
      EWMA = EWMA$\cdot\left(1 - \frac{1}{N}\right)$ + $|h_n|^2\cdot$($\frac{1}{N}$);

    Check for Slow Fading:
      if abs(EWMA - EWMA_old) > $\Delta\cdot$EWMA_old
         re-calculate $\lambda$;
         EWMA_old = EWMA;
      end
end
```

There are two important parameters in this algorithm:

- The relative change Δ of the EWMA filter output (EWMA) compared to the reference EWMA_old that triggers the re-calculation of the waterlevel. If the allowed change Δ is too small, the performance of the algorithm will be equal to an algorithm directly calculating the transmit power in each block, therefore consuming more resources. If Δ is too large, the algorithm will not (or slowly) adapt to a change of the average link quality. This paper's simulations have been done with thresholds of $\Delta = 0.05$ and $\Delta = 0.1$, with excellent results and a slight performance gain for smaller Δ.
- The number of recent channel-coefficients N, which is also the forgetting factor of the EWMA-filter. If this value is too small, the algorithm does not recognize good connection opportunities, and it increases the computational load since the moving average will change a lot. If N is too large, the re-calculation of the waterlevel will take a long time and might not have finished when another re-calculation is already required.

V. Simulation Results

The system model implemented for simulation purposes was the above mentioned Rayleigh block fading model with additive i.i.d. Gaussian noise. The block duration and therefore the minimum coherence time of the channel was $1ms$. Each block consists of $M = 100$ channel uses (symbols) with an average available transmit energy of $0.1Ws$ per symbol. The Rayleigh fading (indicated by b_n in Fig. 1) scale parameter was chosen to be $B = 1/\sqrt{2}$ so that the block fading does not change the mean power of the channel. Additionally, we considered a slowly changing channel by introducing a multiplicative sinusoidal process (a_n in Fig. 1) with a frequency of $1Hz$: $a_n = \sin(2 \cdot \pi \cdot 0.001 \cdot n) \cdot 0.1 + 0.4$. Although the initial ordering of the channel coefficients is irrelevant to the waterfilling result, the slow fading process is needed to examine the results for waterfilling memory $N \neq 1000$.

Table I summarizes results that have been obtained with the above simulation parameters and 10^7 channel coefficients. While the direct calculation of transmit powers is superior

TABLE I

Update Threshold: 5%

	Direct Algorithm	EWMA-Triggered
	$\mathbf{N = 10}$	
Transmitted Bits	$6.826 \cdot 10^7$	$3.853 \cdot 10^7$
Energy used	$2.294 \cdot 10^6$	$2.046 \cdot 10^6$
RunTime	$4.855 \cdot 10^2$	$9.708 \cdot 10^2$
Idle Time	1.58	
	$\mathbf{N = 10^2}$	
Transmitted Bits	$4.487 \cdot 10^7$	$4.521 \cdot 10^7$
Energy used	$1.049 \cdot 10^6$	$1.065 \cdot 10^6$
RunTime	$5.467 \cdot 10^2$	$1.603 \cdot 10^2$
Idle Time	24.75	
	$\mathbf{N = 10^3}$	
Transmitted Bits	$4.744 \cdot 10^7$	$4.733 \cdot 10^7$
Energy used	$1.000 \cdot 10^6$	$9.971 \cdot 10^5$
RunTime	$1.463 \cdot 10^3$	$1.634 \cdot 10^2$
Idle Time	219.81	
	$\mathbf{N = 10^4}$	
Transmitted Bits	$4.746 \cdot 10^7$	$4.695 \cdot 10^7$
Energy used	$1.000 \cdot 10^6$	$9.856 \cdot 10^5$
RunTime	$1.473 \cdot 10^4$	$7.551 \cdot 10^2$
Idle Time	$7.752 \cdot 10^4$	

to the EWMA-triggered approach with $N = 10$, the latter algorithm is superior with respect to runtime in all the other cases. Interestingly the EWMA-triggered approach suffers only very little in terms of throughput, and this seems to hold for all values of N. The gain with respect to *run time* is, on the other hand, clearly visible with growing N: with $N = 10^2$, the *run time* is already reduced by 71%, $N = 10^3$ brings a reduction of 89% and for $N = 10^4$, the reduction in processing time is even 95%. This trend is also visible if the *idle time* is observed: this is the average number of channel coefficients between two updates of the waterlevel. Based on these observations, it seems practical to consider values of N in the range of 10^3. Of course, this depends mainly on the application and hardware-restrictions.

