4-17-2007

Resource Allocation for Multi-agent Problems in the Design of Future Communication Networks

Jorge Piovesan

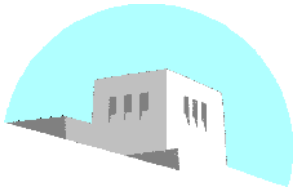
Chaouki Abdallah

Herbert Tanner

Henry Jerez

Joud Khoury

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING



SCHOOL OF ENGINEERING
UNIVERSITY OF NEW MEXICO

Resource Allocation for Multi-agent Problems in the Design of Future Communication Networks

Jorge Piovesan, Chaouki Abdallah, Herbert Tanner, Henry Jerez, and Joud Khoury [1][2]

UNM Technical Report: EECE-TR-07-001

Report Date: April 2, 2007

[1]Jorge Piovesan (corresponding author), Chaouki Abdallah, and Joud Khoury are with the Department of Electrical and Computer Engineering, of the University of New Mexico, Albuquerque NM 87131, (jlpiovesan,chaouki,jkhoury)@ece.unm.edu. Herbert Tanner is with the Department of Mechanical Engineering of the University of New Mexico, Albuquerque NM 87131, tanner@unm.edu. Henry Jerez is with the Corporation for National Research Initiatives, Reston VA 20191, hjerez@cnri.reston.va.us.

[2]The work of Piovesan, Abdallah, Jerez, and Khoury is partially supported by NSF award CNS 0626380 under the FIND initiative.

new architecture for a smarter Internet abstracts the functional components of the network from the hardware that enables it. This is done via software agents that implement such functions and are capable of relocating themselves over the network to optimize resources. To achieve that we need an algorithm that governs the way agents distribute themselves in the physical network, and this paper presents our first effort towards this goal. We formulate the problem as an optimization problem, where agents must be distributed in the network, and can receive resources from the nodes they occupy according to their task requirements. This optimization problem is then solved in a hierarchical manner: A centralized (randomized) algorithm optimizes the agent distribution among the nodes, and a decentralized (convex optimization) algorithm performs the resource allocation within each node. We present simulation results showing that the hierarchical optimization algorithm achieves the desired objective. Finally we discuss our next steps towards decentralizing the proposed algorithm.

1 Introduction

Retransmissions, delays, and communication failures may occur when a mobile device either moves across multiple networks, or has an intermittent connection inside the same network. Such problems are caused by the current implementation of the Internet, which which delivers packets to static locations. This assumption is no longer valid when the communication task involves mobile devices, since such devices may connect/disconnect without no prior notice, and may even move across different networks during the communication task [8].

A novel Internet architecture addressing these issues was proposed by Jerez et al. [8]. This architecture essentially postulates an abstract network that treats the nodes and the traffic as digital entities, separating the functional components of the network from the hardware that enables them. In this fashion, a logical (intelligent) network is created on top of the physical network, enabling functions like persistent identification of different objects and smarter routing to avoid retransmissions. The functional components of the network are conceptualized as software agents (e.g. routing agents, storage agents, DNS implementation agents, etc.), and the hardware is seen as a resource to be used by these agents, such that their tasks can be efficiently executed (Figure 1).

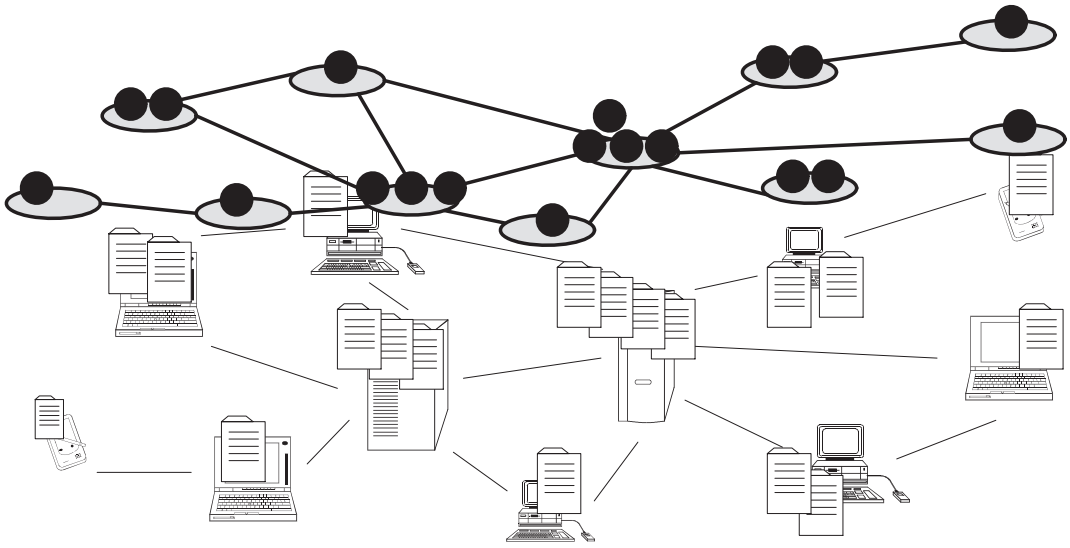


Figure 1: Network example: Each platform in the network can run several processes concurrently. The network is abstracted as a graph and the processes as agents (black circles) that can move among the nodes.

In this setting, each particular node in the physical network is capable of simultaneously hosting more than one agent. Moreover, agents can move around searching for nodes with more resources in order to complete their tasks in the best possible way. Agents are viewed as greedy entities competing with each other for the completion

of their tasks (benefit), by consuming hardware resources required for the task execution. The objective is to use the resources of the network so that the aggregate benefit of all the agents in the network is maximized.

The previous paragraph sets the stage for an optimization problem that, due to the nature of the network, must be solved in a decentralized form. Each node needs to distribute its resources among the agents that it hosts. Agents need to decide whether they must migrate to a different node if this action increases their benefit. These actions must be executed without a centralized authority controlling the whole network, meaning that the desired solution of this problem should be in the form of local and semi-local policies (based on information about the node currently occupied and its immediate neighbors).

Similar theoretical problems have been addressed in the study of different Internet Congestion Control protocols, within an optimization and dynamical systems theory framework [9, 13]. In [9] an optimization problem is proposed to model a transmission control protocol (TCP). In [1, 21] a system based on price and benefit interacting dynamics is proved to solve a congestion control problem in a decentralized manner. These and other related results are surveyed in [18]. Peer-to-peer networks have also been the subject of study from a dynamical systems perspective [13] obtaining characterizations of their behavior and approximate requirements for desirable performance.

It is important to note that an essential difference between the problem posed here, and prior literature on multi-agent control systems is that our problem considers agents moving between discrete locations on a graph; the latter body of literature focuses on agents moving in a continuous space [4, 5, 7, 10, 11, 15, 19].

