# Simulating social-based forwarding in opportunistic networks

Luca Mella - luca.mella@studio.unibo.it
SMA1213 Class Project Report
Alma Mater Studiorum – University of Bologna

April 21, 2013

## 1 Introduction

In the last ten years we have assisted to the explosion of new pervasive technologies which have changed the way we intend computing, modified our personal and social habits, and consequentially opened new opportunities and research branch.
Pervasive computing has been matter of researches by both academia and industry, so a good amount of literature has been published during this period; in a synthetic fashion we can extrapolate that we are facing some kind of open, distributed, and possibly heterogeneous complex-system which have to achieve a goal without a centralized coordination. For this reason developing algorithms and solutions that run on top of this kind of systems are considered at least challenging by software engineers, in fact the difficulty to apply formal methods for validating this kind of systems have enhanced the simulation's role inside development processes.
Having said that, in this class project we aim to exploit this important tool in the software engineer arsenal (simulation) for setting up an experiment related to pervasive computing.
Before describe the experiment we have to provide the reader a bit more information which helps to contextualize it.
Background theme of this project is the social-based forwarding in opportunistic network, so, broadly speaking, we are facing the problem of delivery or acquire a content in an infrastructure-free scenario composed by a multitude of mobile-device (typically carried by humans) by exploiting what we know about social relationship between devices/humans with respect to their movements and their consequential physical contacts.
This class of problems were one of the topics in *SOCIALNETS: Social networking for pervasive adaptation*[14]: an European FP7-ICT research project which investigated this kind of problems and produced several prototypes spendable in infrastructure-free scenario.
In particular, we are focusing in a subtopic treated during this project: algorithms for social-based forwarding in opportunistic, delay tolerant network. Several algorithms have been evaluated during the course of SOCIALNETS and an interesting fact we have noted is that performed experiments were mainly

based on emulation approaches[7] based on contact network data collected during *Haggle Project*[1].
Due to the notorious difficulty of collecting this kind of data, in fact collecting process involved lot of people and required from days to years, we consider interesting to experiment this kind of algorithm in a simulated environment, trying to adopt proper mobility models in order to obtain realistic performance measurement.
In the following sections we'll first review mayor social network topologies2, which are exploited by forwarding algorithms, on mobility models4 that enable to reproduce realistic contact network dynamics, even using social network informations, and also a review on social-based forwarding algorithm6 with a focus on *Bubble* algorithms[7], because they shown better performance in emulations performed during SOCIALNETS project, terminating with the description and the results of the class project experiment7.

[1] `http://www.haggleproject.org`

## 2 Social Networks Models

Field of *Social Network Analysis* became one of the key field in computer science's interdisciplinary research, so before introducing what kind of properties have been observed in real social network we have to briefly describe typical metrics and measurements adopted in social network analysis:

**Degree** defined as the number of links (or relationships) that a network node have[12, 7.1]. In networks represented as directed graph *in-degree* and *out-degree* are used to measure incoming and outgoing links respectively. Degree is one of the basic metrics in social network analysis and the study of it's distribution across nodes plays a relevant role in SNA.

**Clustering** we can define clustering coefficient for a generic node $n_i$ as $\frac{number of pairs of neighbors of node_i that are connected}{number of pairs of neighbors of node_i}$. A whole network clustering coefficient could be calculated as $C = \frac{1}{n}\sum_{i=0}^{n} C_i$[17, Figure 2 caption]. This measurement try to catch the node's tendency to form groups (more formally, completely connected sub-graphs).

**Average path length** measure the efficiency of the network in terms of how easy is the transport of an information from a node to another. It is defined as $\frac{\sum_{i,j} d(i,j)}{n \cdot (n-1)}$ where $i, j$ are network nodes, $n$ is the number of network nodes and $d(\cdot,\cdot)$ is the shortest path cost between two nodes (0 in case of not reachability or $i = j$).

**Centrality** this kind of measures try to answer to the following question *"Which are the most important nodes in a network?"*. Several kinds of centrality measures have been defined in order to characterize node's importance, like:

- *Degree Centrality*, which simply consider node's degree (as described previously)
- *Betweenness*, focus on how much a node is between other nodes, in particular can be defined as $\frac{\sum_{j<k} g_{jk}(i)}{g_j k}$ where $g_{jk}$ is the number of shortest paths connecting $jk$ and $g_{jk}(i)$ is the number that shortest path which includes node $i$.
- *Eigenvector centrality* exploits different concept, this measure consider node centrality directly related to neighbours centralities[12, 7.2] (eg. if you have important neighbours you are much more important). It can be intuitively expressed as $C(i) = \sum_{k \in neighborhood} \omega_{ki} \cdot C(k)$ where $\omega_{ki}$ is the weight of the relationship (if present) and $C(\cdot)$ is the centrality.
- *Katz centrality* is a refinement of eigenvector centrality which enable to modulate the effect of neighbours centrality in node centrality calculation. More formally we can express node-$i$ centrality as $C_i(\beta) = \sum_{j \in neighborhood} (\alpha + \beta \cdot C_j) \cdot \omega_{ij}$ where $\beta$ is modulates the importance of neighbours centrality and $\alpha$ is a normalized constant. However this kind of centrality have an undesired side-effect: if an high degree vertex point to several (eg.millions) low degree ones they also get high centrality values, of course this behaviour is not properly realistic[12, 7.4].

- *PageRank centrality* tries to overcome to Katz centrality misbehaviour by modulating the contribution of the neighbours with the their out degree. So we can enhance previous centrality expression adding a $k_j^{out}$ term which represent the out-degree of node-$j$, resulting in $C_i(\beta) = \sum_{j \in neighborhood} (\alpha + \beta \cdot \frac{C_j}{k_j^{out}}) \cdot \omega_{ij}$.

# 3 Small-World networks

Real world network have been deeply investigated in past decades and similar characteristics have been observed in both biological, social, and computer networks (eg.web-pages or e-mails[6]). The common properties observed during investigations were fundamental for the development of networks models and consequential algorithm for realistic network generation - which are basic for setting up our experiment - so we need to introduce these properties:

**Small-world property** real world networks are typically characterized by the existence of short-cuts between different network areas, this contribute do maintain low average path length despite network size[3, 2.2.1] speeding up the communication among otherwise distant nodes.

**Scale-free** degree distribution across network nodes follow a power-law such as $p(x) \sim x^{-\gamma}$ where $\gamma$ is a parameter typically between 2 and 3. An important property of this kind of distribution (which is an exponential distribution) is that it results to be scale invariant. Note that a distribution is scale invariant if $p(b \cdot x) = f(b)p(x)$, and if we consider a power-law distribution $p(c \cdot x) = (c \cdot x)^{\gamma} = c^{\gamma} f(x) \propto p(x)$.
Power-law distribution is also known as *Pareto distribution* (80/20 principle) so roughly speaking we can observe that there are much more nodes with low-degree values rather than high-degree nodes.

