Distributed Power Control Algorithms for Wireless Networks

Cynara Wu and Dimitri P. Bertsekas, *Fellow, IEEE*

***Abstract*—Power control has been shown to be an effective way to increase capacity in wireless systems. In previous work on power control, it has been assumed that power levels can be assigned from a continuous range. In practice, however, power levels are assigned from a discrete set. In this work, we consider the minimization of the total power transmitted over given discrete sets of available power levels subject to maintaining an acceptable signal quality for each mobile. We have developed distributed iterative algorithms for solving a more general version of this integer programming problem, which is of independent interest, and have shown that they find the optimal solution in a finite number of iterations which is polynomial in the number of power levels and the number of mobiles.**

***Index Terms*—Cellular networks, distributed algorithms, integer programming, power control.**

I. Introduction

EFFICIENT resource utilization is a primary problem in cellular communications systems. Resource issues include assigning transmit power levels to users subject to acceptable signal quality, providing varying levels of service to different priority classes, and maintaining connections in the presence of user movements. Given a set of users that wish to be connected, transmit power levels must be assigned. We propose a distributed algorithm that determines if there is a power assignment providing an acceptable signal quality for each user and if so, provides a solution that minimizes the total transmitted power. Minimizing energy consumption is important when users have limited battery power.

A great deal of work has been done on power control. Algorithms have been developed and shown to minimize the number of channels required to accommodate every user [3], to maximize the minimum signal-to-noise ratio (SNR) with a constraint on the total transmitted power [2], and to minimize the total transmitted power [5], [7]. However, these algorithms all assume that power can be allocated from a continuous range. Our work does not make such assumptions and restricts assigned power levels to be from given discrete sets, more accurately reflecting actual systems. In addition, while our algorithms can be used to assign power levels in order to minimize the total transmitted power subject to obtaining acceptable SNRs for each mobile, our formulation is more general and can be applied to other linear and nonlinear integer programming problems, which are of independent interest. Furthermore, we show that our algorithms admit an on-line and distributed implementation, allowing the addition and deletion of constraints which correspond to arrivals and departures of users.

We first formally define in Section II the problem which we are addressing. Then, in Section III, we generalize the problem to an integer programming problem with certain constraints. In Section IV, we develop an iterative algorithm for solving this integer programming problem and show that the algorithm solves the problem in a finite number of iterations. The algorithm is simple to implement, can be implemented in a distributed environment, and requires computation which is polynomial in the number of variables and the cardinality of the discrete sets. In Section V, we discuss variations of the iterative algorithm involving the addition and deletion of constraints. In Section VI, we discuss distributed versions of the iterative algorithm. In Section VII, we describe some computational results.

II. Problem Formulation

We consider a system of N cells in which M mobiles are to establish a connection. Each cell contains a single base station. Depending on the distance between a mobile and a base station as well as path loss, fading, and shadowing, the power received at base station s that is transmitted by mobile i is attenuated by a gain g_{is}. Mobile i can communicate with a base station provided its SNR is above some given threshold T_i. We are given a finite set X_i of discrete power levels from which to assign to mobile i. The goal is to determine whether there exists an assignment of power levels and base stations to all mobiles so that each mobile's SNR is acceptable, and if so, find an assignment that minimizes the total transmitted power.

Let w_i denote the power transmitted by mobile i and s_i denote the base station to which mobile i is assigned. The SNR of mobile i at base station s_i is then

$$\mathrm{SNR}(i,s_i)=\frac{w_i g_{is_i}}{\sum_{k\neq i} w_k g_{ks_i}+\nu_{s_i}}$$

where ν_{s_i} is the receiver noise at the base station. The power control problem we wish to solve has the form

$$\begin{aligned}&\min \sum_{i=1}^{M} w_i && (P_c)\\ &\text{subject to } \frac{w_i g_{is_i}}{\sum_{k\neq i} w_k g_{ks_i}+\nu_{s_i}} \geq T_i, \quad i=1,\ldots,M,\\ &w_i \in X_i, \qquad i=1,\ldots,M,\\ &s_i \in \{1,2,\ldots,N\}, \quad i=1,\ldots,M.\end{aligned} \tag{1}$$

Manuscript received September 30, 1999; revised April 14, 2000. This work was supported by NSF under Grant NCR-9 622 636.

C. Wu is withMalachite Technologies, Inc., Methuen, MA 01844 USA.

D. Bertsekas is with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology; Cambridge, MA 02135 USA (e-mail: dimitrib@mit.edu).

Publisher Item Identifier S 0018-9545(01)03938-X.

Therefore, if $\{(w_i^*, s_i^*) \mid i = 1, \ldots, M\}$ is an optimal solution, the ith mobile should be assigned to base station s_i^* at power level w_i^*. Equivalently, the problem can be written as

$$\min \sum_{i=1}^{M} w_i$$
subject to
$$\max\left\{ \frac{w_i g_{i1}}{\sum_{k \neq 1} w_k g_{k1} + \nu_1}, \ldots, \frac{w_i g_{iN}}{\sum_{k \neq N} w_k g_{kN} + \nu_N} \right\} \geq T_i, \quad i = 1, \ldots, M; \quad w_i \in X_i.$$

The relaxed version of problem (P_c), in which the power levels at which the mobiles are transmitting are selected from continuous intervals, can be solved by an iterative algorithm described by Yates and Huang [7]. At each iteration of their algorithm, each mobile adjusts its power to the minimum power necessary to obtain the threshold SNR at some base station under the assumption that all other users maintain their previous power levels. They show that synchronous and asynchronous versions of the algorithm converge to optimal solutions.

In the next section, we describe an integer programming generalization of problem (P_c) and provide an iterative algorithm to solve it. This algorithm is similar to that of Yates and Huang for the relaxed version of the power control problem.

III. Integer Programming Generalization

In this section, we provide an integer programming generalization of the power control problem, (P_c), described above. We consider a cost minimization problem with the following form:

$$\begin{aligned} &\min f(x) && (P)\\ &\text{subject to } h_i(x) \geq b_i, \quad i = 1, \ldots, M\\ &x_i \in X_i, \quad i = 1, \ldots, M \end{aligned}$$

where

$x = (x_1, \ldots, x_M)$	optimization vector;
b_i	real numbers;
f and h	functions mapping vectors in $\Re^M$ to real numbers;
sets X_i	given finite sets of real numbers.

In the above problem statement and in what follows, all vectors are viewed as column vectors. The problem is illustrated by Fig. 1 for the case where

$$\begin{aligned} h_1(x) &= x_1 - \ln(x_2 + 1),\\ h_2(x) &= -\ln(x_1 + 1) + x_2,\\ b_1 &= b_2 = 1 \end{aligned}$$

and X_i is the set of nonnegative integers less than some arbitrary constant for $i = 1, 2$.

It can be seen that the power control problem is a special case of problem (P). This is illustrated in the following example for the case of just two mobiles and two base stations.

Example 1: Consider the power control problem in which we have two mobiles and two base stations. The power levels are constrained to be integers less than or equal to some positive integer C. The problem can be formulated as a problem of type (P) as follows:

$$\min w_1 + w_2$$
subject to
$$\max\left\{ \frac{g_{11}}{\nu_1} w_1 - \frac{T_1 g_{21}}{\nu_1} w_2, \frac{g_{12}}{\nu_2} w_1 - \frac{T_1 g_{22}}{\nu_2} w_2 \right\} \geq T_1,$$
$$\max\left\{ \frac{-T_2 g_{11}}{\nu_1} p_1 + \frac{g_{21}}{\nu_1} p_2, \frac{-T_2 g_{12}}{\nu_2} p_1 + \frac{g_{22}}{\nu_2} p_2 \right\} \geq T_2,$$
$$w_i \in \{0, 1, \ldots, C\}, \quad i = 1, 2.$$

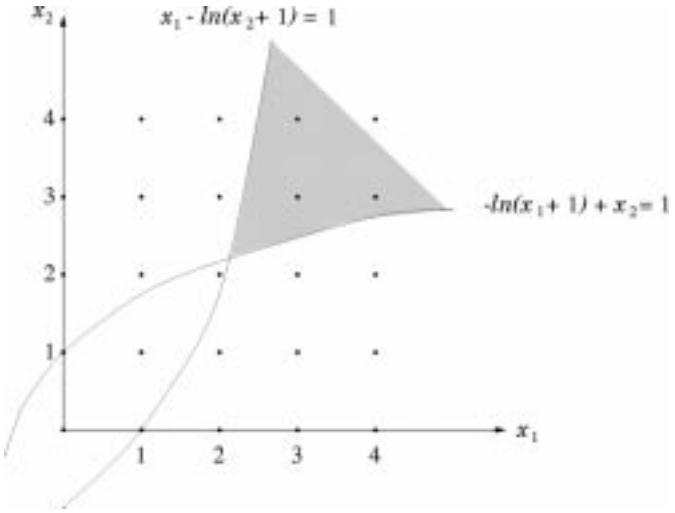

Fig. 1. Illustration of problem (P). The region specified by the constraints $h_i(x) \geq b_i$ is shaded. The feasible region consists of the dots/points that lie within the shaded region.