In this paper we present an approach to solve the network resource allocation problem described above. We approach the problem from an optimization perspective, assuming initially that the topology of the network and the number of agents are fixed. We show that this problem can be solved hierarchically: On top we have the combinatorial optimization problem of the distribution of the agents among the nodes, and at the bottom the fully decentralized convex optimization problem of resource assignment among agents within each node. The solution to the convex optimization problem is obtained using classical techniques (Karush-Khun-Tucker conditions [2, 14]), while the agent distribution problem is solved using randomized algorithms to avoid computational complexity issues common to model-based techniques (e.g. combinatorial optimization [3], mixed integer programming [6], etc). We illustrate the problem and the solution using a numerical example. Finally we discuss future research steps to fully decentralize the algorithm and relax some of the current assumptions related to the topology of the network and the number of agents.

2 Problem Formulation

Consider a set of agents capable of calculating their own benefit as a function of the resources that they are using from the network. The network has nodes with different types of resources, capable of deciding how much of each resource is allocated to each agent within each node. The nodes are connected over a network of fixed topology, and agents can move between any two connected nodes.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph with nodes indexed by $\mathcal{V} = \{1, 2, ..., N_v\}$ and edges $\mathcal{E} = \{(v, w) : v, w \in \mathcal{V}, v \neq w,$ and v connected to $w\}$. We call the graph undirected if $(v, w) \in \mathcal{E}$ whenever $(w, v) \in \mathcal{E}$, and time-invariant if the set of nodes $\mathcal{V}$ and the set of edges $\mathcal{E}$ remain unchanged over time. A graph is connected if there is a path between any pair of nodes in the graph, where a path from v to w is a sequence of different nodes starting at v and ending at w such that consecutive nodes are connected.

Assumption 1 (Network)**.** *The network is a time-invariant undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where the total number of nodes is N_v.*

Assumption 2 (Resources)**.** *We assume there exist N_r types of resources in the network (e.g. bandwidth, memory, connectivity). For each node $i \in \mathcal{V}$ we have the set of fixed available amount of resources $R_i = \{r_{i,1}, r_{i,2}, ...r_{i,N_r}\}$, where $r_{i,j} \in \mathbb{R}$ is the amount of resource of type j available at node i. We assume that $0 \leq r_{i,j} < \infty$ for all $i \in \mathcal{V}$ and all $j \in \mathcal{R} = \{1, 2, ..., N_r\}$.*

Assumption 3 (Agents)**.** *There is a fixed number of agents N_a, indexed by the set $\mathcal{A} = \{1,2,...,N_a\}$. The state of each agent $k \in \mathcal{A}$ consists of an ordered tuple $(x_{k,1}, x_{k,2}, ..., x_{k,N_r}, v_k)$, where $x_{k,j} \in \mathbb{R}$ represents the amount of resource of type j allocated to agent k, and $v_k \in \mathcal{V}$ denotes the location of agent k in the graph. Note that $0 \leq x_{k,j} < \infty$ for all $k \in \mathcal{A}$ and all $j \in \mathcal{R}$.*

If Assumption 1 is relaxed then we seek a solution to the technical problem described in section 1 where the topology, and the number of nodes change over time. If Assumption 2 is relaxed, we can then allow for time varying resources. Note however that this is strongly related to the topology of the network, because as we show below, the unavailability of resources at a particular node is equivalent to the disappearance of that node from the network. Finally, if Assumption 3 is relaxed, we can allow variations in the number of agents over time.

The description of the network as an undirected graph in Assumption 1, the existence of a fixed number of type of (non-negative but finite) resources in Assumption 2, and the description of the agents states in Assumption 3 are reasonable. Communications in a network are usually bidirectional so that information flow in both directions may safely be assumed[1]. The devices enabling the network probably generate a large set of type of resources to be allocated, but this set is still finite. The agents benefit depends upon their location in the network and the resources allocated to them, so the relevant information for the agents is contained in the state description introduced in Assumption 3.

Assumption 4 (Utility functions)**.** *Each agent k has an expression of its utility function $U_k(\mathrm{x}_k) : \mathbb{R}^{N_r} \to \mathbb{R}$ where $\mathrm{x}_k = (x_{k,1}, x_{k,1}, ..., x_{k,N_r})^T$. The utility function is of the form:*

$$U_k(\mathrm{x}_k) = \sum_{j=1}^{N_r} u_{k,j}(x_{k,j}) \tag{1}$$

where $u_{k,j}(x_{k,j}) : \mathbb{R} \to \mathbb{R}$ is assumed to be a strictly concave, strictly increasing, and differentiable function of $x_{k,j}$ for all $k \in \mathcal{A}$ and all $j \in \mathcal{R}$. Moreover, we assume that $u_{k,j}(x_{k,j}) \to -\infty$ as $x_{k,j} \to 0$

Note that this assumption is not very restrictive; the least information that each agent should have is its own benefit. Moreover, it is reasonable to assume that the more resources an agent obtains, the more benefit it achieves (strictly increasing utility function). The concavity assumption allows us to apply convex optimization techniques [2,14] without restricting the problem solution (it is only necessary to look for the appropriate function that fits this constraint), and the requirement that $u_{k,j}(x_{k,j}) \to -\infty$ as $x_{k,j} \to 0$ allows us to avoid the possibility of any agent getting no resources.

We thus pose the following optimization problem:

$$\max_{\substack{(\mathrm{x}_1, \mathrm{x}_2, ..., \mathrm{x}_{N_a}), \\ (v_1, v_2, ..., v_{N_a})}} \sum_{k=1}^{N_a} U_k(\mathrm{x}_k) \tag{2}$$

subject to

$$x_{k,j} \geq 0, \quad \text{for all } (k,j) \in \mathcal{A} \times \mathcal{R} \tag{3a}$$

$$v_k \in \mathcal{V}, \quad \text{for all } k \in \mathcal{A} \tag{3b}$$

$$\sum_{\{k : v_k = i\}} x_{k,j} \leq r_{i,j}, \quad \text{for all } (i,j) \in \mathcal{V} \times \mathcal{R} \tag{3c}$$

Remark 1. *Implicit in equation* (3c) *is the fact that given that an agent is located at a particular node, it can only have access to the resources of that particular node.*

[1]The communication links however may be asymmetric. This would have an impact on how often and with how much delay the nodes may be able to access information about their neighbors. This is a possible future line of research, but is outside the scope of this paper.

To solve the optimization problem (2)-(3), it is only necessary for the graph $\mathcal{G}$ to be connected because in the current setting, the problem is centralized. We assume that all the information about the network is globally available. The topology of the network becomes important when we pursue the decentralization of the optimization algorithm.

3 Hierarchical Solution Architecture

The optimization problem stated in equations (2) and (3), depends on two types of variables: The continuous resource allocated to the agents, and the discrete locations chosen for each one of them. This type of problem is in general quite complex, but Remark 1 greatly simplifies the problem into one that can be solved hierarchically as explained below.