**Community structures** real world networks are typically characterized by high[2] clustering coefficient's values, in fact fully connected sub-graphs (k-cliques) are a common motif in small-world networks[3, 2.1.5]. A recent analysis performed on data collected by an ad hoc massive multiplayer online game[15] have observed that high values of clustering are related to positive interactions, showing a correlation between communities and performance.

## 3.1 Barabási-Albert model

BA model[2] is a network model based on two observable concepts of real networks: growth, and preferential attachment. Older nodes become more popular during network growth and consequentially it's more probable that new nodes will link to them.
The algorithm consider degree as measure of popularity and use a probability of attachment calculated as follow: $p_i = \frac{k_i}{\sum_j k_j}$, where $p_i$ is the probability of creation of a connection (edge) between the new node and the node-$i$, $k_i$ is the

[2]High clustering coefficient - and also low average path - are referred to the same measures taken from random graphs (eg. Erdos–Renyi model)

node-$i$ degree, and $j \in 1 \cdots N$.
However BA-generated network does not provide a full small-word network. Despite the scale-free ($p(x) = x^{-3}$) degree distribution and the relatively low average short path ($\ell \sim \frac{\ln N}{\ln \ln N}$.), it doesn't capture typical motifs of real world networks: high clustering values and communities.
In fact in BA-model the clustering coefficient scales with network size in a power-law fashion, this behaviour is still different from small-world networks where clustering is independent from graph size.

**Algorithm 1** BA network generation algorithm

**for** $node_i$ such that $node_i \in graph$ **do**
  $p_i = \frac{k_i}{\sum_{j \in 1 \cdots N} k_j}$
  **if** $random() \leq p_i$ **then**
    $connect(newnode, node_i)$
  **end if**
**end for**

## 3.2 Watt-Strogatz model

The WS-model[17] use a different approach for describing the small-world networks formation, it exploits a different concepts related to observation of both random, regular, and small-world networks.
Considering the structure of small-word networks is possible to observe that they aren't neither regular nor random, so the main idea is to obtain a realistic network topology by moving from regular to random network by increasing randomness in some way. The algorithm used for network generation2 consists in two main steps: the creation of a regular lattice graph (with $K$ indicating the mean degree), and a successive random rewiring leaded by a probability of rewire $\beta$.

**Algorithm 2** WS network generation algorithm

{Create a graph with N nodes each connected to K neighbours, K/2 on each side}
$graph = createLatticeRing(nodes)$
{Perform a random rewiring}
**for** $i$ from 1 to $N$ **do**
  **for** $j$ from $i + 1$ to $N$ **do**
    **if** $random() \leq \beta$ **then**
      $connect(node_i, node_j)$
    **end if**
  **end for**
**end for**

This model generates networks with logarithmic average path length, a clustering coefficient which depends on $K$ (not on size of graph), but does not generate a realistic - scale-free - degree distribution.

### 3.3 Caveman model

Another interesting network model proposed in[16] consists in a refinement of the classical caveman model, which is based on a set of connected sub-graphs wired each other in a regular lattice as shown in figure 1.
Once generated the base caveman graph, further random rewiring will be performed in a distinct way w.r.t. WS-model.
This model use the *"s-metric"* as probability of rewiring: this metric aim to quantify the propensity of high-degree nodes to connect to other high-degree nodes, it is defined as $s(graph) = \sum_{i,j \in graph} k_i \cdot k_j$, where $k_i$ is the node-$i$ degree.
Whereas a probability is needed, this measure is used for normalizing single node's *"s-value"*, so probability of rewiring results in $p(i,j) = \frac{k_i \cdot k_j}{\sum_{u,v \in graph, u \neq v} k_u \cdot k_v}$ with $i, j \in graph$, $i \neq j$.

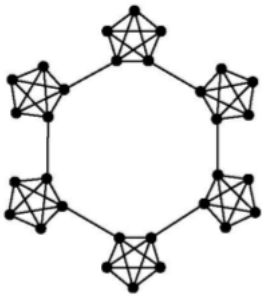

Figure 1: Caveman network model (initial) with $m = 5$ sub-groups in[16]

**Algorithm 3** Caveman enhanced generation algorithm

{Create a caveman graph with $m$ groups/communities}
$graph = createInitialCavemanRing(nodes, m)$
{Perform a random rewiring}
**for** $i$ from 1 to $N$ **do**
  **for** $j$ from $i+1$ to $N$ **do**
    $p_{i,j} = \frac{k_i \cdot k_j}{\sum_{u,v \in graph, u \neq v} k_u \cdot k_v}$
    **if** $random() \leq p_{i,j}$ **then**
      $connect(node_i, node_j)$
    **end if**
  **end for**
**end for**

This model can generate networks with a high clustering coefficient, relatively low average shortest path, and - as reported in[16, 4] - with sufficient network size, a degree distribution very similar to power law distribution.

# 4 Mobility Models

Mobility models are key aspect in simulations that involves mobile users, in fact running an experiment where movements of nodes should generate a realistic contact network dynamic require some non trivial considerations.
In order to reproduce credible contact networks we need to choose a mobility model which is able to simulate human mobility patterns and dynamics, for this reason we can't use *Brownian mobility models* or similar.
We focus on a community based model which exploit social network information5, however we point out here[4] for an useful review of several types of mobility models.

# 5 Community Based Mobility Model

In[10] is presented a particularly interesting mobility model which explicitly use social-network for generating mobile users traces and contact network dynamics.
The model preface a simulation space subdivided in distinct areas, these squares are initially populated by communities found in the social-network provided as input, one community per area.
At the same time each mobile user (node) is associated to a goal in an agent based fashion, and initially each node desires to reach the square where it is.
When a goal is reached, a new goal have to be chosen, and the concept of "social attractivity" of a place is introduced as follow: the place $S$ exerts a desirability to node-$i$ proportional to the relationships between $i$ and the other nodes that belong to that particular square, normalized by the total number of nodes placed in $S$.
More formally $A(S,i) = \frac{\sum_{j \in nodesIn(S)} \omega_{i,j}}{\sum_{j \in nodesIn(S)} j}$ where $S$ is the place, $i$ is the mobile user (node), $\omega$ is the relationship between $i$ and $j$ (can be the edge weight or $0, 1$ in case of un-weighted graph).
The new goal is then randomly chosen inside the square characterised by the highest desirability, so a node can decide to stay in the same place or move to a different one.
Experimental results in [10, 3.2.2] show that simulated inter-contacts times present a time distribution similar to real mobility traces. For this reason the community based mobility represent a good foundation for our experiment.