The problem is illustrated in Fig. 2 for the case where

$$g_{11} = 4/5, \quad g_{21} = 4/5, \quad T_1 = T_2 = 3/4,$$
$$g_{12} = 1/4, \quad g_{22} = 2/3, \quad \text{and} \quad \nu_1 = \nu_2 = 1.$$

The feasible region for each of the constraints are indicated in parts (a) and (b), and the intersection is indicated in part (c). The optimal solution is $w^* = (3, 2)$. Note that the feasible region need not be a convex polyhedron.

Let X denote the Cartesian product of $X_1, \ldots, X_M$:

$$X = X_1 \times \cdots \times X_M.$$

For any $x = (x_1, \ldots, x_M) \in X$, $i \in \{1, \ldots, M\}$, and $z \in X_i$, we make the following assumptions regarding (P):

Assumption 1: If $z > x_i$, then

$$h_i(x + (z - x_i)e_i) > h_i(x),$$

where e_i is a unit vector in $\Re^M$ with a 1 in the ith position and 0's elsewhere. Furthermore, we have

$$h_j(x + (z - x_i)e_i) \leq h_j(x), \quad \text{for } j \neq i.$$

Assumption 2: f is monotonically nondecreasing in x; i.e.,

$$\text{if } z > x_i, \quad \text{then } f((x + (z - x_i)e_i) \geq f(x).$$

Assumption 1 states that increasing the ith component of x increases $h_i(x)$ and either decreases or does not affect $h_j(x)$

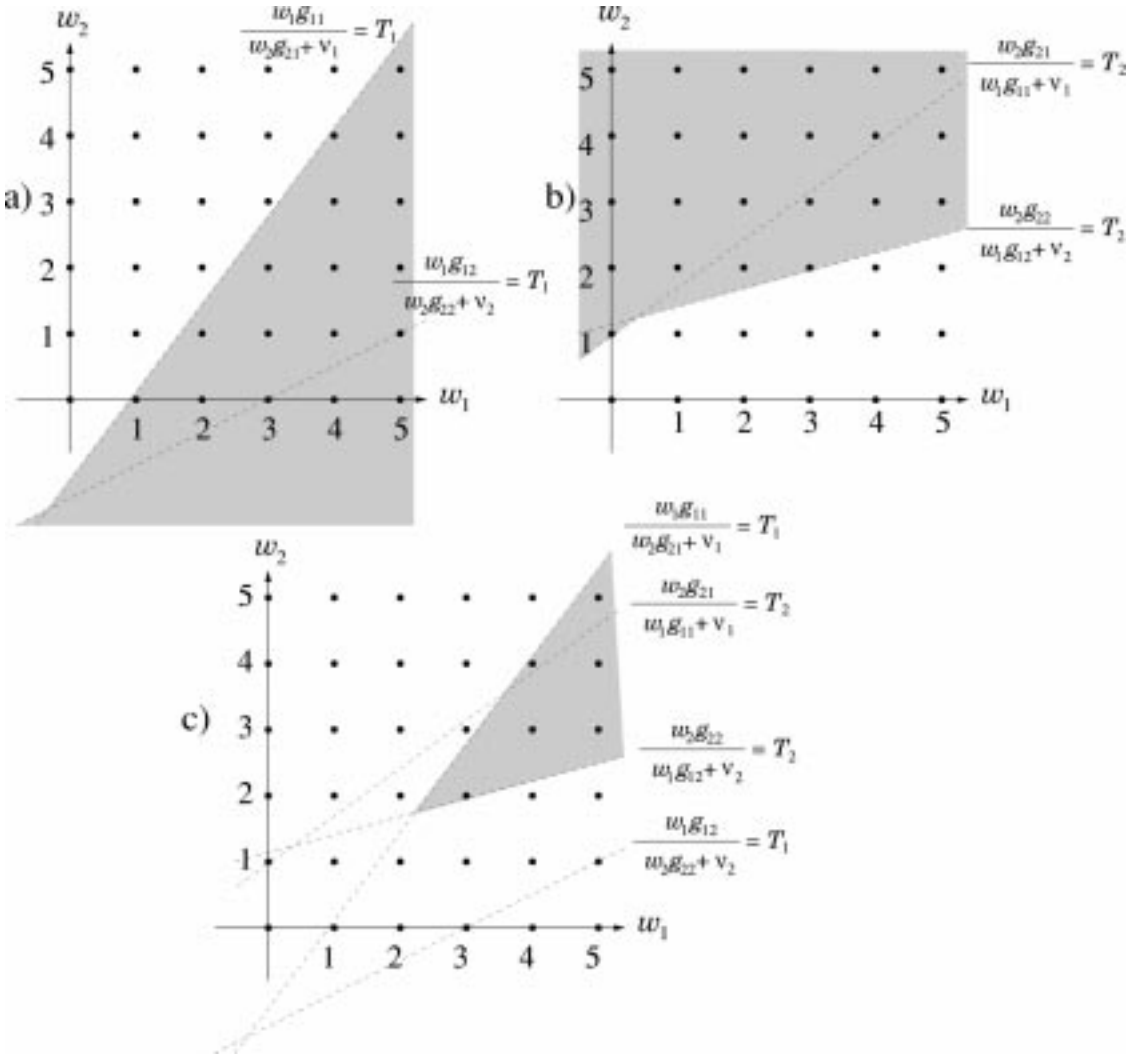

Fig. 2. Illustration of the power control problem for Example 1. The shaded regions of parts (a) and (b) represent the region satisfying the first and second constraints. The shaded region of part (c) represents the region satisfying both constraints.

for $j \neq i$. Note that from Assumption 1, we have $z < x_i \Rightarrow h_i(x + (z - x_i)e_i) < h_i(x)$ so that

$$h_i(x + (z - x_i)e_i) > h_i(x) \Longrightarrow z > x_i. \tag{2}$$

In the case where the constraint functions h_i are linear, Assumption 1 is satisfied if the corresponding constraint matrix has positive elements along the diagonal and nonpositive elements off the diagonal.

As was seen in Example 1, the power control problem (P_c) is a special case of (P). It is straightforward to show that it satisfies Assumptions 1 and 2.

In general, we can view (P) as a problem involving the allocation of discrete resources to M users. The resource allocation is represented by the vector x, where x_i is the quantity of resource allocated to user i. Each user i requires that some objective, or constraint, is met, i.e., $h_i(x) \geq b_i$. Assumption 1 essentially specifies that the effects of resources allocated to other users impede or have no effect on a user's ability to satisfy its constraint. Assumption 2 specifies that resources have nonnegative incremental costs. The goal is to minimize some function of the resources being allocated subject to satisfying each user's constraint. This integer programming problem and the algorithms described in this paper should be useful in contexts more general than power control.

IV. Algorithms for Solving Problem (P)

In this section, we describe an algorithm that will determine an optimal solution of problem (P) if one exists. If the problem has no feasible solution, the algorithm will determine that none exists. Note that if the problem is feasible, there is an optimal solution since there can be only a finite number of feasible points.

We first define some additional notation

$$\begin{aligned} &x(t) = \text{value of } x \text{ after the } t\text{th iteration of} \\ &\qquad \text{an algorithm.} \\ &x(t) \leq x(t') \text{ means } x_i(t) \leq x_i(t'), \quad i = 1, \ldots, M. \\ &x(t) < x(t') \text{ means } x_i(t) \leq x_i(t'), \quad i = 1, \ldots, M, \\ &\qquad \text{where } x_i(t) < x_i(t'), \text{ for some } i. \end{aligned}$$

We assume that we start with a point $x(0) = (x_1(0), \ldots, x_M(0))$ such that if (P) has a feasible solution, $x(0)$ satisfies $x(0) \leq x^*$, where $x^* = (x_1^*, \ldots, x_M^*)$ is some optimal

solution. (An optimal solution is guaranteed to exist when (P) has a feasible solution since the sets X_i are assumed finite.) One possibility is to set $x_i(0)$ to the minimum value in X_i

$$x_i(0) = \min\{z \in X_i\}, \quad i = 1, \ldots, M.$$

If problem (P) has no feasible solution, there is no restriction on $x(0)$. Note that if $x(0)$ is feasible, then since $x(0) \leq x^*$ and Assumption 2 holds, $x(0)$ must be optimal.