Inspection of equations (2) and (3) reveals that if agents were not allowed to move, so that they may only compete for the resources at the node where each one of them is located, the problem would become that of solving a separate optimization problem inside each node. Moreover, the individual solution of each node's optimization problem would guarantee the solution of the complete network optimization problem of equations (2) and (3).

Let V_i be the set of agents located at node i, i.e. $V_i = \{k \in \mathcal{A} : v_k = i\}$.

Proposition 1. *Given a fixed possible distribution of agents* $(v_1, v_2, ...v_{N_a})$*, the solution of equations* (2) *and* (3) *is given by the solution of:*

$$\max_{\substack{(x_\alpha, x_\beta, ..., x_\gamma):\\ \alpha,\beta,...,\gamma \in V_i}} \sum_{\{\kappa \in V_i\}} U_\kappa(x_\kappa) \tag{4}$$

subject to

$$x_{k,j} \geq 0, \quad \text{for all } (k,j) \in V_i \times \mathcal{R} \tag{5a}$$

$$\sum_{\{k \in V_i\}} x_{k,j} \leq r_{i,j}, \quad \text{for all } j \in \mathcal{R} \tag{5b}$$

for each $i \in \mathcal{V}$.

Proof. Assigning a fixed value $i \in \mathcal{V}$ to each v_k allows us to discard equation (3b), rewrite equation (2) as

$$\sum_{\{i \in \mathcal{V}\}} \left[\max_{\substack{(x_\alpha, x_\beta, ..., x_\gamma):\\ v_\alpha = i, v_\beta = i,\\ ..., v_\gamma = i}} \sum_{\{\kappa : v_\kappa = i\}} U_\kappa(x_\kappa) \right]$$

using the observation in Remark 1. Equations (3a) and (3c) are also rewritten as a set of equations indexed by $\mathcal{V}$ that are independent of the choice of v_k. This proves the claim. □

The problem described in Proposition 1, represents part of the original optimization problem (2)-(3). In order to achieve an equivalent description, the agent distribution $(v_1, v_2, ..., v_{N_a})$ must be considered as a decision variable. Let $N_d = N_v^{N_a}$ be the number of possible ways of distributing the agents among the nodes in the network. This number is guaranteed to be finite because the nodes and the agents are finite, but it may be very large. Let $\mathbf{D} = \{D^l : l \in \{1, 2, ..., N_d\}\}$ be the set of possible distributions of agents in the network, where each individual distribution D^l can be expressed in the form of a bipartite graph, where the set of nodes in D^l is composed by the nodes in the network $\mathcal{V}$ and the agents $\mathcal{A}$, and the edges (that may only link an agent to a node) represent the node assignment for each agent, i.e. $D^l = (\mathcal{V} \bigcup \mathcal{A}, \mathcal{E}_D)$, where $\mathcal{E}_D^l = \{(\chi, \alpha) : \chi \in \mathcal{V}, \alpha \in \mathcal{A}, v_\alpha = \chi\}$.

Assume that the solution of the problem in Proposition 1 is available for any given distribution of agents D^l, then in order to solve the problem in (2)-(3), it is sufficient to obtain the agent distribution that maximizes the benefit of the complete network, i.e.

$$D^* = \arg \max_{\{D^l : 1 \le l \le N_d\}} \mathbf{U}^l \tag{6}$$

where

$$\mathbf{U}^l = \sum_{\{i \in \mathcal{V}\}} \left[\sum_{\{\kappa : \kappa \in V_i\}} U_\kappa(\mathrm{x}_\kappa^{l*}) \right] \tag{7}$$

where x_κ^{l*} denotes the optimal solution of the convex optimization problem (4)-(5) for agent κ during l^{th} possible distribution.

We have thus transformed the mixed continuous and discrete optimization problem (2)-(3) into the hierarchical optimization problem (4),(5), and (6), where the continuous part (4)-(5) can be solved inside each node in the network, while the discrete part (6) has to be solved in a centralized form. The proposed approach in this paper is to obtain an analytical solution for the continuous portion (4)-(5), such that its numerical solution requires a low-computational-cost algorithm. The discrete part can then be solved using randomized algorithms such that a quasi-optimal solution (with high confidence) can be achieved. The procedure is summarized in Figure 2.

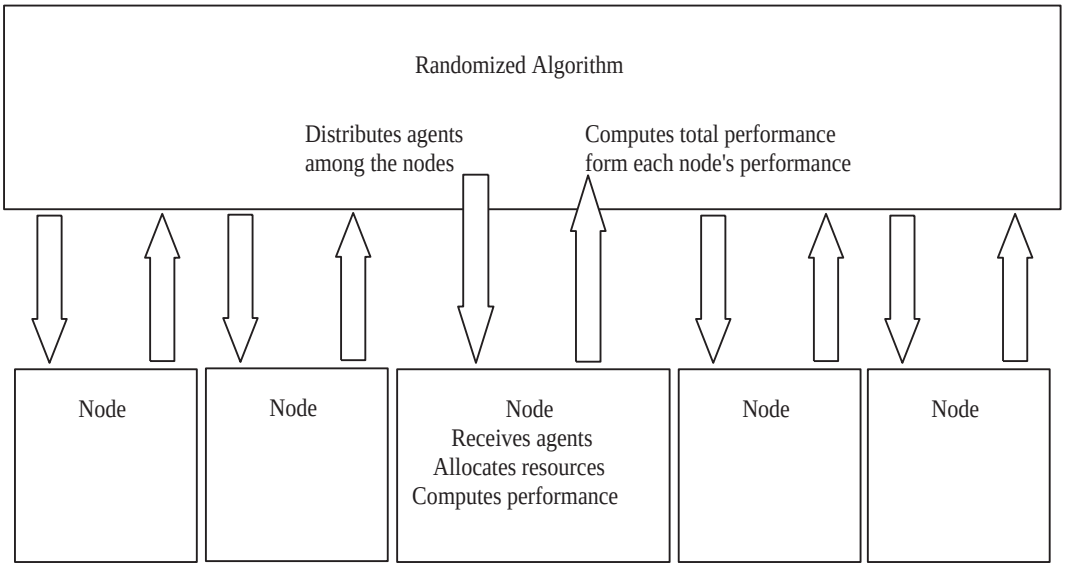


Figure 2: Hierarchical solution structure. The randomized algorithm distributes the agents. The resource is then allocated by each node to the agents it hosts so it can compute its benefit. Finally the centralized algorithm obtains the aggregate benefit.

4 Decentralized Part: Convex Optimization

In this section we provide a solution for the problem stated in Proposition 1. The solution is obtained by applying Karush-Khun-Tucker conditions [14] to our optimization problem, after some transformations that are described below.