# 6 Forwarding in opportunistic networks

Opportunistic networks are highly dynamical network, connections are not durable like internet ones and contact network can be characterized by more than one connected components dynamics, even without a giant connected component at all. For this reason forwarding decision might be not so oblivious and using traditional forwarding schemes might lead to limitations in terms of both efficiency and effectiveness.
In this section we describe several algorithms that could be used for forwarding a message in a dynamic, Delay Tolerant Network (DTN), with particular regards to algorithms that exploit network topology informations.

## 6.1 Non social-based algorithms

Before talking about social-based algorithm we want perform a non exhaustive survey of canonical forwarding algorithm, that ones which does not exploit any network topology information. This kind of algorithms are a fundamental reference framework even during analysis of newer forwarding algorithms because they can provide a solid base for benchmarking. We can individuate three canonical forwarding algorithm:

**Wait** consists in merely holding the message until recipient is directed encountered. Pretty ineffective, but it provide a lower bound in term of both delivery cost and delivery ratio.

**Flood** used also in several routing protocols (eg.OSPF) for network image construction, consists in a flooding through the entire system. Represent the upper bound in term of both delivery cost and delivery ratio.

**MCP** *Multiple-Copy-multiple-hoP*, it's a mix of the previous approach. Each node can forward only a predefined number of copies for each message; each message has also a TTL measured in Hops (like common IP packets). It represent a pretty efficient and effective forwarding scheme for naive algorithms.

## 6.2 Social-based algorithms

Basic approaches introduced previously does not make any assumption on contact network so a newer set of forwarding algorithm have been investigated in the last decade.
Considering that humans will carry these mobile nodes enable us to make new assumptions that could be exploited in forwarding decisions, so we will perform a qualitative survey on forwarding algorithms which exploit this consideration in order to extract concepts and measures behind forwarding decision-making:

### 6.2.1 Label algorithms

This approach exploit the human dimension of the problem, in fact messages are forwarded only to encountered nodes with the same (or also similar) interest of the recipient. The concept of similarity of interest is important because typically similar tends to stay close, in other words this forwarding algorithm exploit

communities structures (eg. *k-cliques*) which are characteristic of small-world networks.

### 6.2.2 Rank algorithms

Here the focus is on node centrality, this is the key measure considered in forwarding decisions. For example in *Greedy Rank Algorithm* messages are forwarded only to neighbours that have higher centrality value, or also can be performed sort of climb for reaching nodes with maximum rank, and then a descent to the recipient[1], but identifying the maximum rank value in a distributed system is not cost-free and in general assuming the knowledge of the whole network structure or global values could be unrealistic, so this kind of algorithms typically use heuristics to estimate centrality values.

### 6.2.3 Bubble algorithms

Bubble algorithm[7] family basically play with both rank and labels in order to enhance efficiency in terms of minimizing in flight copies. The approach is pretty interesting due to it considers two level of ranking: a *global ranking* which coincide with measure of centrality w.r.t. the whole network, and a *local ranking* that consider centrality w.r.t. nodes in the same community. Successively it mixes these notions of rank with the community membership.
The following algorithm's sketch4 shows how these elements are combined.

**Algorithm 4** Bubble RAP forwarding algorithm

**for** $neighbor$ such that $neighbor \in neighborhood(node)$ **do**
  **if** $LabelOf(node) == LabelOf(recipient)$ **then**
    **if** $LabelOf(neighbor) == LabelOf(recipient)$ **and** $LocalRank(neighbor) > LocalRank(node)$ **then**
      $forwardMessageTo(neighbor)$
    **end if**
  **else**
    **if** $LabelOf(neighbor) == LabelOf(recipient)$ **or** $GlobalRank(neighbor) > GlobalRank(node)$ **then**
      $forwardMessageTo(neighbor)$
    **end if**
  **end if**
**end for**

In [7, 6.1,6.2] experimental results have shown that this forwarding strategy is effective like MCP (slightly less than flooding algorithm) but with a notable difference in term of delivery cost. In figure2 are reported the result of an experiment with two main communities.

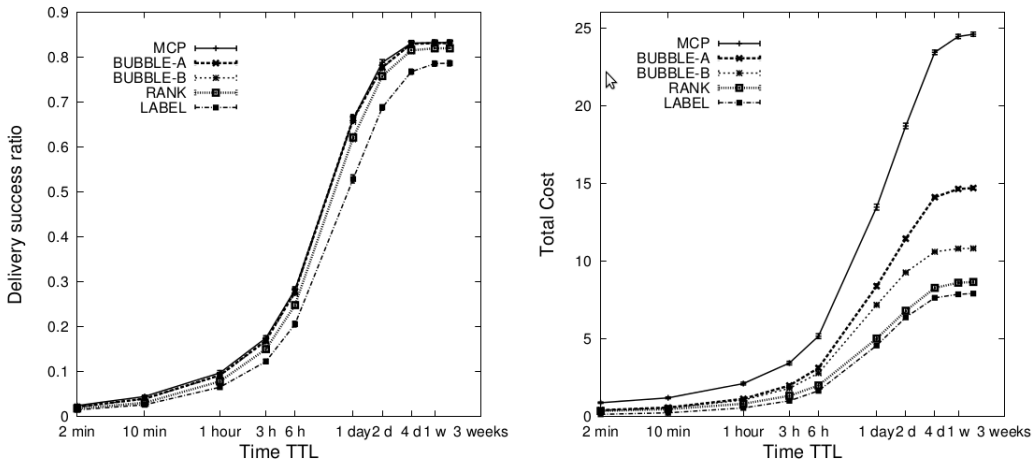


Figure 2: Comparison of several algorithms in two-community experiment based on *Cambridge dataset* data sets performed in[7, Figure 14 on SOCIALNETS Deliverable D2.3]

# 7 The Experiment

The goal of this section is to explain the experiment with all technical detail needed, from adopted models to results through scenarios and measures.
First of all experiment aims to measure efficiency of several forwarding algorithm in an infrastructure free scenario, where mobile nodes are able to interact with others only in a limited local area around them.
We measure forwarding algorithm performance in terms of:

$$\textit{Delivery ratio}: \frac{NumberOfMessagesDelivered}{NumberOfMessagesSent}$$

$$\textit{Delivery cost}: \frac{NumberOfInflightMessages}{NumberOfMessagesSent}$$

Delivery ratio could range from 0 to 1, and we are interested in the maximum delivery ratio value obtained through all simulation time. We care about maximum value in the all time span because we are simulating an opportunistic delay tolerant network. For delivery cost we consider it's maximum value (cost peek), it's average value and it's variance across the simulation time, and its correlation with delivery ratio.