We define the set V_i, for $i = 1, \ldots, M$, as follows:

$$V_i = \{x \in X \,|\, h_i(x) \geq b_i\}.$$

Essentially, x is an element of V_i if it satisfies the ith constraint. For the problem in Example 1, the shaded regions in parts (a) and (b) of Fig. 2 represent V_1 and V_2, respectively. For $i = 1, \ldots, M$, we also define the following set of scalars associated with a point $x \in X$:

$$S_i(x) = \{z \in X_i \,|\, h_i(x + (z - x_i)e_i) \geq b_i\}.$$

Essentially, $S_i(x)$ is the set of values in X_i such that setting the ith component of x to any value in the set, while leaving the remaining components of x unchanged, results in the updated value of x satisfying the ith constraint. In other words, for $\bar{i} \in \{1, \ldots, M\}$, the scalar z is an element of $S_{\bar{i}}(x)$ if the vector $\hat{x}$ given by

$$\hat{x}_i = \begin{cases} z & i = \bar{i} \\ x_i & i \neq \bar{i}, \end{cases} \quad i = 1, \ldots, M,$$

is an element of V_i. Again referring to the problem in Example 1, for any point $w = (w_1, w_2)$, $S_1(w)$ is the set of values i such that (i, w_2) is in the shaded region of Fig. 2, part (a), and $S_2(w)$ is the set of values i such that (w_1, i) is in the shaded region of Fig. 2, part (b).

Note that by definition of the sets V_i and $S_i(x)$, if x is not an element of V_i for any $i = 1, \ldots, M$, then $S_i(x)$ cannot contain any values less than or equal to x_i due to Assumption 1. Furthermore, if $x_i^{\min}$ is the minimum value of $S_i(x)$, then $S_i(x)$ is the set of values in X_i that are greater than or equal to $x_i^{\min}$

$$S_i(x) = \left\{z \in X_i \,|\, z \geq x_i^{\min}\right\}, \quad i = 1, \ldots, M.$$

A. Typical Iteration of the Minimum Feasible Value Assignment (MFVA)

Given a point $x(t) = (x_l(t), \ldots, x_M(t))$ such that $x(t) \in X$, select some index $\bar{i} \in \{1, \ldots, M\}$ satisfying $x(t) \notin V_{\bar{i}}$.

- If no such $\bar{i}$ exists, the algorithm terminates and returns the point $x(t)$. (It will be shown that in this case, $x(t)$ is optimal.)
- If such an $\bar{i}$ does exist, then if the set $S_{\bar{i}}(x(t))$ is empty, the algorithm terminates (it will be shown that in this case, the problem is infeasible). Otherwise, set $x_{\bar{i}}(t+1)$ to the smallest value in the set $S_{\bar{i}}(x(t))$; i.e., set $x_{\bar{i}}(t+1)$ to the smallest value in $X_{\bar{i}}$ such that $x(t+1) \in V_{\bar{i}}$. For $i \neq \bar{i}$, we set $x_i(t+1) = x_i(t)$:

$$x_i(t+1) = \begin{cases} \min S_{\bar{i}}(x(t)) & \text{for } i = \bar{i} \\ x_i(t) & \text{for } i \neq \bar{i}, \end{cases} \quad i = 1, \ldots, M. \tag{3}$$

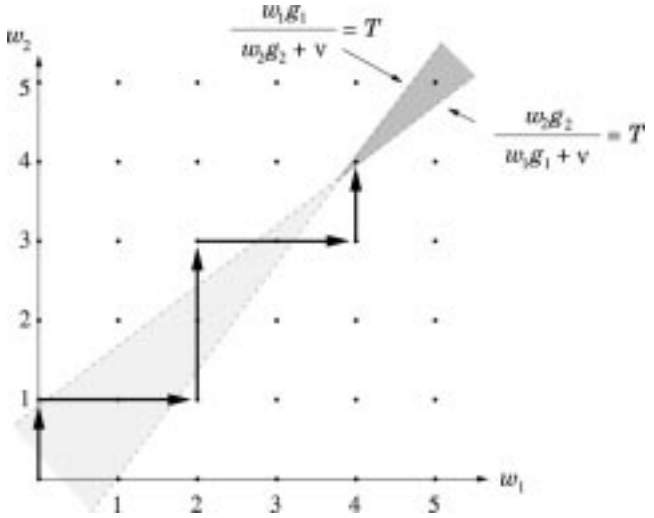

Fig. 3. Illustration of the MFVA algorithm for the problem of Example 2.

The MFVA algorithm starts with an initial point $x(0)$ and continuously applies the iteration described by (3) until a termination condition is reached. It is illustrated by the example that follows.

Example 2: Consider the following power control problem:

$$\begin{aligned} &\min w_1 + w_2 \\ &\text{subject to } \frac{\frac{4}{5}w_1}{\frac{4}{5}w_2 + 1} \geq \frac{3}{4}, \quad \frac{\frac{4}{5}w_2}{\frac{4}{5}w_1 + 1} \geq \frac{3}{4}, \\ &\qquad w_i \in \{0, 1, \ldots, C\}, \quad i = 1, 2. \end{aligned}$$

As noted previously, the problem satisfies Assumptions 1 and 2. The results of applying the MFVA algorithm to this problem is illustrated in Fig. 3. The darkly shaded region represents the feasible region in which both constraints are satisfied. The lightly shaded region represents the region in which neither constraint is satisfied. At each iteration of the algorithm, each component remains less than or equal to that of the optimal solution, and a different constraint is satisfied, possibly causing a previously satisfied constraint to no longer be satisfied. The sequence of points generated by the algorithm therefore alternates between the regions where exactly one of the constraints is satisfied until a point is reached in which both of the constraints are satisfied.

To prove that the algorithm terminates, we first show that each application of iteration (3) to a vector $x(t)$ results in a vector $x(t+1)$ such that one component is strictly greater than the corresponding component of $x(t)$, i.e., that $x_{\bar{i}}(t+1) > x_{\bar{i}}(t)$, for some $\bar{i} \in \{1, \ldots, M\}$, while the remaining components remain unchanged. The number of iterations involving updating any particular component x_i is then bounded by the number of values in the set X_i since once x_i is equal to the maximum value in X_i, the set $S_i(x)$ must be empty. The number of iterations is therefore bounded by the number of values in all of the sets of X_i.

Proposition 1: If $x(t) \notin V_{\bar{i}}$ for some $\bar{i} \in \{1, \ldots, M\}$ and $S_{\bar{i}}(x(t))$ is nonempty, then $x_i(t) < \min S_{\bar{i}}(x(t))$.

Proof: Since $x(t) \notin V_{\bar{i}}$, we have $x_{\bar{i}}(t) \notin S_{\bar{i}}(x(t))$. As a result, we obtain

$$h_{\bar{i}}(x(t)) < b_{\bar{i}}. \tag{4}$$

For any $z \in S_{\bar{i}}(x(t))$, we have

$$h_{\bar{i}}(x(t) + (z - x_{\bar{i}}(t))e_i) \geq b_{\bar{i}}. \tag{5}$$

Combining (4) and (5), we have

$$h_{\bar{i}}(x(t)) < h_{\bar{i}}((x(t) + (z - x_{\bar{i}}(t))e_i). \tag{6}$$

From Assumption 1, (6) and (2) imply that $x_{\bar{i}}(t) < z$. ■

As a result of Prop. 1, an iteration of the MFVA algorithm starting with a vector $x(t)$ yields a vector $x(t+1)$ of which the $\bar{i}$th component is strictly greater than that of $x(t)$. Therefore, the MFVA algorithm terminates after a finite number of iterations for any initial point $x(0)$. The algorithm either terminates when the current point $x(t)$ satisfies $x(t) \in V_i$ for $i = 1, \ldots, M$, or when the set $S_{\bar{i}}(x(t))$ is empty for some $\bar{i} \in \{1, \ldots, M\}$ such that $x(t) \notin V_{\bar{i}}$. The following proposition and corollary show that given an appropriate starting point $x(0)$, the algorithm will terminate with an optimal solution in the former situation when the problem is feasible and will terminate in the latter situation when the problem is infeasible. In what follows, x^* refers to some optimal solution whenever problem (P) is feasible. If problem (P) is not feasible, x^* refers to the point such that $x_i^* = \max\{z \in X_i\}$ for $i = 1, \ldots, M$.