From equations (1) and (4) the utility function for each node $i \in \mathcal{V}$ is written, (by reordering the sums) as,

$$\mathcal{U}_i = \sum_{j=1}^{N_r} \left(\sum_{\{k \in V_i\}} u_{k,j}(x_{k,j}) \right) \tag{8}$$

Since $u_{k,j}(x_{k,j})$ is a strictly concave function of $x_{k,j}$ for each $(k,j) \in \mathcal{A} \times \mathcal{R}$, the terms inside the parentheses of equation (8), and the utility function $\mathcal{U}_i$ are all concave functions of their arguments. This fact allows us to decentralize the problem even further, and to consider N_r independent optimization problems within each node (one for each resource). Thus, the optimization problem in equations (5) and (8) can be restated as: For each $(i,j) \in \mathcal{V} \times \mathcal{R}$ maximize:

$$\mathcal{U}_{i,j} = \sum_{\{k \in V_i\}} u_{k,j}(x_{k,j}) \tag{9}$$

subject to

$$x_{k,j} \geq 0, \quad \text{for all } k \in V_i \tag{10a}$$

$$\sum_{\{k \in V_i\}} x_{k,j} \leq r_{i,j} \tag{10b}$$

In order to simplify our notation, we drop the i and j subindexes from the previous equations, with the understanding that the problem stated below is solved for each resource inside each node of the network. The problem is then stated as: Maximize

$$\mathcal{U} = \sum_{k=1}^{n_a} u_k(x_k), \tag{11}$$

subject to

$$x_k \geq 0, \quad \text{for all } k \in \{1,2,...,n_a\} \tag{12a}$$

$$\sum_{k=1}^{n_a} x_k \leq r \tag{12b}$$

where n_a is the number of agents residing in each node (the notation of this variable is also simplified by dropping its dependence on $i \in \mathcal{V}$).

Lemma 1. *Let $\lambda_p \in \mathbb{R}$ for all $p \in \{1,2,...,n_a+1\}$. The necessary and sufficient conditions for $(x_1^*, x_2^*, ..., x_{n_a}^*)$ to be the maximal solution of the problem in* (11)-(12) *are:*

$$\frac{du_p}{dx_p} + \lambda_{n_a+1} - \lambda_p = 0 \;\; \forall p \in \{1,2,...,n_a\} \tag{13a}$$

$$\lambda_p x_p = 0 \;\; \forall p \in \{1,2,...,n_a\} \tag{13b}$$

$$\lambda_{n_a+1}\left(\sum_{k=1}^{n_a}(x_k) - r\right) = 0 \tag{13c}$$

$$-x_p \leq 0 \;\; \forall p \in \{1,2,...,n_a\} \tag{13d}$$

$$\sum_{k=1}^{n_a}(x_k) - r \leq 0 \tag{13e}$$

$$\lambda_p \leq 0 \;\; \forall p \in \{1,2,...,n_a+1\} \tag{13f}$$

Proof. Let $g_p = -x_p$ for all $p \in \{1,2,...,n_a\}$, and $g_{n_a+1} = \sum_{k=1}^{n_a}(x_k^*) - r$ which are the constraints (12) of the problem. Then, and using Lagrange multipliers, the Karush-Khun-Tucker conditions for optimality [14] become

$$\begin{aligned}
\frac{\partial \mathcal{U}}{\partial x_p} + \sum_{j=1}^{n_a+1} \lambda_j \frac{\partial g_j}{\partial x_p} = 0 \qquad & \forall p \in \{1,2,...,n_a\} \\
\lambda_p g_p = 0 \qquad & \forall p \in \{1,2,...,n_a+1\} \\
g_p \leq 0 \qquad & \forall p \in \{1,2,...,n_a+1\} \\
\lambda_p \leq 0 \qquad & \forall p \in \{1,2,...,n_a+1\}
\end{aligned}$$

Evaluating the derivatives we obtain $\frac{\partial \mathcal{U}}{\partial x_p} = \frac{du_p}{x_p}$ for all $p \in \{1,2,...,n_a\}$,

$$\frac{\partial g_j}{\partial x_p} = \begin{cases} 0 & j \neq p \\ -1 & j = p \end{cases}$$

for all $j, p \in \{1,2,...,n_a\}$, and $\frac{\partial g_{n_a+1}}{\partial x_p} = 1$ for all $p \in \{1,2,...,n_a\}$. Substituting these derivatives back into the previous equations we obtain equation (13). Since both the utility function (11) and the constraints (12) are strictly concave, equation (13) becomes a necessary and sufficient condition for the optimality of $(x_1^*, x_2^*, ..., x_{n_a}^*)$. □

Consider equation (13e), and note that since $x_p \geq 0$ for all $p \in \{1,2,...,n_a\}$ and that $u_p(x_p)$ is a strictly increasing function of x_p for all $p \in \{1,2,...,n_a\}$, any choice of $(x_1, x_2, ..., x_{n_a})$ for (13e) such that $\sum_{k=1}^{n_a}(x_k) - r < 0$ will be suboptimal. Thus equation (13e) must be modified as $\sum_{k=1}^{n_a}(x_k) - r = 0$. This conclusion automatically discards equation (13c) because it is trivially satisfied.

Equation (13b) provides two choices for each p: $\lambda_p = 0$ or $x_p = 0$. The second choice however yields $u_p(x_p) = -\infty$, violating the maximization of the utility function $\mathcal{U}$. Thus $\lambda_p = 0$ for all $p \in \{1,2,...,n_a\}$. The same argument leads to the modification of equation (13d) to $-x_p < 0$ for all $p \in \{1,2,...,n_a\}$. As a consequence of these observations (13a) and (13f) are simplified. Conditions (13) are then modified as:

$$\frac{du_p}{dx_p} + \lambda = 0 \ \ \forall p \in \{1,2,...,n_a\} \tag{14a}$$

$$\sum_{k=1}^{n_a}(x_k) - r = 0 \tag{14b}$$

$$x_p > 0 \ \ \forall p \in \{1,2,...,n_a\} \tag{14c}$$

$$\lambda \leq 0 \tag{14d}$$

where λ_{n_a+1} has been renamed as λ. We thus have the following result.

Lemma 2. *The solution* $(x_1^*, x_2^*, .., x_{n_a}^*, \lambda_1^*, \lambda_2^*, ...\lambda_{n_a+1}^*)$ *of equation* (13) *is given by* $\lambda_p^* = 0$ *for* $p \in \{1,2,...,n_a\}$ *and by equations* (14) *for* $(x_1^*, x_2^*, .., x_{n_a}^*, \lambda_{n_a+1}^*)$ *where* $\lambda = \lambda_{n_a+1}^*$.