## 7.1 Models and algorithms

Since experiment needs to simulate a realistic contact network dynamics, we based our simulation on the community based mobility model introduced in

section5 which is able to reproduce credible human-like interaction traces. In order to use this tool we had to generate a social-network to feed the mobility model algorithm.
For this reason we choose to adopt the caveman enhanced network model described in section3.3 where community structures are first class concept.
Also we implement a subset of the forwarding algorithm described in6, both social-based and non-social ones; in detail we ran experiment with the following forwarding algorithms:

- *Flood*, which is the most effective algorithm (but also the most expensive in terms of message copies)
- *Label*, which exploit node's interests and consequential community patterns.
- *Rank*, where nodes popularity lead forwarding decisions.
- *Bubble*, where both node's interest and popularity are exploited according to algorithm4.

## 7.2 Simulation platform

We decide to use a simulation platform that lessen the abstraction gap between the conceptual space of our experiment and the platform abstractions. In detail our experiment mainly involves two main concepts: mobile nodes and social networks.
Since we think supporting mobility simulation is generally harder than supporting social network simulations we choose to adopt the *Alchemist* simulation platform[13] because it has been explicitly designed to handle nodes mobility and contact networks.
A part of the class project work has been allocated in the extension of the Alchemist platform in order to:

- Integrate the concept of social network and enable simulation on arbitrary topologies.
- Design a mechanism which permit to define arbitrary network formation algorithm (aka auto-linking rule)[3].
- Support environment with multiple networking layers. For instance support an euclidean environment with mobile nodes that links each others based on their physical distance, and at the same time support the concept social neighbourhood, or in general another linking layer based on other - non euclidean - criteria.

With this extensions we are fully enabled to run the experiment described above, which exploits only a small part of the expressive power of this extension. In fact adding a social networking conceptual space to alchemist's domain may enable simulations on social network formation and dynamics by simulating contacts and interactions between mobile nodes in several environmental conditions.

[3]Alchemist use an auto-linking euclidean rule (based on euclidean distance) for determining when a mobile node has a neighbourhood relationship with another one

### 7.2.1 Experiment set-up

We set-up two experimental session with initial parameters inspired from [7, 6.2]:

- First one composed by 100 runs for each forwarding algorithm with 60 mobile nodes[4] distributed in 6 communities and 16 message to deliver (uniformly distributed across nodes population), each time with a randomized[5] initial configuration in terms of relationship between mobile users and consequential contact traces.
- Second one with the same characteristics of the previous except for the number of message to deliver: 100 messages per run.

In these experiments we set-up all messages at configuration time, so every sender start delivering from the beginning of the simulation. This configuration make sense if we assume that the stochastic process we are simulating is ergodic, under this assumption we might consider the simulation as a valid time sample of the behaviour of the system under the specified load (in terms of message to deliver in a time span). Further consideration on chosen initial parameters are argued in section 7.3.1.
In figure 3 and 4 are represented environment and nodes in the simulation demo running on Alchemist.

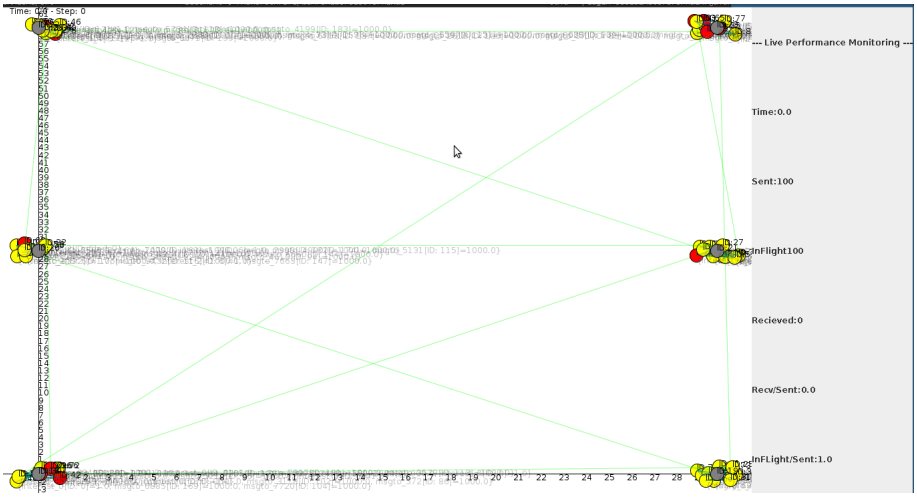


Figure 3: Simulation initial configuration. Green lines represent social relationship while blue (shorter) lines physical links.

[4]Datasets used for emulations in[7] provide traces of a number of mobile devices between 40 and 100 units, so system size is comparable

[5]We used the built-in Mersenne Twister algorithm provided by simulation platform

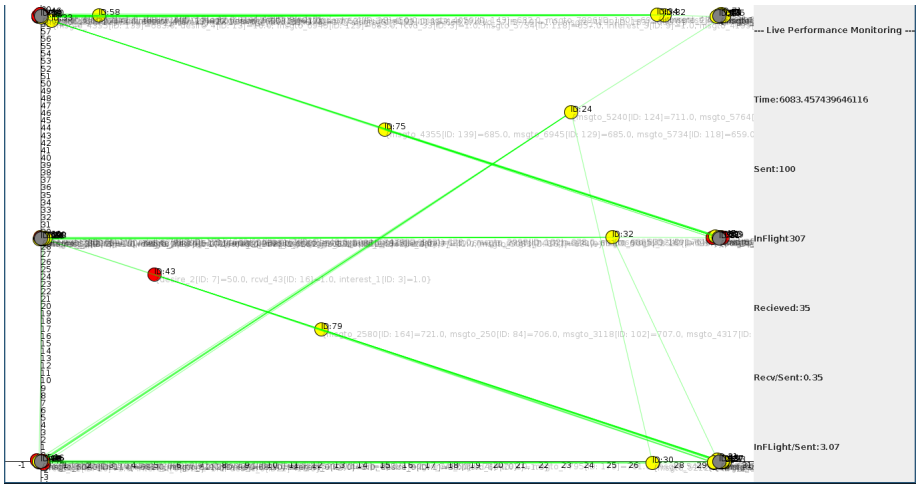


Figure 4: Running simulation (bubble)

### 7.2.2 Original Experiment

In honour of completeness we briefly report emulation parameters and measurements reported in[7, Table 3,6.2]. The reference experiment - based on *Reality data set* - involves 73 nodes subdivided in 8 communities (average size $\sim 9$) and a simulation time of 3 weeks. In figure 5 are shown delivery ratio and cost measured during simulations.