Proposition 2: If problem (P) is feasible, and $x(t)$ satisfies $x(t) \leq x^*$ and $x(t) \notin V_{\bar{i}}$ for some $\bar{i} \in \{1, \ldots, M\}$, then the set $S_{\bar{i}}(x(t))$ is nonempty. Furthermore, $x_{\bar{i}}(t) < \min S_{\bar{i}}(x(t)) \leq x_{\bar{i}}^*$.

Proof: Define $\hat{x}$ as follows:

$$\hat{x}_i = \begin{cases} x_i(t), & \text{for } i \neq \bar{i}, \\ x_i^*, & \text{for } i = \bar{i}. \end{cases}$$

Since x^* is an optimal solution, we have

$$h_{\bar{i}}(x^*) \geq b_{\bar{i}}.$$

Since $\hat{x}_{\bar{i}} = x_{\bar{i}}^*$ and $\hat{x}_i \leq x_i^*$, for $i \neq \bar{i}$, we have from Assumption 1

$$h_{\bar{i}}(\hat{x}) \geq h_{\bar{i}}(x^*).$$

Therefore, we have $h_{\bar{i}}(\hat{x}) \geq b_{\bar{i}}$, and it follows that $x_{\bar{i}}^*$ is an element of $S_{\bar{i}}(x(t))$. Therefore, $S_{\bar{i}}(x(t))$ is nonempty. The minimum of this set must be less than or equal to $x_{\bar{i}}^*$, and it follows from Prop. 1 that $x_{\bar{i}}(t) < \min S_{\bar{i}}(x) \leq x_{\bar{i}}^*$. ■

It follows from Prop. 2 that if an iteration of the MFVA algorithm is executed, we have $x(t) < x(t+1) \leq x^*$.

Corollary 3: Given an initial starting point $x(0) \leq x^*$, the MFVA algorithm terminates in a finite number of iterations under one of the following two conditions: either we have $x(t) \in V_i$ for $i = 1, \ldots, M$, in which case $x(t)$ is an optimal solution to (P), or the set $S_{\bar{i}}(x(t))$ is empty for some $\bar{i} \in \{1, \ldots, M\}$ such that $x(t) \notin V_{\bar{i}}$, in which case there is no feasible solution to (P).

Proof: We have shown that the MFVA algorithm terminates and that when this occurs, either the current point $x(t)$ satisfies $x(t) \in V_i$ for $i = 1, \ldots, M$, or the set $S_{\bar{i}}(x(t))$ is empty for some $\bar{i} \in \{1, \ldots, M\}$ such that $x(t) \notin V_{\bar{i}}$. Consider the former situation, and let $x(t)$ be the resulting vector. Then we have $x(t) \in V_i$ for $i = 1, \ldots, M$, and therefore $x(t)$ is feasible. By induction using the results from Prop. 2, we have $x(t) \leq x^*$, so $x(t)$ must be optimal. Consider next the latter situation. If $S_{\bar{i}}(x(t))$ is empty for any $\bar{i}$ such that $x(t) \notin V_{\bar{i}}$, then from Prop. 2, problem (P) is infeasible. ■

The main idea of the algorithm is to increase some component $x_i(t)$ of $x(t)$ at each iteration while keeping $x(t) \leq x^*$. Each component x_i can be adjusted at most $|X_i|$ times, where $|\cdot|$ denotes the cardinality of a set. There are therefore at most $\sum_{i=1}^{M} |X_i|$ or $\max_{i=1,\ldots,M} |X_i| M$ iterations, by which point either the optimal solution has been found, or it is determined that the problem is infeasible.

As noted earlier, if during a particular iteration of the MFVA algorithm, $\bar{i}$ is some index satisfying $x(t) \notin V_{\bar{i}}$ and the set $S_{\bar{i}}(x(t))$ is not empty, then $S_{\bar{i}}(x(t))$ consists of points greater than or equal to $x_{\bar{i}}(t)$. We can construct a variant of the MFVA algorithm in which instead of setting $x_{\bar{i}}(t+1)$ to the smallest value in $S_{\bar{i}}(x(t))$, we simply increase $x_{\bar{i}}(t+1)$ to the next higher value in the set $X_{\bar{i}}$. It is straightforward to show that this single-step variant of the MFVA algorithm also finds the optimal solution of (P) if one exists in a finite number of iterations, and that it has the same complexity bound on the number of required iterations as the standard MFVA algorithm. This variant may be useful in contexts where it is difficult to determine the set $S_{\bar{i}}(x(t))$. For instance, in the power control problem, it may be possible to only determine whether a mobile user has a sufficient SNR and not to determine what power level is necessary to satisfy the signal to noise threshold if the current ratio is not sufficient. In this case, the power level of the mobile user can be incrementally increased until its threshold ratio is reached. Computational experiments show that the number of iterations required for the variant to find the optimal solution is typically less than 5% greater than the number required for the standard MFVA algorithm to find the optimal solution, so the potential convenience afforded by the variant described may be well worth the extra computation involved.

V. Modifying the Constraints of Problem (P)

In a cellular network, the number of users that need to be connected to the base station varies as users arrive and depart. The optimal power assignments change as a result of the arrivals and departures. In this section, we consider how the optimal solution to (P) changes when the problem is modified as a result of increasing or decreasing the dimension of the optimization vector x, as well as adding or removing a constraint. In the context of the power control problem, this corresponds to the arrival or departure of a mobile.

Given an optimal solution x^* to problem (P), we will show that an optimal solution to a new problem $(\hat{P})$ that augments problem (P) with an additional constraint can be found by using x^* as a starting point for the MFVA algorithm. Given an optimal solution x^* to problem (P), we will show that an optimal solution to a new problem $(\bar{P})$ that removes a constraint from problem (P) can be found by using x^* as a starting point for an

"inverse" algorithm that finds an appropriate starting point for the MFVA algorithm.

B. Addition of a Constraint

Assume problem (P) is feasible and we have an optimal solution. Suppose we are given a new problem $(\hat{P})$ that differs from the original in that the dimension of the optimization vector is increased by one, and it has one more constraint. Problem $(\hat{P})$ has the form

$$\begin{aligned}&\min \hat{f}(x)\\ &\text{subject to } \hat{h}_i(x) \geq b_i, \quad i=1,\ldots,M+1,\\ &x_i \in X_i, \quad i=1,\ldots,M+1\end{aligned}$$

where the optimization variable $x=(x_1,\ldots,x_{M+1})$ is now a vector in $\Re^{M+1}$, the b_i, for $i=1,\ldots,M$, are as before, and b_{M+1} is a given real number. The sets X_i, for $i=1,\ldots,M$, are as before, and X_{M+1} is a given finite set of nonnegative real numbers. The functions $\hat{f}$ and $\hat{h}_i$, for $i=1,\ldots,M+1$ map vectors in $\Re^{M+1}$ to real numbers. Furthermore, we assume that if $x_{M+1}=0$, we have

$$\hat{f}(x_1,\ldots,x_{M+1}) = f(x_1,\ldots,x_M),$$

and

$$\hat{h}_i(x_1,\ldots,x_{M+1}) = h_i(x_1,\ldots,x_M), \quad i=1,\ldots,M.$$

We assume that Assumption 1 holds for the new problem, while Assumption 2 is modified as follows:

Assumption 2': $\hat{f}$ is monotonically increasing; i.e.,

$$\text{if } z > x_i, \quad \text{then } \hat{f}((x+(z-x_i)e_i) > \hat{f}(x).$$

Using an optimal solution to the original problem as a partial starting point, the MFVA algorithm can be used to solve the new problem, as shown in the next proposition.

Proposition 4: Let x^* be an optimal solution of (P). Let $\hat{x}^*$ be an optimal solution of $(\hat{P})$ if $(\hat{P})$ is feasible, and let $\hat{x}^*$ be such that $\hat{x}_i^* = \max\{z \in X_i\}$ for $i=1,\ldots,M+1$, if $(\hat{P})$ is not feasible. Given the starting point $\hat{x}(0) = (\hat{x}_1(0),\ldots,\hat{x}_{M+1}(0))$, where $\hat{x}_i(0)$ satisfies $\hat{x}_i(0) \leq x_i^*$ for $i=1,\ldots,M$ and $\hat{x}_{M+1}(0)$ satisfies $0 \leq \hat{x}_{M+1}(0) \leq \hat{x}_{M+1}^*$, the MFVA algorithm terminates in a finite number of iterations under one of the following two conditions: either we have $\hat{x}(t) \in V_i$ for $i=1,\ldots,M+1$, in which case $\hat{x}(t)$ is an optimal solution to $(\hat{P})$, or the set $S_{\overline{i}}(\hat{x}(t))$ is empty for some $\overline{i} \in \{1,\ldots,M+1\}$ such that $\hat{x}(t) \notin V_{\overline{i}}$, in which case there is no feasible solution to $(\hat{P})$.