Applying Lemmas 1 and 2 to equations (9) and (10), the main result of this section is stated as:

Theorem 1. *Given the available resource* $r_{i,j}$ *of type* $j \in \mathcal{R}$ *in the node* $i \in \mathcal{V}$*, the utility function* (9) *is maximized by* $(x_{\alpha,j}^*, x_{\beta,j}^*, ..., x_{\gamma,j}^*)$*, where* $\alpha, \beta, ...\gamma \in V_i$ *subject to* (10) *if and only if for each* $(i,j) \in \mathcal{V} \times \mathcal{R}$*,* $(x_{\alpha,j}^*, x_{\beta,j}^*, ..., x_{\gamma,j}^*)$ *satisfies:*

$$\left.\frac{du_{\kappa,j}}{dx_{\kappa,j}}\right|_{x_{\kappa,j}=x_{\kappa,j}^*} + \lambda_{i,j}^* = 0 \ \ \forall \kappa \in V_i \tag{15a}$$

$$\sum_{\kappa \in V_i}(x_{\kappa,j}^*) - r_{i,j} = 0 \tag{15b}$$

$$x_{\kappa,j}^* > 0 \ \ \forall \kappa \in V_i \tag{15c}$$

$$\lambda_{i,j}^* \leq 0 \tag{15d}$$

5 Centralized Part: Randomized Optimization

In this section we propose a centralized algorithm to solve equation (6). This is a combinatorial optimization problem [3] in general, but in this paper we pursue an alternative path through the use of randomized algorithms, which have proven useful in a different but somewhat related problem [12]. We choose randomized algorithms for

this part of the problem because as opposed to model-based techniques, their computational complexity does not depend exclusively on the complexity of the problem itself. For example, an alternative would have been to use mixed integer programming [6] to solve the whole problem, but the computational complexity of this technique is NP-complete on the integer variables ([17] Ch. 18), which in our case is large as noted in the simulation example presented in Section 6.

The main idea of using randomized algorithms is that instead of looking for the solution that exactly optimizes the problem, we seek a solution that is close to the optimal one with high confidence. This is done by generating a given number of samples from the set of possible solutions and choosing the one with the best performance. This can be stated more formally as follows: Let $\mathcal{D} = \{l : 1 \leq l \leq N_d\}$ be the index set of all possible agent distributions.

Definition 1 (Probable Near Maximum)**.** *Given the set of all possible network benefits* $\{\mathbf{U}^l : l \in \mathcal{D}\}$, $\delta \in (0,1)$, *and* $\alpha \in (0,1)$, *a number* $\mathbf{U}^0 \in \mathbb{R}$ *is said to be a probable near maximum of* $\{\mathbf{U}^l : l \in \mathcal{D}\}$ *to level* α *and confidence* $1-\delta$ *if there exists a set* $\tilde{\mathcal{D}} \subseteq \mathcal{D}$ *with* $Pr\{\tilde{\mathcal{D}}\} \leq \alpha$ *such that*

$$Pr\Big\{ \sup_{l \in \mathcal{D}} \mathbf{U}^l \geq \mathbf{U}^0 \geq \sup_{l \in \mathcal{D} \setminus \tilde{\mathcal{D}}} \mathbf{U}^l \Big\} \geq 1-\delta \tag{16}$$

Thus, this definition states that the probability of finding a better solution than the Probable Near Maximum is less than α with a confidence greater than $1-\delta$. One result that will be used to obtain the number of samples needed to achieve a Probable Near Maximum as a function of the desired confidence and probability of finding a better solution is the following:

Lemma 3 (Theorem 9.1 [20])**.** *Suppose* $(Y, \mathcal{S}_Y, P_Y)$ *is a probability space, and that* $\nu : Y \to \mathbb{R}$ *is a random variable. Let* $y_1, y_2, ..., y_m \in Y$ *be m independent and identically distributed (i.i.d.) samples drawn according to* P_Y, *and define*

$$\nu^0(\mathbf{y}) = \max_{1 \leq i \leq m} \nu(y_i) \tag{17}$$

Given any $\alpha \in (0,1)$ *and* $\delta \in (0,1)$, *if*

$$m \geq \frac{\ln(1/\delta)}{\ln(1/(1-\alpha))} \tag{18}$$

then, with probability greater than $1-\delta$, *we have* $P_Y\{\nu(y) > \nu^0(\mathbf{y})\} \leq \alpha$, *i.e.*

$$Pr^m\{P_Y\{\nu(y) > \nu^0(\mathbf{y})\} \leq \alpha\} \geq 1-\delta \tag{19}$$

Note that equation (19) resembles (16), implying that the bound provided in (18) may be used to compute the minimum number of samples needed to guarantee that the solution obtained through the randomized algorithm is sufficiently close to the actual solution. We thus have the following result:

Theorem 2. *Given* $\delta \in (0,1)$ *and* $\alpha \in (0,1)$, *if M i.i.d. samples* $D^1, D^2, ..., D^M$ *are taken from* $\mathcal{D}$ *such that* $M \geq \frac{\ln(1/\delta)}{\ln(1/(1-\alpha))}$ *then*

$$\mathbf{U}^0 = \max_{\{1 \leq \eta \leq M\}} \mathbf{U}^\eta$$

is a Probable Near Maximum of $\{\mathbf{U}^l : l \in \mathcal{D}\}$ *to level* α *and confidence* $1-\delta$ *and the corresponding* D^0 *is the Probable Near Optimal distribution of agents in the network.*

Theorems 1 and 2 allow us to establish Algorithm 1.

Algorithm 1 Hierarchical Optimization

Require: α, δ, $\mathcal{V}$, $\mathcal{A}$, $r_{i,j}$ for all $(i,j) \in \mathcal{V} \times \mathcal{R}$, and $U_k(\mathbf{x}_k)$ for all $k \in \mathcal{A}$.
Ensure: $\mathbf{U}^0$ and D^0.

1: $l \Leftarrow 1$
2: Compute M according to Theorem 2.
3: **while** $l \leq M$ **do**
4: Generate an agent distribution $D^l \in \mathcal{D}$.
5: For each $(i,j) \in \mathcal{V} \times \mathcal{R}$, compute $x^*_{\kappa,j}$ for all $\kappa \in V_i$ (according to Theorem 1).
6: Compute the aggregate benefit

$$\mathbf{U}^l = \sum_{\mathcal{A}} U_k(\mathbf{x}_k).$$

7: $l \Leftarrow l + 1$
8: **end while**
9: Choose $\mathbf{U}^0 = \min_{1 \leq l \leq M} \mathbf{U}^l$ and the corresponding D^0.

6 Simulation Results

6.1 Experimental set-up

We present simulation results for the complete algorithm (Algorithm 1) to illustrate the benefits and limitations of the proposed technique. The test involves 40 nodes ($N_v = 40$) in the network, and 50 agents ($N_a = 50$) that must be distributed among the nodes.

Each node is assumed to have three types of resources, e.g. bandwidth, processing power, and storage memory ($N_r = 3$), so three constants $r_{i,1},\ r_{i,2},\ r_{i,3}$ are assigned to each node $i \in \mathcal{V}$, representing the amount of each type of resource available to each node, i.e. the capability of each node. For the test, these constants were chosen at random according to a uniform distribution between 0 and 1. The choices were made independently for each constant, and before starting the test.