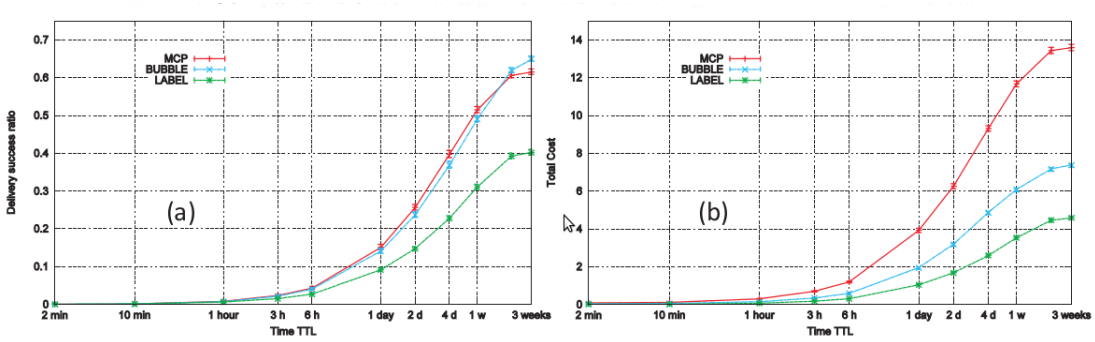


Figure 5: Comparison of several algorithm performance reported in [7] with *Reality data set*. Figures *(a)* and *(b)* shows delivery ratio and cost respectively

## 7.3 Results

### 7.3.1 Discussion on Results

Simulation results have shown similar motifs with respect to emulation performed in[7], we can observe that social based algorithm have lower delivery cost with respect to classical ones and bubble algorithm has higher delivery ratio with respect to other social based algorithms. See Fugure5 and Figure6 for a comparison of original experiment and our simulation results.
However absolute value differences exist if we compare our experimental sessions to the reference one: this discrepancy could be caused by the arbitrary load assumptions performed and the arbitrary time span considered.
In fact in[7, 6.2] time span considered is 3 week large ($1814400sec$), with a presumable node velocity of $1 \sim 1.5 \frac{m}{sec^{-1}}$, instead simulated experiments range within 7 hours ($\sim 25000sec$). So the 1000 message of the emulation are spread in a larger time span, with an average ratio of $5.5 \cdot 10^{-4} \frac{msgs}{secs}$, while the 100 messages of the simulation are cumulated in a smaller time interval, with average ratio of $4 \cdot 10^{-3} \frac{msgs}{secs}$. This mean a more stressed system.
However, as specified in7.2.1, we set-up the environment with all message ready to be sent, simulating the dynamics of the system under arbitrary load conditions.
Further simulation parameter tuning can be performed in order to more precisely replicate conditions in7.2.2: for example a finest set-up of nodes speed (and consequentially to the node movement reaction rate[6]) or the rate of the moving decision's reactions should lead to more accurate results. However obtained results are also pretty interesting because they reveal similar algorithms characteristics and led to the same efficiency conclusions.

### 7.3.2 Experimental Results

Collected data of each experiment's run have been aggregated per time-interval, in other words we made an average of each runs measurement in each time sub-interval (one point on these graphs represent the average value measured for all the runs within the time interval considered). In addition we performed some classical statistic on unaggregated data.
In Figure67 and Figure89 are represented delivery ratio and delivery cost of each algorithm during the simulation time (aggregated data).

[6] Alchemist is based on chemical metaphor, so reactions (conditions and actions) happens with arbitrary rate

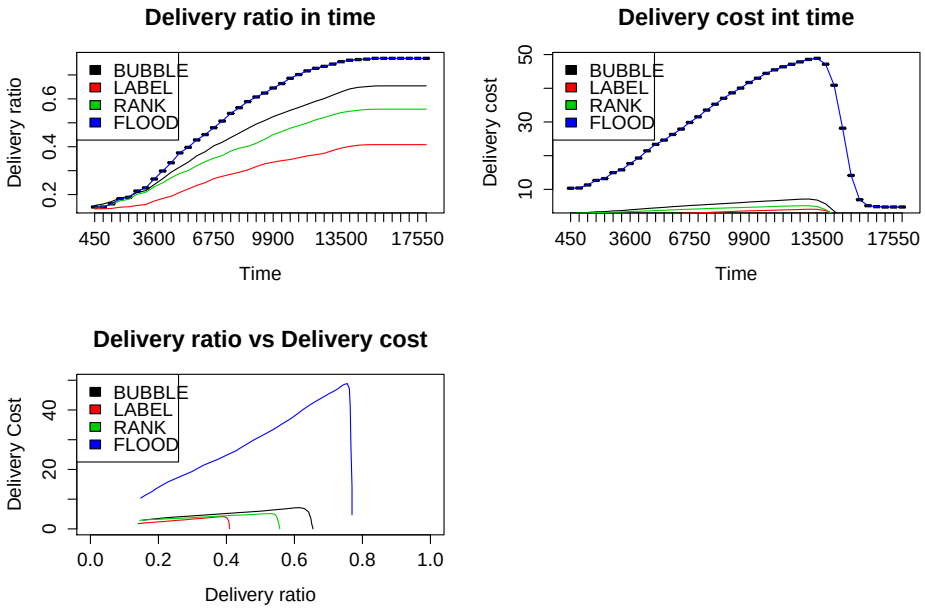


Figure 6: Experimental session n.1 aggregated data.
The descendent trend of delivery cost and delivery ratio vs. delivery cost is caused by message TTL expiration which begins around t=14000.

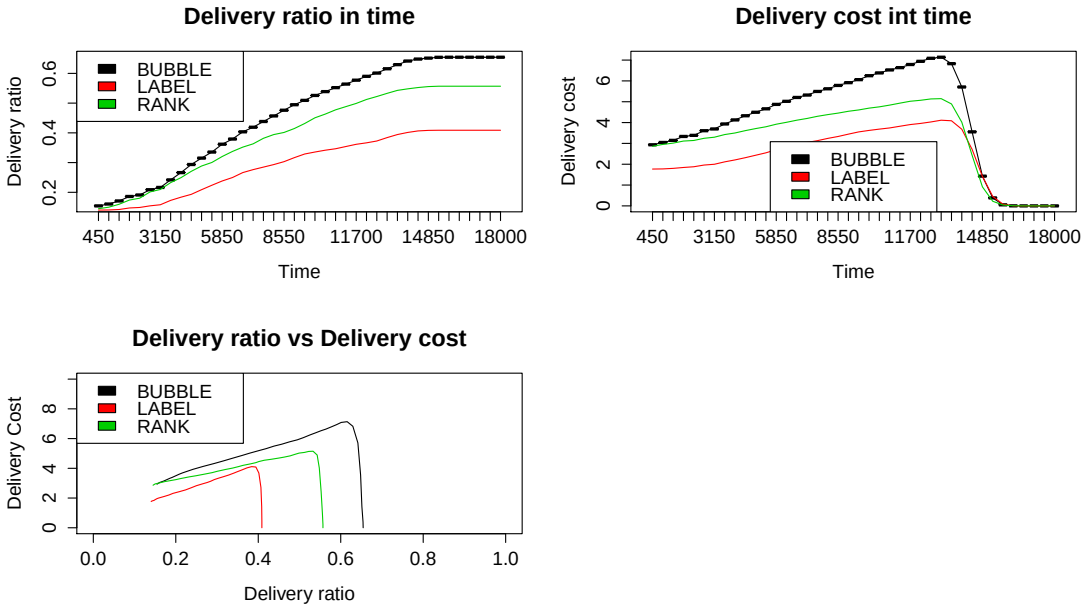


Figure 7: Experimental session n.1 aggregated data without flood algorithm.