As mentioned earlier, the optimum solution, though not causal and therefore impossible to implement, is another important benchmark. It was evaluated in the same simulation by calculating the waterfilling solution for all 10^7 channel coefficients at once, and resulted in a total amount of $6.85 \cdot 10^7$ transmitted bits. This value is 31% better than the result we obtained with EWMA-triggered waterfilling for $N = 10^3$. While the performance of the proposed algorithm seems to be not very close to this optimum solution, we want to stress that the performance compared to the best *causal* solution is still extremely high, if not virtually identical for large N.

A. Implementation Complexity

The question of practicability in terms of implementational complexity and demand for resources is still unanswered. Based on the simulation results for $N = 10^3$, and considering a block duration of $1ms$, the following estimations can be made: If every 10^{th} channel coefficient is sampled and stored into a cyclic buffer of length $N = 10^3$, the waterfilling algorithm has to be run approximately every 2.2 seconds. If all the channel coefficients are stored with a resolution of 16Bits, this will cause a memory requirement of 2kB, which seems absolutely reasonable. In the case of an EWMA-triggered update of the waterlevel, the processor would copy the whole cyclic buffer to another memory and perform the waterfilling on this set of channel coefficients. There is no need to finish the calculation within $1ms$ - the assumed minimum channel coherence time - the transmit decisions can be made on the old waterlevel, until the new one is available.

VI. Conclusion

This paper has introduced a new strategy for optimal use of the time-varying wireless channel in high-mobility, ad-hoc and limited lifetime communication scenarios. Maximizing the amount of data transmitted in an energy-efficient way calls for the use of a waterfilling approach. The application of a waterfilling-algorithm also provides a decision for determining "good opportunities for transmission", based on the recently experienced history. However, classic waterfilling algorithms, though efficient and robust, demand the calculation of the currently optimum transmit power in every time-slot while considering the recent past. The discussed alternative approach, on the other hand, calculates the waterlevel as a decision threshold and updates it only if the output of an exponentially-weighted moving-average (EWMA) filter changes more than a certain percentage compared to a recently stored reference value. This ensures that time- and energy-consuming calculations are performed only if absolutely necessary, while maintaining a performance virtually identical to the traditional approach. Finally, an implementation is briefly discussed to show that the new approach does not require the use of too many resources. We therefore believe that the proposed approach provides a valuable strategy by enabling small communications devices to perform complex decisions on channel use while maintaining small resource requirements.

References

[1] J. Gonter and N. Görtz, "An algorithm for highly efficient waterfilling with guaranteed convergence," in *Systems, Signals and Image Processing (IWSSIP), 2012 19th International Conference on*, 2012, pp. 126–129.

[2] N. Görtz and J. Gonter, "Information transmission over a finite-lifetime channel under energy-constraints," in *Proceedings European Wireless 2011*, Vienna, April 2011.

[3] P. Palomar and J. R. Fonollosa, "Practical algorithms for a family of waterfilling solutions," *IEEE Transactions on Signal Processing*, vol. 53, no. 2, pp. 686–695, Feb. 2005.

[4] D. Tse and P. Viswanath, *Fundamentals of Wireless Communications*, Cambridge University Press, 2005.

[5] H. W. Kuhn and A. W. Tucker, "Nonlinear programming," in *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability (available online at http://projecteuclid.org)*, Berkeley, CA, USA, July/August 1950, pp. 481–492.