The utility function for each agent $k \in \mathcal{A}$ is of the form

$$U_k(\mathbf{x}_k) = \sum_{j=1}^{3} w_{k,j} \ln(x_{k,j}) \tag{20}$$

where $w_{i,j}$ were also independently chosen at random according to a uniform distribution between 0 and 1, and before running the test. Utility function (20) satisfies Assumption 4. Note that $w_{k,j}$ are weighting factors for each type of resource on each agent, and are used to quantify the importance that each resource has for each agent (the greater the value of $w_{k,j}$ the more important is resource j for agent k.) Note that the particular choice of utility function for this test is commonly referred to as proportional fairness in the mathematical modeling for the Internet literature [1, 9, 16, 18, 21].

6.2 Algorithm execution

We now explain how Algorithm 1 behaves in this example:

Step 2 Given α and δ, the number of samples M is calculated. The following steps are then repeated M times.

Steps 3-4 A sample of an agent distribution is generated. This is done by assigning a node to each agent in the system. Each assignment is picked independently and according to a uniform distribution among the nodes. Note that each distribution sample must be independent of the previous ones.

Step 5 Given the agent distribution obtained in the previous step, the algorithm must now solve the decentralized resource allocation for each resource in each node. This implies that (9), (10) must be solved using Theorem 1 (with the substitution (20) as utility the function for each agent.) However, it is possible to simplify this step analytically so that the computational time for the simulation is reduced. Then, applying Theorem 1 to (20) we obtain the following optimal solution for each $(i, j) \in \mathcal{V} \times \mathcal{R}$:

$$x^*_{k,j} = \frac{r_{i,j} w_{k,j}}{\sum_{\{\kappa \in V_i\}} w_{\kappa,j}} \quad \forall k \in V_i \tag{21a}$$

$$\lambda^*_{k,j} = -\frac{1}{r_{i,j}} \sum_{\{\kappa \in V_i\}} w_{\kappa,j} \quad \forall k \in V_i \tag{21b}$$

Thus, given the resources for each node $r_{i,j}$ and the weights for each agent $w_{k,j}$, this step can be completed by substituting this information into (21).

Step 6 Once the resources have been distributed among the agents, the algorithm calculates the utility of the current distribution.

Step 9 Finally the algorithm chooses as optimal the distribution that yields the best performance among the tested samples.

6.3 Overall performance of the algorithm

The first test consists of running the optimization algorithm several times in order to relate its performance to different values for α and δ. The chosen values for the experiment are summarized in Table I. The third row of this table provides the number of samples M corresponding to each $\alpha - \delta$ pair. The test is designed as follows.

α	0.01	0.01	0.005	0.005	0.005	0.001	0.001
δ	0.01	0.005	0.01	0.005	0.001	0.005	0.001
M	458	528	919	1058	1379	5296	6905

Table I: Values for α and δ used in the first test.

For each $\alpha - \delta$ pair, twelve different instantiations of Algorithm 1 are executed. From these twelve cases, the average, the standard deviation, and the maximum performance are computed. The results for each $\alpha - \delta$ pair are shown in Figure 3. As can be seen from this figure, the performance of the algorithm improves on average as the number of samples is increased. Moreover, the standard deviation tends to decrease with a higher number of samples which is an indication of the improvement on the repeatability of the algorithm. However, there is also an increase of the standard deviation on the tests with 6905 samples, which contradicts the tendency. This is because the algorithm obtained a much better, but rare optimal solution, that can be seen in the "$*$" located approximately at $(6905, -78.3)$. This rare case does not a contradict Theorem 2, because the Theorem only guarantees that the probability of finding a better solution is small, not zero. In fact, there could be a much better solution that is never identified by the samples. This is the price to pay for the simplicity provided by the randomized algorithm. Even so, it must be noted that the number of samples needed to obtain a reasonable result (6905), is much smaller compared to the number of possible arrangements of agents in the network that is given by $N_v^{N_a}$, which for our test is 1.2677×10^{80}.

6.4 Further discussion on the results

To gain insight in the behavior of the algorithm, we perform a second test, where only one optimization (with identical set-up to the previous) was performed, but more variables were stored. The confidence and level pa-

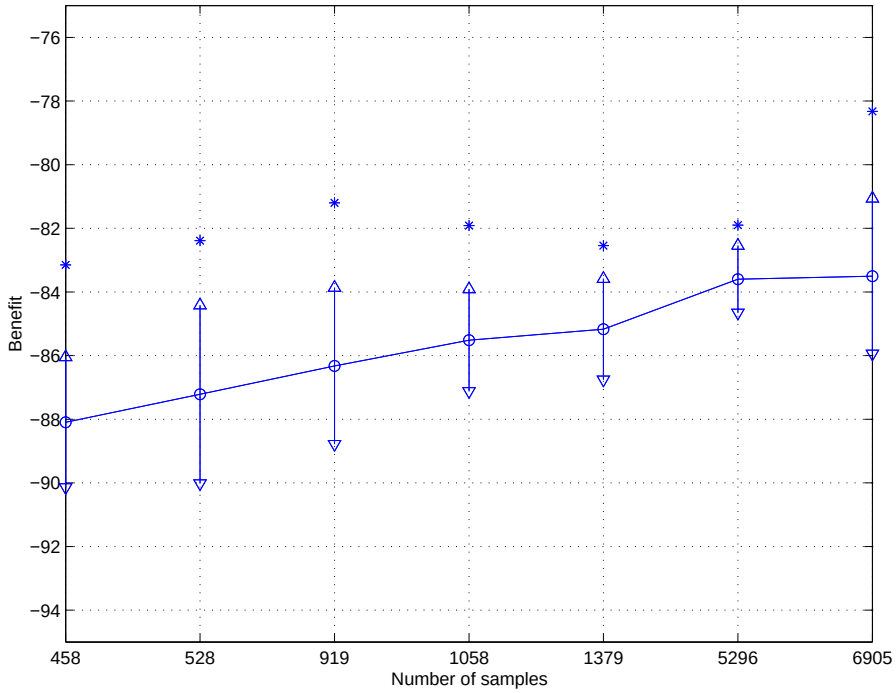


Figure 3: Plot of the average, standard deviation and maximum performance for each number of samples in table I. "o" is used for average performance, "△" for average plus standard deviation, "▽" for average minus standard deviation and "∗" for maximum performance.

rameters are chosen as $\alpha = \delta = 0.0001$ yielding a number of samples $M = 6905$. The optimal performance is $\mathbf{U}^0 = -82.8748$, which is close to the average performance obtained for that number of samples (shown in Figure 3) but not the best possible known which was about -78.3 (also shown in Figure 3). This confirms that the randomized approach only guarantees a good approximation to the optimal solution.