Proof: Due to Corollary 3, it is sufficient to show that if $(\hat{P})$ is feasible, $x_i^* \leq \hat{x}_i^*$, for $i=1,\ldots,M$. Assume the contrary, that for some $\overline{i} \in \{1,\ldots,M\}$, $x_{\overline{i}}^* > \hat{x}_{\overline{i}}^*$. Define $\tilde{x}$ so that

$$\tilde{x}_i = \min(x_i^*, \hat{x}_i^*), \quad i=1,\ldots,M.$$

For any $i=1,\ldots,M$, if $\tilde{x}_i = x_i^*$, we have

$$h_i(\tilde{x}) \geq h_i(x^*) \geq b_i$$

where the first inequality follows from Assumption 1 and the second inequality follows since x^* is feasible for problem (P). For any $i=1,\ldots,M$, if $\tilde{x}_i = \hat{x}_i^*$, we have

$$\begin{aligned}h_i(\tilde{x}) &\geq h_i(\hat{x}_1^*,\ldots,\hat{x}_M^*) = \hat{h}_i(\hat{x}_1^*,\ldots\hat{x}_M^*,0)\\ &\geq \hat{h}_i(\hat{x}_1^*,\ldots\hat{x}_M^*,\hat{x}_{M+1}(0))\\ &\geq \hat{h}_i(\hat{x}^*) \geq b_i.\end{aligned}$$

The first three inequalities follow from Assumption 1 and the fact that we have $0 \leq \hat{x}_{M+1}(0) \leq \hat{x}^*$. The last inequality follows since $\hat{x}^*$ is feasible for problem $(\hat{P})$. $\tilde{x}$ is therefore a feasible solution to problem (P). Since for some $\overline{i} \in \{1,\ldots,M\}$, we have $x_{\overline{i}}^* > \hat{x}_{\overline{i}}^*$, it follows that $f(\tilde{x}) < f(x^*)$, and $\tilde{x}$ is a better solution than x^* to problem (P), yielding a contradiction. ■

To insure that $x_{M+1}(0) \leq \hat{x}_{M+1}^*$, we can initialize $x_{M+1}(0)$ as follows:

$$x_{M+1}(0) = \min\{z \in X_{M+1}\}.$$

C. Removal of a Constraint

Assume problem $(\hat{P})$ is feasible and we have an optimal solution. We are given a new problem (P) that differs from the original in that it has one fewer variable and one fewer set of constraints. As in Section V-A, if $x_{M+1}=0$, we have

$$\hat{f}(x_1,\ldots,x_{M+1}) = f(x_1,\ldots,x_M),$$

and

$$\hat{h}_i(x_1,\ldots,x_{M+1}) = h_i(x_1,\ldots,x_M), \quad i=1,\ldots,M.$$

We also use the same assumptions as in Section V-A.

Let $\hat{x}^*$ be some optimal solution to $(\hat{P})$. Since $\hat{x}^*$ is feasible for $(\hat{P})$, we have

$$\hat{h}_i(\hat{x}^*) \geq b_i, \quad i=1,\ldots,M+1.$$

Due to Assumption 1, we also have

$$\begin{aligned}h_i(\hat{x}_1^*,\ldots,\hat{x}_M^*) &= \hat{h}_i(\hat{x}_1^*,\ldots,\hat{x}_M^*,0)\\ &\geq \hat{h}_i(\hat{x}^*) \geq b_i, \quad i=1,\ldots,M.\end{aligned}$$

Therefore, $\hat{x}^*$ is feasible for (P), and (P) has an optimal solution. In order to find an optimal solution to problem (P), we can start with a solution $x(0)$ initialized as follows:

$$x_i(0) = \min\{z \in X_i\}, \quad i=1,\ldots,M$$

and run the MFVA algorithm to solve problem (P). Another possibility, which aims at a better starting point $x(0)$, is to initialize $x(0)$ as

$$x_i(0) = \hat{x}_i^*, \quad i=1,\ldots,M$$

and run an "inverse" version of the MFVA algorithm. Recall that during an iteration of the MFVA algorithm, some component of x is increased to the smallest value in its feasible set so that its corresponding constraint is satisfied. Consider an "inverse" algorithm in which an iteration consists of some component of x being decreased to the smallest value in its feasible set so that

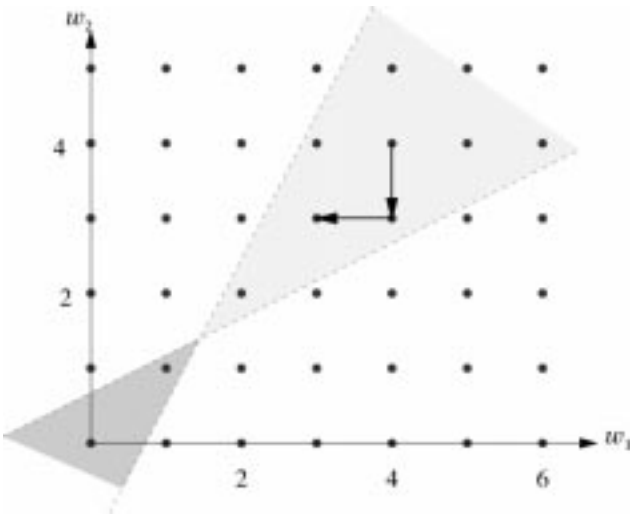

Fig. 4. Illustration of an "inverse" MFVA algorithm that does not correctly find the optimal solution in the case where a constraint is removed. The algorithm iteratively decreases some component so that its corresponding constraint is still satisfied. Starting at the point $(4,4)$, the algorithm terminates at the point $(3,3)$ even though the optimal solution is the point $(2,2)$.

its corresponding constraint is still satisfied. As illustrated in Fig. 4, such an algorithm may not find an optimal solution.

Instead, we consider an alternative "inverse" version of the MFVA algorithm that obtains a point $\underline{x} = (\underline{x}_1, \ldots, \underline{x}_M)$ such that $\underline{x} \leq x^*$, and then run the MFVA algorithm starting with $\underline{x}$. This "inverse" algorithm is described below. We first define for $i = 1, \ldots, M$, the set U_i as follows:

$$U_i = \{x \in X \,|\, h_i(x) \leq b_i,\}.$$

For $i = 1, \ldots, M$, we also define the following set of scalars associated with a point $x \in X$:

$$R_i(x) = \{z \in X_i \,|\, h_i(x_1, \ldots, x_{i-1}, z, x_{i+1}, \ldots, x_M) \leq b_i,\}.$$

Note that the sets U_i and $R_i(x)$ are analogous to the sets V_i and $S_i(x)$ defined in Section IV. In these sets, the inequalities are reversed. Also note that by definition of the two sets, if x is not an element of $U_{\bar{i}}$ for any $\bar{i} = 1, \ldots, M$, then $R_{\bar{i}}(x)$ can not contain any values greater than or equal to $x_{\bar{i}}$ due to Assumption 1. Furthermore, if $x_i^{\max}$ is the maximum value of $R_i(x)$, then $R_i(x)$ is the set of values in X_i that are less than or equal to $x_i^{\max}$

$$R_i(x) = \{z \in X_i \,|\, z \leq x_i^{\max}\}, \quad i = 1, \ldots, M.$$

C. Typical Iteration of the Maximum Infeasible Value Assignment (MIVA)

Given a point $x(t) = (x_1(t), \ldots, x_M(t))$ such that $x(t) \in X$, select some index $\bar{i} \in \{1, \ldots, M\}$ satisfying $x(t) \notin U_{\bar{i}}$.