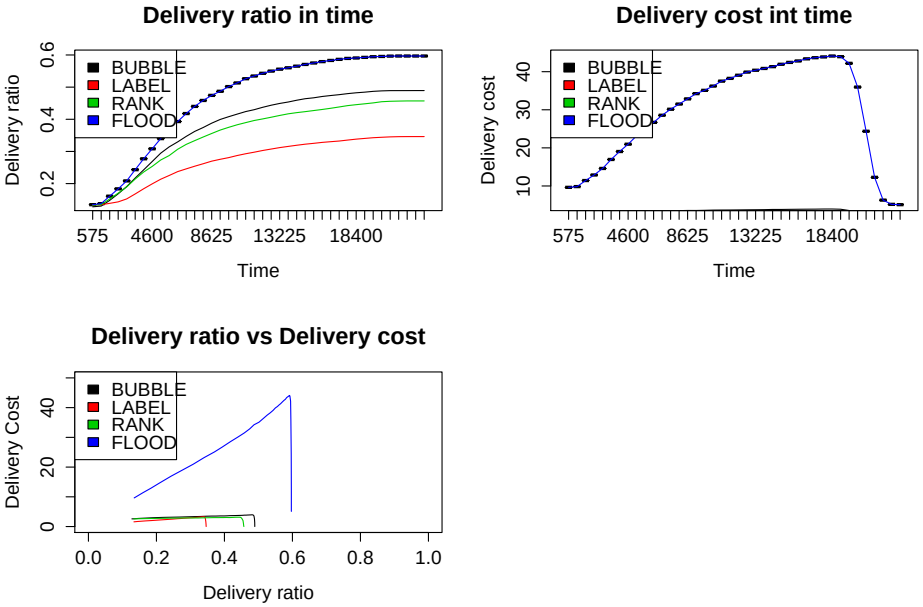


Figure 8: Experimental session n.2 aggregated data.

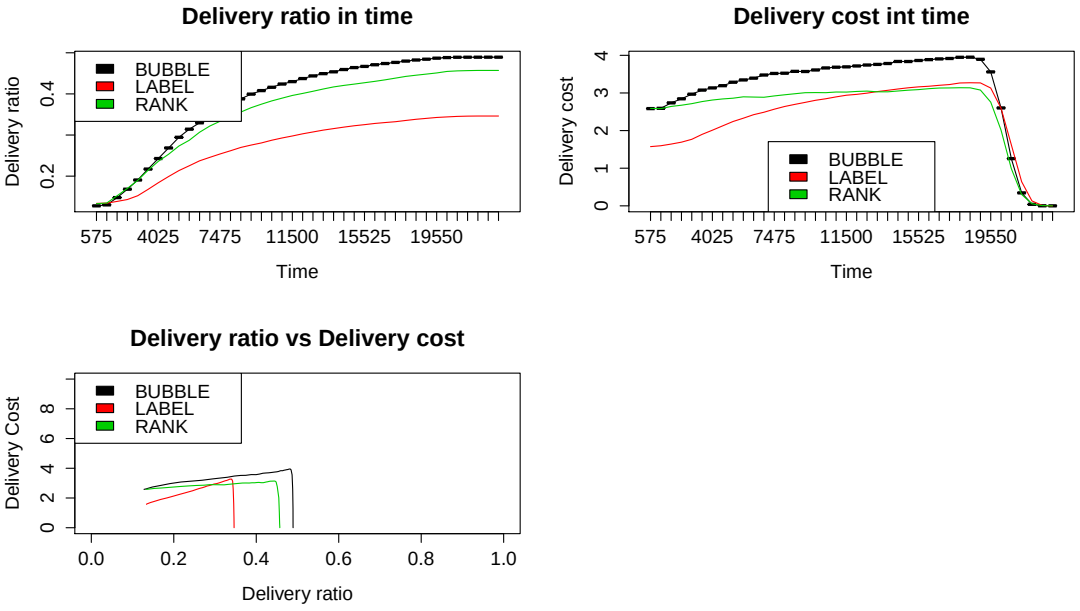


Figure 9: Experimental session n.2 aggregated data without flood algorithm.

While in Figure1011 are shown statistics about simulation session 1 and 2 subdivided per algorithm, and in Figure1213 are represented the histograms of delivery ratio measured in all the experiment's runs.

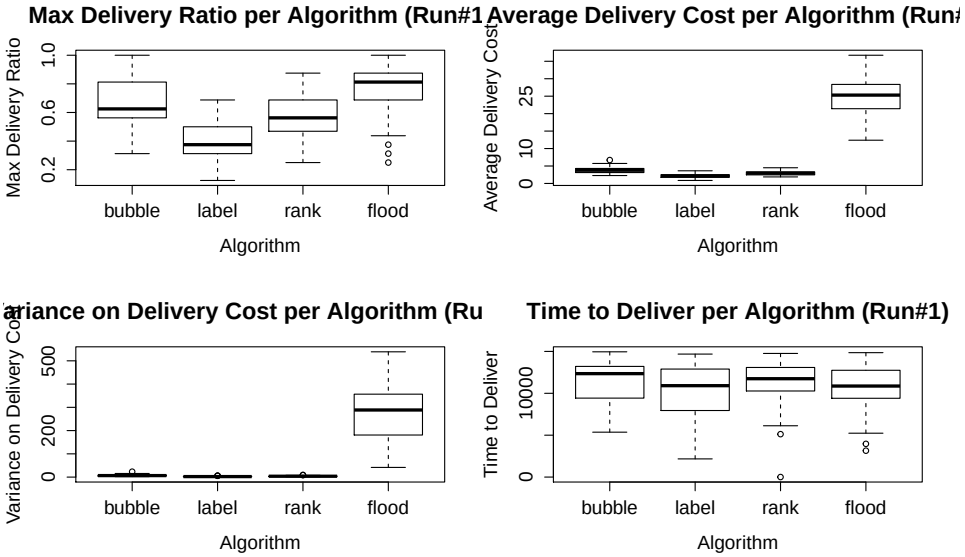


Figure 10: Experimental session n.1 statistics.