The decentralized (convex) optimization algorithm distributes the resources on each node among the agents located in it, according to the weights $w_{k,j}$ of each agent's utility function. An example of such distribution is given in Table II, which summarizes the resource allocation for node 1, where agents 21 and 30 are located after the optimization. In this table, each resource $r_{1,j}$ is distributed according to the weights $w_{21,j}$, and $w_{30,j}$ (following 21), resulting on the allocation given by $x_{21,j}$, and $x_{30,j}$. These results satisfy the conditions given in Theorem 1 for the proportional fairness allocation (21a) as expected.

j	1	2	3
$r_{1,j}$	0.9501	0.8381	0.7948
$w_{21,j}$	0.6833	0.2319	0.2974
$w_{30,j}$	0.5751	0.9943	0.7334
$x_{21,j}$	0.5159	0.1585	0.2293
$x_{30,j}$	0.4342	0.6796	0.5655

Table II: Resource distribution for node 1 with agents 21 and 30.

We were also interested in observing the behavior of the randomized algorithm with respect to the weights in each agent's utility function. Specifically, it would be desirable that large amounts of resources were allocated to agents with large weight values. Figures 4, 5, 6 show that this happens on average, as we explain below.

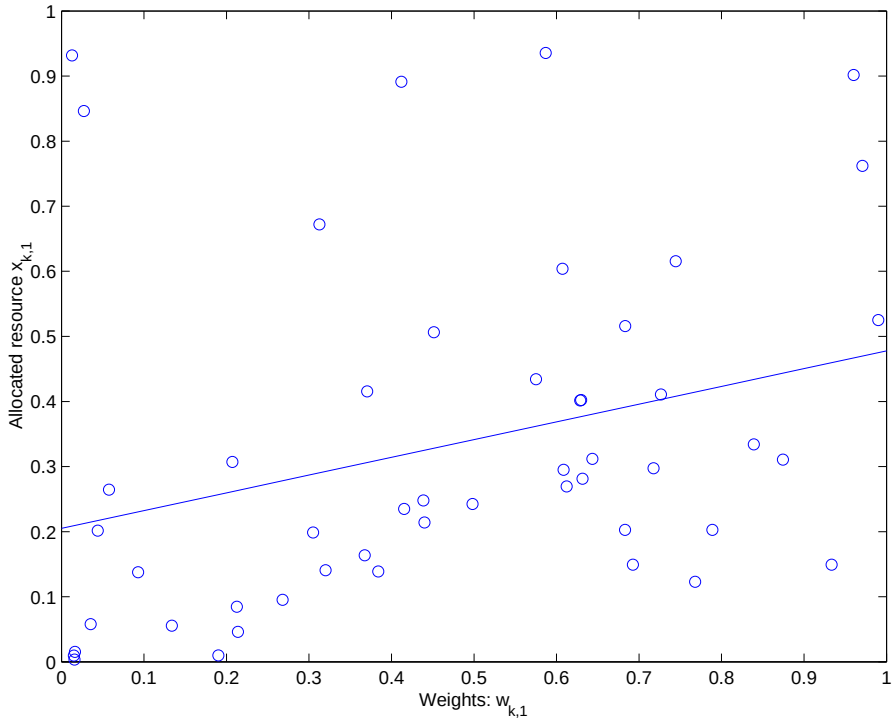


Figure 4: Allocated resource $x_{k,1}$ with respect to weighing factor $w_{k,1}$ for resource $j = 1$. "o" represents each $(w_{k,1}, x_{k,1})$ pair and the line shows the tendency according to a linear fit function (Although the distribution of points does not suggest a linear model, we use the line to highlight the tendency of the algorithm to allocate more resources to agents with higher weights in their utility function).

We present three figures, one for each resource in the experiment (Figure 4 for the first resource, Figure 5 for the second, and Figure 6 for the third). Each "o"-mark in the figures represents an amount of allocated resource that corresponds to each agents weight, i.e. $(w_{k,j}, x_{k,j})$. Then if for resource $j = 1$ an "o"-mark is located at $(0.4, 0.2)$ it means that 0.2 of the first resource was allocated to an agent that had a weight of 0.4 for such resource. Additionally, each figure shows a tendency line that was constructed by sorting the pairs $(w_{k,j}, x_{k,j})$ in increasing order with respect to $(w_{k,j}$ and then obtaining a linear fit function from the ordered data $(w_{k,j}, x_{k,j})$ for $k \in \mathcal{A}$. The linear fit function was only used to highlight the overall tendency of the solution. We do not claim that the data follows a linear model, in fact the obtained regression coefficient is too low to imply that.

These figures show that for all three resources the expected tendency of allocating more resources to agents with higher weights is fulfilled. However, these figures also show several weight-allocation pairs that contradict the tendency completely ("o"-marks located near the points $(0,1)$ and $(1,0)$). These points are indicators that a better solution may yet be achieved, which we know is true because the best possible known performance was higher than that achieved in this test. This happens because the randomized algorithm only guarantees that the solution will be very good with very high confidence, but not necessarily the best one.

7 Conclusion

We have modeled a network agent distribution problem as a continuous resource allocation on a discrete environment (a graph), where the decision variables are the location of each agent in the network and the resources each of them receive. This optimization problem was solved using two different algorithms configured in a hierarchical structure. The agent distribution algorithm is centralized and of a randomized nature, while within each node,

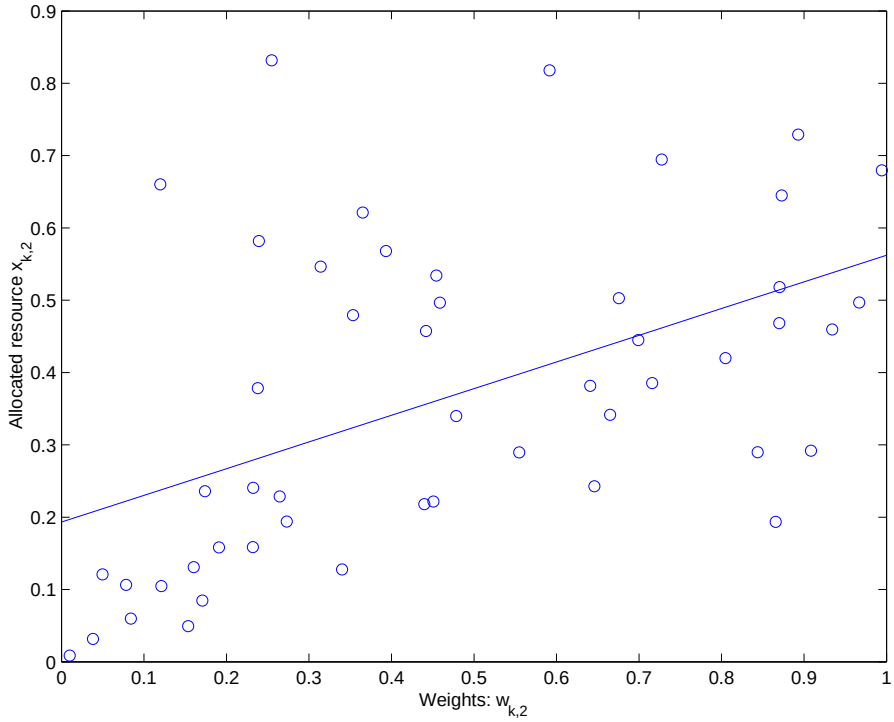


Figure 5: Allocated resource $x_{k,2}$ with respect to weighing factor $w_{k,2}$ for resource $j = 2$. "o" represents each $(w_{k,2}, x_{k,2})$ pair and the line shows the tendency according to a linear fit function

the decentralized resource allocation algorithm relies on the Karush-Khun-Tucker conditions. Simulation results have been presented showing that the approach yields the anticipated results.