- If no such $\bar{i}$ exists, the algorithm terminates and returns the point $x(t)$. (It will be shown that in this case, $x(t)$ is optimal.)
- If such an $\bar{i}$ does exist, then if the set $R_{\bar{i}}(x(t))$ is empty, the algorithm terminates (it will be shown that in this case, the problem is infeasible). Otherwise, set $x_{\bar{i}}(t+1)$ to the largest value in $X_{\bar{i}}$ such that $x(t+1) \in U_{\bar{i}}$; i.e., set $x_{\bar{i}}(t+1)$ to the largest value in the set $R_{\bar{i}}(x(t))$. For $i \neq \bar{i}$, we set $x_i(t+1) = x_i(t)$:

$$x_i(t+1) = \begin{cases} \max R_{\bar{i}}(x(t)), & \text{for } i = \bar{i} \\ x_i(t), & \text{for } i \neq \bar{i}, \quad i = 1, \ldots, M. \end{cases} \tag{7}$$

The Maximum Infeasible Value Assignment algorithm (MIVA) starts with an initial point $x(0)$ and continuously applies the iteration described by (7) until a termination condition is reached. It is illustrated by the example below.

Example 3: Consider the following power control problem in which we have three mobile users in a single cell with a threshold requirement of 0.4635:

$$\begin{aligned} &\min w_1 + w_2 + w_3 \\ &\text{subject to } \frac{\frac{4}{5}w_1}{\frac{4}{5}w_2 + \frac{5}{6}w_3 + 1} \geq .4635 \\ &\frac{\frac{4}{5}w_2}{\frac{4}{5}w_1 + \frac{5}{6}w_3 + 1} \geq .4635 \\ &\frac{\frac{5}{6}w_3}{\frac{4}{5}w_1 + \frac{4}{5}w_2 + 1} \geq .4635 \\ &\qquad w_i \in \{0, 1, \ldots, C\}, \quad i = 1, 2, 3. \end{aligned}$$

Using the MFVA algorithm, we can obtain the optimal solution $w^* = (11, 11, 11)$.

Suppose the second mobile user ends its connection, resulting in the following problem

$$\begin{aligned} &\min w_1 + w_3 \\ &\text{subject to } \frac{\frac{4}{5}w_1}{\frac{5}{6}w_3 + 1} \geq .4635 \\ &\frac{\frac{5}{6}w_3}{\frac{4}{5}w_1 + 1} \geq .4635 \\ &\qquad w_i \in \{0, 1, \ldots\}, \quad i = 1, 3. \end{aligned}$$

Using the resulting point from the original problem, $w = (11, 11)$, as the starting point, running the MIVA algorithm results in the point $\hat{w} = (1, 1)$. The progression of the algorithm is illustrated in Fig. 5. Note that $\hat{w}_i \leq w_i^*$, for $i = 1, 2$, where $w^* = (2, 2)$ is the optimal solution for this problem (P), and therefore $\hat{w}$ would be an appropriate starting point for the MFVA algorithm.

The proof of the algorithm terminating in a finite number of iterations is analogous to that of the MFVA algorithm. As in that case, each application of the iteration to a vector $x(t)$ results in a vector $x(t+1)$ such that one component is strictly less than the corresponding component of $x(t)$, i.e., that $x_{\bar{i}}(t+1) < x_{\bar{i}}(t)$, for some $\bar{i} \in \{1, \ldots, M\}$, while the remaining components remain unchanged. The number of iterations involving any particular component x_i is then bounded by the number of values in the set X_i since once x_i is equal to the minimum value in X_i, the set $R_i(x)$ must be empty. The number of iterations is therefore bounded by the number of values in all of the sets of X_i.

Consider the following problem:

$$\begin{aligned} &\max f(x) \quad (Q) \\ &\text{subject to } h_i(x) \leq b_i, \quad i = 1, \ldots, M \\ &x_i \in X_i, \quad i = 1, \ldots, M. \end{aligned}$$

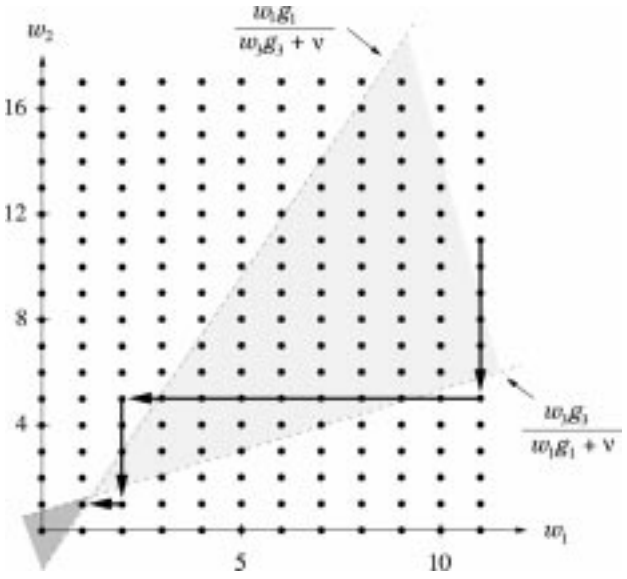

Fig. 5. Illustration of the MIVA algorithm.

As shown in the following proposition, given an appropriate starting point, the MIVA algorithm will terminate with an optimal solution to problem (Q) if (Q) is feasible. If (Q) is not feasible, the algorithm will terminate without a solution.

Proposition 5: Let x^* be an optimal solution to problem (Q) if (Q) is feasible. If (Q) is not feasible, let x^* be such that $x_i^* = \min\{z \in X_i\}$ for $i = 1, \ldots, M+1$. Given a starting point $x(0) \geq x^*$, the MIVA algorithm terminates in a finite number of iterations under one of the following conditions: either we have $x(t) \in U_i$ for $i = 1, \ldots, M$, in which case $x(t)$ is an optimal solution to (Q), or the set $R_{\bar{i}}(x(t))$ is empty for some $\bar{i} \in \{1, \ldots, M\}$ such that $x(t) \notin U_{\bar{i}}$, in which case there is no feasible solution to (Q).

Proof: The proof is analogous to those of Proposition 2 and Correlation 3 and is omitted. ∎

Returning to the problem of finding an optimal solution to (P) given an optimal solution to $(\hat{P})$, we propose to run the MIVA algorithm starting with the point $x(0)$ initialized as follows:

$$x_i(0) = \hat{x}_i^*, \quad i = 1, \ldots, M \tag{8}$$

where $\hat{x}^*$ is an optimal solution to $(\hat{P})$. If the algorithm terminates with a solution, we use the resulting point $\underline{x}$ as a starting point for the MFVA algorithm. If the algorithm terminates without a solution, we use the point $\underline{x}$, where $\underline{x}_i = \min\{z \in X_i\}$ as a starting point for the MFVA algorithm. For this method to yield an optimal solution to problem (P), we need the following condition.

D. Constraint Monotonicity Condition

Let $\underline{x} = (\underline{x}_1, \ldots, \underline{x}_M)$ be any point satisfying the constraints

$$h_i(\underline{x}) \leq b_i, \quad i = 1, \ldots, M \tag{9}$$

and let $\hat{x} = (\hat{x}_1, \ldots, \hat{x}_M)$ be any point satisfying the constraints

$$h_i(\hat{x}) \geq b_i, \quad i = 1, \ldots, M. \tag{10}$$

Then we have $\underline{x}_i \leq \hat{x}_i$, for $i = 1, \ldots, M$.

Under this condition, we can show that the proposed algorithm will find an optimal solution to problem (P). Note that the constraints provided by (9) and (10) form the feasible regions of (Q) and (P), respectively. In what follows, let $\underline{x}^*$ denote an optimal solution to problem (Q) if (Q) is feasible. If (Q) is not feasible, let $\underline{x}^*$ be the point such that

$$\underline{x}_i^* = \min\{z \in X_i\}, \quad i = 1, \ldots, M. \tag{11}$$

Proposition 6: Let the Constraint Monotonicity Condition hold and consider the following sequence of steps. Starting with a point $x(0)$ initialized according to (8), run the MIVA algorithm. If the algorithm finds an optimal solution to problem (Q), let $\underline{x}$ be this solution. Otherwise, if problem (Q) is infeasible, let $\underline{x}$ be the point given in (11). Run the MFVA algorithm starting with the point $\underline{x}$. The resulting vector is an optimal solution to problem (P).