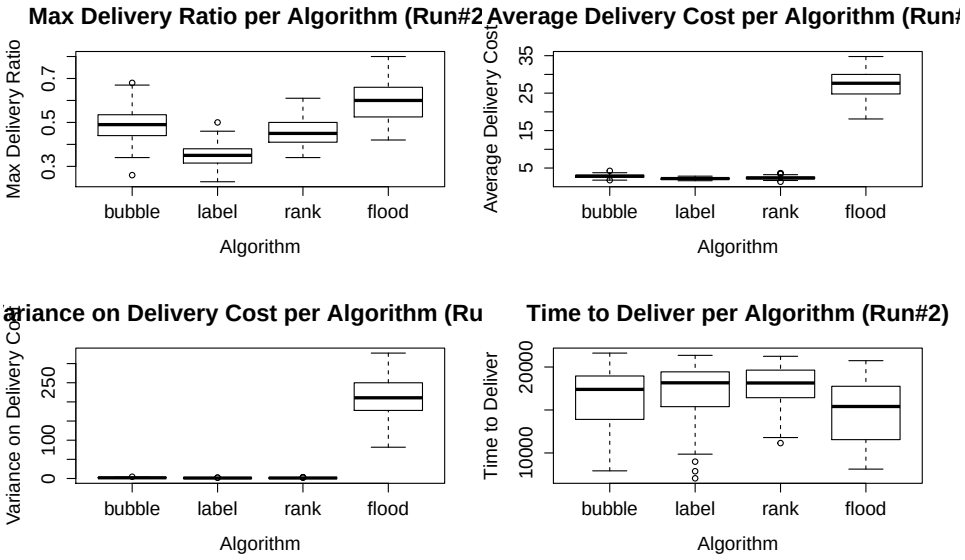


Figure 11: Experimental session n.2 statistics.

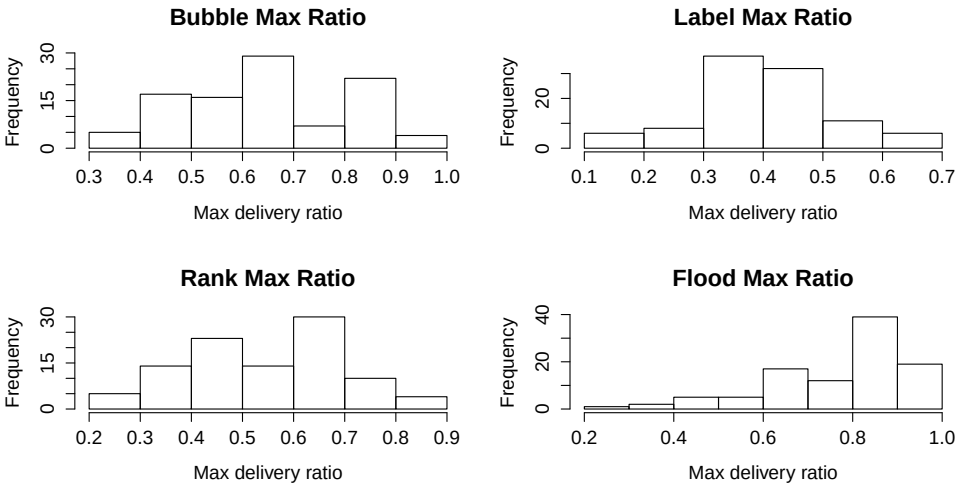


Figure 12: Experimental session n.1 delivery ratio distribution.

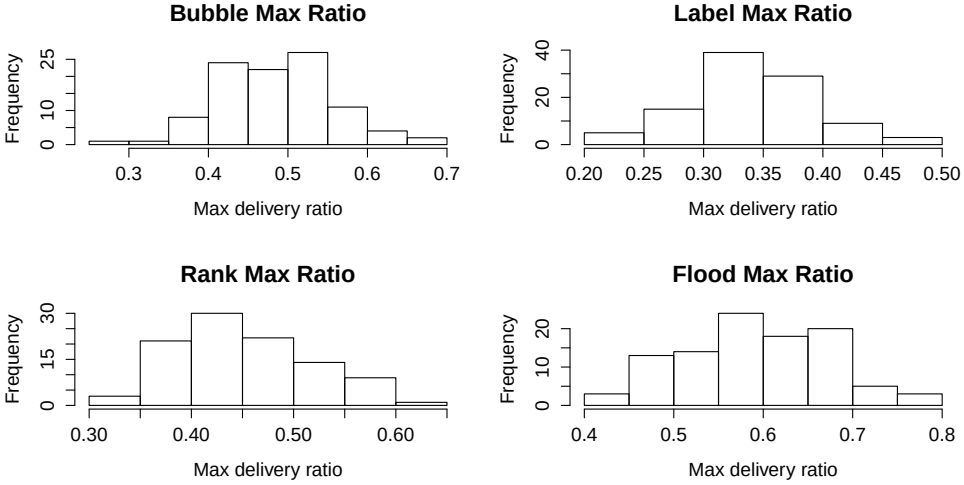


Figure 13: Experimental session n.2 delivery ratio distribution.

# 8 Conclusions

Despite the differences analyzed in 7.3, we can derive same conclusion about algorithm efficiency: in fact flood algorithm results to be the upper bound in term of efficacy, label and rank algorithms have both lower cost w.r.t flood and also lower efficacy, instead bubble algorithm presents good delivered message/-cost ratio, it also has comparable delivery delay with other algorithms.
The similarity of the outcome produced by our experiment w.r.t. the original - emulation based - one, could indicate that this kind of simulations are a valid approach even in social based forwarding performance evaluation. As told before, further parameter refinement could be performed in order to better replicate original experiment scenario, but we have to remember that even considering a more congested scenario7.3.1 we point out to the same conclusions about algorithms efficiency.
However, we can note that *Alchemist*'s extensions developed during this class project could also enable a new set of experimental scenarios: from social network formation to it's dynamics, maybe even the side effect on users mobility patterns, and forwarding algorithm evaluation on arbitrary conditions too.
Naturally the limited amount of time available for class project are not enough for investigating the wide range of opportunities that the social network and mobility abstractions enable, so this class project could be considered as our first step into these topics.

# References


[1] Lada A. Adamic, Rajan M. Lukose, Amit R. Puniyani, and Bernardo A. Huberman. Search in power-law networks. *Phys. Rev. E*, 64:046135, Sep 2001.

[2] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, October 1999.

[3] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D. Hwang. Complex networks: Structure and dynamics. *Physics Reports*, 424(4-5):175–308, February 2006.

[4] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing (WCMC): Special issue on mobile had hoc networking: research, trends and applications*, 2:483–502, 2002.

[5] A. Chaintreau, J. Crowcroft P. Hui, R. Gass C. Diot, and J. Scott. Impact of human mobility on the design of opportunistic forwarding algorithms. In *INFOCOM*, 2006.