The main limitation of our current algorithm lies in its centralized part. Within each node however, the decentralized part of the optimization problem resembles that of Kelly's mathematical modeling of TCP protocols [9, 18]. This suggests that the decentralization of our algorithm may be achieved using the primal-dual approach that corresponds to this type of modeling [1, 16, 18] as starting point, augmenting this algorithm with switching logic that controls the movement of agents between nodes in the network.

References

[1] T. Alpcan and T. Başar. A utility-based congestion control scheme for internet-style networks with delay. In *Proc. of the IEEE Infocom*, volume 3, pages 2039–2048, San Francisco, CA, USA, April 2003.

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, United Kingdom, 2004.

[3] W. Cook, W. Cunningham, W. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. John Wiley and Sons, 1997.

[4] J. Cortez and F. Bullo. Coordination and geometric optimization via distributed dynamical systems. *SIAM Journal on Control and Optimization*, 44(5):1543–1574, 2005.

[5] V. Gazi and M. Passino. Stability analysis of swarms. *IEEE Transactions on Automatic Control*, 48(4):692–697, Apr. 2003.

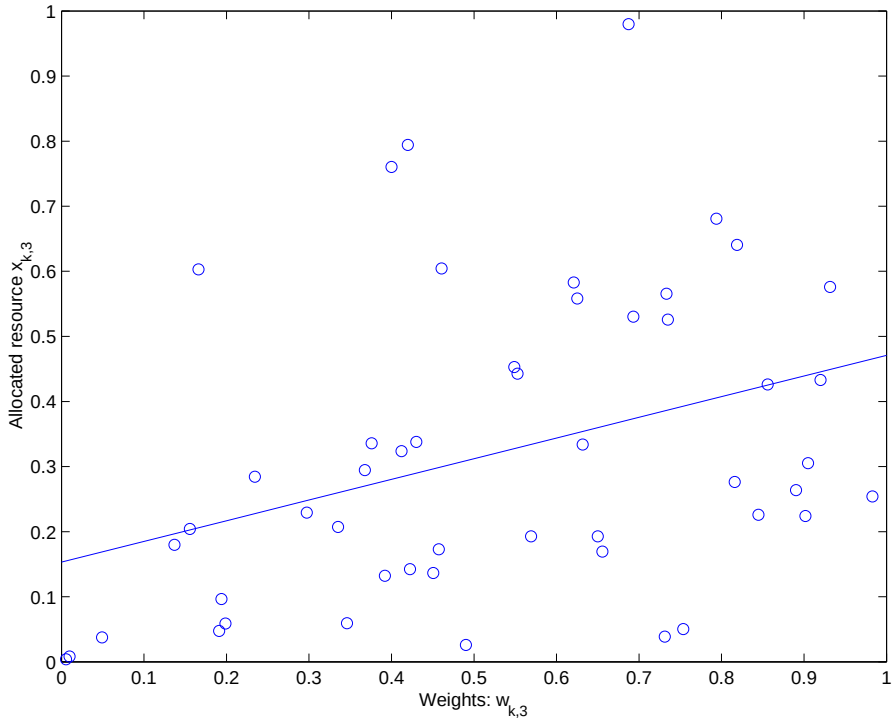


Figure 6: Allocated resource $x_{k,3}$ with respect to weighing factor $w_{k,3}$ for resource $j = 3$. "o" represents each $(w_{k,3}, x_{k,3})$ pair and the line shows the tendency according to a linear fit function

[6] I. Grossmann. Review of nonlinear mixed-integer and disjunctive programming techniques. *Journal on Optimization and Engineering*, 3(3):227–252, Sept. 2002.

[7] A. Jadbabaie, J. Lin, and S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, June 2003.

[8] H. Jerez, C. Abdallah, and J. Khoury. A mobile transient network architecture. Pre-print available at `http://hdl.handle.net/2118/hj_tran_06`, 2006.

[9] F. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8:33–37, 1997.

[10] L. Moreau. Stability of multiagent systems with time-dependent communication links. *IEEE Transactions on Automatic Control*, 50(2):169–182, Feb. 2005.

[11] R. Olfati-Saber. Consensus problems in networks of agents with switching topology and time delay systems. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, Sept. 2004.

[12] J. Piovesan, C. Abdallah, M. Egerstedt, H. Tanner, and Y. Wardi. Statistical learning for optimal control of hybrid systems. In *Proceedings of the American Control Conference*, New York, NY, USA, July 2007. To appear.

[13] D. Qiu and R. Srikant. Modeling and performance analysis of BitTorrent-like peer-to-peer networks. In *Proc. of the SIGCOMM'04*, Portland, Oregon, USA, Aug. 30-Sept. 3 2004. ACM 1-58113-862-8/04/0008.

[14] S. Rao. *Engineering Optimization: Theory and Practice*. John Wiley and Sons, Inc., third edition, 1996.

[15] W. Ren, R. Beard, and E. Atkins. A survey of consensus problems in multi-agent coordination. In *Proceedings of the American Control Conference*, pages 1859–1864, Portland, OR, USA, June 8-10 2005.

[16] R. Sandoval-Rodriguez, C. Abdallah, P. Hokayem, E. Schamiloglu, and R. Byrne. Robust mobile robotic formation control using internet-like protocols. In *Proceedings of the IEEE Conference on Decision and Control*, pages 5109–5112, Maui, Hawaii, USA, Dec 2003.

[17] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1998.

[18] R. Srikant. *The Mathematics of Internet Congestion Control*. Birkhauser, 2004.

[19] H. Tanner, A. Jadbabaie, and G. Pappas. Stable flocking of mobile agents, part I: Fixed topology. In *Proceedings of the IEEE Conference on Decision and Control*, pages 2010–2015, Maui, Hawaii, USA, Dec 2003.

[20] R. Tempo, G. Calafiore, and F. Dabbene. *Randomized Algorithms for Analysis and Control of Uncertain Systems*. Communications and Control Engineering. Springer, London, UK, 2003.

[21] J. Wen and M. Arcak. A unifying passivity framework for network flow control. In *Proc. of the IEEE Infocom*, volume 2, pages 1156–1166, San Francisco, CA, USA, April 2003.