Proof: We have already shown that the point $x(0)$ is feasible for problem (P). As a result of the Constraint Monotonicity Condition, $x(0)$ satisfies $x(0) \geq \underline{x}^*$. The MIVA algorithm will therefore find an optimal solution to (Q) if problem (Q) is feasible. Also as a result of the Constraint Monotonicity Condition, any solution $\underline{x}$ to (Q) satisfies $\underline{x} \leq x^*$, where x^* is any optimal solution to (P). If (Q) is not feasible, the point $\underline{x}$ given by (11) also satisfies $\underline{x} \leq x^*$. Since (P) has an optimal solution, it follows from Corollary 3 that running the MFVA algorithm starting with $\underline{x}$ will result in an optimal solution. ∎

It can be shown that the power control problem satisfies the Constraint Monotonicity Condition. If the Constraint Monotonicity Condition does not hold, the sequence of steps described in Prop. 6 is not guaranteed to find an optimal solution to problem (P) since any optimal solution $\underline{x}$ of problem (Q) may not satisfy $\underline{x}_i \leq x_i^*$, for $i = 1, \ldots, M$, where x^* is some optimal solution to problem (P). To guarantee finding an optimal solution to problem (P) we can start with a solution $x(0)$ initialized as follows:

$$x_i(0) = \min\{z \in X_i\}, \quad i = 1, \ldots, M$$

and run the MFVA algorithm to solve problem (P).

The following proposition provides an alternative condition under which the Constraint Monotonicity Condition holds.

Proposition 7: Suppose that

$$h_i(y) > h_i(x), \quad \text{for } i = 1, \ldots, M \tag{12}$$

for all vectors y of the form $y = (c + x_1, \ldots, c + x_M)$, where c is a positive constant. Then the Constraint Monotonicity Condition holds.

Proof: Assume the contrary, that for some $i \in \{1, \ldots, M\}$, we have $\underline{x}_i > \hat{x}_i$, where $\underline{x} = (\underline{x}_1, \ldots, \underline{x}_M)$ satisfies (9) and $\hat{x} = (\hat{x}_1, \ldots, \hat{x}_M)$ satisfies (10). Let i_m be the index such that $\underline{x}_i - \hat{x}_i$ is maximized:

$$i_m = \arg\max_i \{\underline{x}_i - \hat{x}_i\}.$$

Let y be a vector in $\Re^M$ such that $y_i = \underline{x}_{i_m} - \hat{x}_{i_m} + \hat{x}_i$. Since $\underline{x}_{i_m} - \hat{x}_{i_m} > 0$, we have from (12)

$$h_{i_m}(y) > h_{i_m}(\hat{x}) \geq b_{i_m}.$$

Note that $y_{i_m} = \underline{x}_{i_m}$. For $i \neq i_m$, we have

$$\underline{x}_{i_m} - \hat{x}_{i_m} \geq \underline{x}_i - \hat{x}_i$$

since i_m maximizes $\underline{x}_i - \hat{x}_i$. Therefore, we have

$$y_i \geq \underline{x}_i, \quad \text{for } i \neq i_m.$$

It follows that

$$h_{i_m}(\underline{x}) \geq h_{i_m}(y) > b_{i_m}$$

contradicting the assumption that $\underline{x}$ satisfies (9). ■

When the constraints h_i are linear, we can rewrite problem (P) as

$$\begin{aligned} &\min f(x) \\ &\text{subject to } Hx \geq b_i, \\ &x_i \in X_i, \quad i = 1, \ldots, M \end{aligned}$$

and the condition described by Prop. 7 corresponds to the $M \times M$ matrix H being diagonally dominant.

VI. Distributed Algorithms

In this section, we consider synchronous and asynchronous distributed versions of the MFVA algorithm. We assume that problem (P) is feasible and show that the algorithms will find an optimal solution to problem (P) in a finite number of iterations. If problem (P) is not feasible, the arguments below can be modified to show that in this case, similar to the sequential version, the distributed algorithms will terminate with an infeasible solution.

B. Synchronous MFVA

Given a point $x(t) = (x_1(t), \ldots, x_M(t))$ such that $x(t) \in X$, we assume we have M processors, each of which is responsible for updating a component of $x(t)$. Specifically, for $i = 1, \ldots, M$, we have

$$x_i(t+1) = d_i(x_1(t), \ldots, x_M(t))$$

where x_i is updated by processor i according to some given function d_i. We also assume the updates occur simultaneously, and each processor receives all of the updated values in time for the next iteration. The synchronous parallel version of the MFVA algorithm is given below and illustrated in Fig. 6.

Given a point $x(t) = (x_1(t), \ldots, x_M(t))$ such that $x(t) \in X$, each processor i such that $x(t) \notin V_i$, sets $x_i(t+1)$ to the smallest value in the set $S_i(x(t))$; i.e., $x_i(t+1)$ is the smallest value in X_i such that $x(t+1) \in V_i$. Each processor i such that $x(t) \in V_i$, sets $x_i(t+1) = x_i(t)$

$$x_i(t+1) = \begin{cases} \min S_i(x(t)), & \text{if } x(t) \notin V_i, \\ x_i(t), & \text{if } x(t) \in V_i. \end{cases} \tag{13}$$

The following proposition shows that given an appropriate starting point, this algorithm obtains an optimal solution to problem (P) in a finite number of iterations.

Proposition 8: Let x^* be some optimal solution to problem (P). Suppose the synchronous distributed version of the MFVA algorithm described by (13) is run with a starting point $x(0) \leq x^*$. Then for some finite $\bar{t}$, $x(t)$ is an optimal solution to (P) for all $t \geq \bar{t}$.

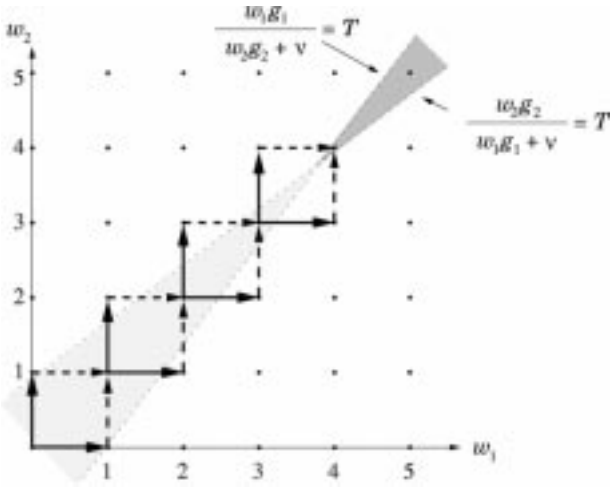

Fig. 6. Illustration of the synchronous parallel MFVA algorithm for the problem of Example 2.

Proof: From Proposition 2, we have that $x_i^* \geq x_i(t+1) > x_i(t)$ for all i such that $x(t) \notin V_i$. During each iteration for which we have $x(t) \notin V_i$ for at least some i, at least one component of x is strictly increased. Since the number of times each component of x can be increased is finite, the number of such iterations must also be finite. Therefore, for some finite $\bar{t}$, we must have $x(\bar{t}) \in V_i$ for $i = 1, \ldots, M$, so $x(\bar{t})$ must be feasible. By induction, we have $x(\bar{t}) \leq x^*$, so $x(\bar{t})$ must be optimal. According to the algorithm, we have $x(t+1) = x(t)$ for $t \geq \bar{t}$ since $x(\bar{t}) \in V_i$, for $i = 1, \ldots, M$. ■

Note that as currently stated, the algorithm continues indefinitely, even after an optimal solution has been obtained. One possible method for determining when an optimal solution has been obtained is for each processor to compare updated values of components with the previous values. When none of the components has changed, $x(t)$ is an optimal solution and processors no longer need to update their components.

C. Asynchronous MFVA

In describing the asynchronous distributed version, we use the framework presented in [1, Sect. 6.1]. Let $x(t) = (x_1(t), \ldots, x_M(t))$, where

$$x_i(t) = \text{Value of the } i\text{th component of } x \text{ at time } t.$$

We assume that there is a set $T = \{0, 1, 2, \ldots\}$ of times at which one or more components $x_i(t)$ of $x(t)$ are updated. Let T^i be the set of times at which $x_i(t)$ is updated. As in the synchronous case, we assume we have M processors, each of which is responsible for updating a component of $x(t)$. Here, however, we assume that the processors do not necessarily have access to the most recent values of the other components. Specifically, we assume that for any $i, j \in \{1, \ldots, M\}$, $\tau_j^i(t)$ is the time at which the value of the jth component that was most recently available at the processor updating $x_i(t)$ was last updated. Therefore, if each processor used the values of components most recently received, we would have

$$x_i(t+1) = d_i\left(x_1\left(\tau_1^i(t)\right), \ldots, x_M(\tau_M^i(t))\right), \quad \forall t \in T^i$$

where d_i are given functions and $\tau_j^i(t)$ are times such that

$$0 \leq \tau_j^i(t) \leq t, \quad \forall t \in T.$$

Note that a processor may not necessarily receive updated values of a component in the order that they were sent; i.e.,

$$\tau_j^i(t_1) < \tau_j^i(t_2), \quad \text{for } t_1 < t_2$$

may not necessarily hold.

The asynchronous version of the MFVA algorithm is given below.

For all $i = 1, \ldots, M$ and all $t \in T^i$, let

$$x^i(t) = \left(x_1^i(t), \ldots, x_M^i(t)\right)$$

be the vector of components being stored at time t by the processor updating $x_i(t)$. We have

$$x_i^i(t+1) = \begin{cases} \min S_i(x^i(t)), & \text{if } x^i(t) \notin V_i, \\ x_i^i(t), & \text{if } x^i(t) \in V_i, \end{cases} \tag{14}$$

where

$$x_j^i(t) = \max_{\bar{t} \in T^i, \bar{t} \leq t} x_j^i\left(\tau_j^i(\bar{t})\right), \quad j = 1, \ldots, M.$$

Note that the values used to update $x_i(t)$ are the maximum of the ones received for each component instead of the ones most recently received. As will be seen in Proposition 9, updates of any component results in a value that is greater than or equal to the previous value. Therefore, using the maximum of the values received results in using the most recently updated values.

The following proposition shows that given an appropriate starting point, this algorithm eventually obtains an optimal solution to problem (P).

Proposition 9: Let x^* be some optimal solution to problem (P). Suppose the asynchronous distributed version of the MFVA algorithm described by (14) is run with a starting point $x(0) \leq x^*$. Let the set $\tilde{T} \subset T$ be the set of times in which at least one component's value is changed. Then the set $\tilde{T}$ contains a finite number of elements. Furthermore, if $\bar{t}$ is the maximum element of $\tilde{T}$, then $x(\bar{t})$ is an optimal solution to (P).

Proof: From Proposition 2, we have that $x_i^* \geq x_i(t+1) > x_i(t)$ for all i such that $t \in T^i$ and $(x_1^i(t), \ldots, x_M^i(t)) \notin V_i$. If $t \notin T^i$ or $(x_1^i(t), \ldots, x_M^i(t)) \in V_i$, we have $x_i(t+1) = x_i(t)$. Since the number of times each component of x can be increased is finite, the set $\tilde{T}$ must contain a finite number of times. Let $\bar{t}$ be the maximum value in $\tilde{T}$. Since each update sent by a processor is eventually received by every other processor, we must have $(x_1^i(\bar{t}), \ldots, x_M^i(\bar{t})) \in V_i$ for $i = 1, \ldots, M$, and $x_j^i(\bar{t}) = x_j(\bar{t})$ for $i, j = 1, \ldots, M$. Therefore, $x(\bar{t})$ must be feasible. By induction, we have $x(\bar{t}) \leq x^*$, so $x(\bar{t})$ must be optimal. ∎

We do not discuss the issue of detecting when an optimal solution has been found. This issue is addressed in [1, Sect. 8.1], as well as in [6]. A related issue is how to construct a parallel synchronous or asynchronous implementation of the algorithm of Section V-B for the case where a constraint is removed. One possibility is to run a (synchronous or asynchronous) parallel version of the MIVA, detect its termination using one of the schemes of [1] and [6], and then run a (synchronous or asynchronous) parallel version of the MFVA.

TABLE I
NUMBER OF ITERATIONS REQUIRED TO FIND AN OPTIMAL SOLUTION USING THE PREVIOUS SOLUTION AS A STARTING POINT AND USING THE VECTOR $(0, \ldots, 0)$ AS A STARTING POINT

Number of Mobile Users	Number of iterations using previous solution as starting point	Number of iterations starting from 0
914	5626*	5626
915	46	5673
916	128	5803
917	26	5825
918	27	5847
919	79	5929
920	441	6372

* This value corresponds to a starting point of 0.

VII. COMPUTATIONAL RESULTS

We have implemented sequential versions of the MFVA and MIVA algorithms to obtain empirical results. We summarize the results as follows.

- As noted in Section IV, the number of iterations is bounded by the number of values in all of the sets of feasible points for each component. In fact, the maximum number of iterations is the number required for the single-step variant to terminate. If the problem is feasible, the maximum number is the number of values in all of the sets of feasible points for each component less than or equal to the corresponding component of some optimal solution. If the problem is infeasible, the maximum number is the number of values in all of the sets of feasible points for each component. Computational experiments show that the number of iterations required for the MFVA algorithm to terminate is typically greater than 95% of the maximum bound.
- When a new user enters the system, running the MFVA algorithm using as a starting point the optimal solution for the problem prior to the new user's arrival typically results in substantial computational savings. Table I shows results from a power control problem involving a system of ten by ten cells and approximately nine hundred mobile users. The number of iterations required to find the optimal solution for an initial problem is given, along with the number of iterations required to find the optimal solution when additional users enter the system.
- When an existing user departs from the system, running the MIVA algorithm to find an appropriate point from which to run the MFVA algorithm typically results in solutions in which each component is very close to the minimum value in its corresponding feasible set. It therefore seems more efficient to simply run the MFVA algorithm starting with a vector in which each component is initialized to the minimum value in its corresponding feasible set. Additional computational results should be conducted to determine whether there are situations in which the MIVA algorithm yields computational savings. In practice, one may want to use the simpler heuristic algorithm

mentioned first in Section V-B (cf. Fig. 4) whenever a user departs from the system, and perform a reoptimization periodically or when a new user arrives that cannot be accommodated by running the MFVA algorithm starting from the current operating point.

VIII. Summary

In this paper, we considered the problem of minimizing the total power transmitted subject to providing acceptable SNRs to all users when power levels are to be assigned from discrete sets. We generalized this problem to an integer programming problem with certain constraints. We presented an iterative algorithm for the problem as well as synchronous and asynchronous distributed versions, and showed that they solve the problem in a finite number of iterations. We also discussed extensions applicable to the power control problem involving situations where a new user enters the system or where an existing user exits the system.

References

[1] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Englewood Cliffs, NJ: Prentice-Hall, 1989; Belmont, MA: Athena Scientific, 1997.

[2] A. Grandhi, J. Zander, and R. Yates, "Wireless personal communications," , vol. 1, pp. 257–270, 1994–95.

[3] S. Papavassiliou and L. Tassiulas, "Improving the capacity in wireless networks through integrated channel base station and power assignment," *IEEE Trans. Veh. Technol.*, vol. 47, pp. 417–427, 1998.

[4] ——, "Joint optimal channel base station and power assignment for wireless access," *IEEE Trans. Network*, vol. 4, pp. 857–872, 1996.

[5] J. Rulnick and N. Bambos, "Mobile power management for wireless communication networks," *Wireless Networks*, vol. 3, pp. 3–14, 1997.

[6] S. A. Savari and D. P. Bertsekas, "Finite termination of asynchronous iterative algorithms," *Parallel Comput.*, vol. 22, pp. 39–56, 1996.

[7] R. Yates and C. Huang, "Integrated power control and base station assignment," *IEEE Trans. Veh. Technol.*, vol. 44, pp. 638–644, 1995.

Cynara Wu received the S.B., S.M., and Ph.D. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, in 1989, 1990, and 1999, respectively. Her doctoral research focused on dynamic resource allocation for cellular communications systems.

She is currently with Malachite Technologies, Methuen, MA, conducting research on network optimization.

Dimitri P. Bertsekas (S'70–SM'77–F'84) received the combined B.S.E.E. and B.S.M.E. degrees from the National Technical University of Athens, Greece, in 1965, the M.S.E.E. degree from George Washington University, Washington, DC, in 1969, and the Ph.D. degree in system science from the Massachusetts Institute of Technology, Cambridge, in 1971.

He has held faculty positions with the Engineering-Economic Systems Department, Stanford University (1971–1974) and the Electrical Engineering Department of the University of Illinois, Urbana (1974–1979). He is currently Professor of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology. He consults regularly with private industry and has held editorial positions in several journals. He has conducted research in the areas of estimation and control of stochastic systems, linear, nonlinear and dynamic programming, data communication networks, parallel and distributed computation, and neural networks, and has written numerous papers in each of these areas. He is the author or coauthor of 11 textbooks and research monographs, including *Data Networks* (Englewood Cliffs, NJ: Prentice-Hall, 1992, 2nd ed.), *Dynamic Programming and Optimal Control*, (2 volumes), (Athena Scientific, 1995), *Neuro-Dynamic Programming*, (Belmont, MA: Athena Scientific, 1996), *Network Optimization*, (Belmont, MA: Athena Scientific, 1998), and *Nonlinear Programming*, (Belmont, MA: Athena Scientific 1999, 2nd ed.).