[6] H. Ebel, L.-I. Mielsch, and S. Bornholdt. Scale-free topology of e-mail networks. Technical report, University of Kiel, February 2002.

[7] Pan Hui, J. Crowcroft, and E. Yoneki. Bubble rap: Social-based forwarding in delay-tolerant networks. *Mobile Computing, IEEE Transactions on*, 10(11):1576 –1589, nov. 2011.

[8] T. Karagiannis, J.-Y. Le Boudec, and M. Vojnovi. Power law and exponential decay of inter contact times between mobile devices. In *ACM MobiCom '07*, 2007.

[9] J. Leguay, A. Lindgren, J. Scott, T. Friedman, and J. Crowcroft. Opportunistic content distribution in an urban setting. In *ACM CHANTS*, page 205–212, 2006.

[10] M. Musolesi and C. Mascolo. A community based mobility model for ad hoc network research. In *Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality*, REALMAN '06, pages 31–38, New York, NY, USA, 2006. ACM.

[11] M. Musolesi and C. Mascolo. A community based mobility model for ad hoc network research. In *Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality*, REALMAN '06, pages 31–38, New York, NY, USA, 2006. ACM.

[12] M. Newman. *Networks: An Introduction*. Oxford University Press, Inc., New York, NY, USA, 2010.

[13] D. Pianini, S. Montagna, and M. Viroli. Chemical-oriented simulation of computational systems with alchemist. *Journal of Simulation*, January 2013. Advance online publication.

[14] SocialNets. Socialnets: Social networking for pervasive adaptation.

[15] M. Szell, R. Lambiotte, and S. Thurner. Multirelational organization of large-scale social networks in an online world. -, July 2010.

[16] D. J. Watts and S. H. Strogatz. Collective dynamics of'small-world'networks. *Nature*, 393(6684):409–10, 1998.

[17] T. Yihjia, H. Ping-Nan, and L. Ching-Chang. A social network model based on caveman network. In *Communications and Networking in China, 2006. ChinaCom '06. First International Conference on*, pages 1–6, Oct.

# Appendices

## A Alchemist Repository

Alchemist main line is reachable at `https://bitbucket.org/danysk/alchemist`, while the extensions developed during class project ( *incarnation-socialnets* and *plugin-socialnets*) are publicly available at `https://bitbucket.org/lucam/alchemist-socialnets`.

## B Collected Data

Experimental session data are also available at `TBD`.

## C Data Processing

Listing 1: R Script used for data processing and visualization

```
GlobalSimStats<-function(path='.', pattern='[0-9]*\\.csv') {
  #   Computes maximum delivery ratio, average delivery cost,
  #   variance of delivery cost, and time to deliver on multiple
  #   simulations run output
  #
  # Args:
  #   path: the base directory where data files are stored
  #   pattern: regex that match a valid csv data file name
  #
  # Returns:
  #   A data frame which contains maximum delivery ratio, average
  #   delivery cost, variance of delivery cost, and time to deliver for
  #   each data file (one row per file/run)
  #
  files<-list.files(path=path, pattern=pattern)
  slist<-data.frame()
  for (f in files ){
    s<-SimStats(paste(path,f,sep=""))
    slist<-rbind(slist,s)  # Pretty
  }
  return(slist)
}
SimStats<-function(file){
  #   Computes maximum delivery ratio, average delivery cost,
  #   variance of delivery cost, and time to deliver on a
  #   simulations run output
  #
  # Args:
  #   file: file where simulation data are stored
  #
  # Returns:
  #   A data frame which contains maximum delivery ratio, average
  #   delivery cost, variance of delivery cost, and time to deliver for
  #   the specified data file (just one row)
  #
  d<-read.csv(file)
  tmp<-max(max(d$delivery_ratio))
  return(data.frame(max_delivery_ratio=tmp,
            avg_delivery_cost=mean(d$delivery_cost),
            var_delivery_cost=var(d$delivery_cost),
            time_to_deliver=d[d$delivery_ratio==tmp,][1,]$time))
}
PlotStats<-function(){
  # Graph global simulation statistics
  #
  par(mfrow=c(2,2))
  boxplot( g1$max_delivery_ratio~g1$alg,
            ylab="Max_Delivery_Ratio",xlab="Algorithm",
            main="Max_Delivery_Ratio_per_Algorithm_(Run#1)" )
  boxplot( g1$avg_delivery_cost~g1$alg,
            ylab="Average_Delivery_Cost",xlab="Algorithm",
            main="Average_Delivery_Cost_per_Algorithm_(Run#1)" )
  boxplot( g1$var_delivery_cost~g1$alg,
            ylab="Variance_on_Delivery_Cost",xlab="Algorithm",
            main="Variance_on_Delivery_Cost_per_Algorithm_(Run#1)" )
  boxplot( g1$time_to_deliver~g1$alg,
            ylab="Time_to_Deliver",xlab="Algorithm",
            main="Time_to_Deliver_per_Algorithm_(Run#1)" )
  par(mfrow=c(2,2))
  boxplot( g2$max_delivery_ratio~g2$alg,
            ylab="Max_Delivery_Ratio",xlab="Algorithm",
```

```
62                main="Max Delivery Ratio per Algorithm (Run#2)" )
63      boxplot( g2$avg_delivery_cost~g2$alg,
64                ylab="Average Delivery Cost",xlab="Algorithm",
65                main="Average Delivery Cost per Algorithm (Run#2)" )
66      boxplot( g2$var_delivery_cost~g2$alg,
67                ylab="Variance on Delivery Cost",xlab="Algorithm",
68                main="Variance on Delivery Cost per Algorithm (Run#2)" )
69      boxplot( g2$time_to_deliver~g2$alg,
70                ylab="Time to Deliver",xlab="Algorithm",
71                main="Time to Deliver per Algorithm (Run#2)" )
72    }
73    PlotAggregated<-function(){
74      #    Plot aggregated data for the two experimental sessions
75      #
76      PlotGraphs(a1b,a1l,a1r,a1f)
77      PlotGraphs(a2b,a2l,a2r,a2f)
78    }
79    PlotGraphs<-function(ab,al,ar,af){
80      #    Plot aggregated data
81      #
82      par(mfrow=c(2,2))
83      PlotDelRatio(ab,al,ar,af)
84      PlotDelCost(ab,al,ar,af)
85      PlotDelRatioVsDelCost(ab,al,ar,af)
86    }
87    PlotDelRatioVsDelCost<-function(ab,al,ar,af){
88      #    Plot delivery ratio vs delivery cost
89      #
90      plot(y=range(0,10),x=range(0.0,1.0),
91           type="n", xlab="Delivery_ratio", ylab="Delivery_Cost",
92           main="Delivery_ratio_vs_Delivery_cost")
93      #d=ab[1:length((which(ab$delivery_ratio<max(ab$delivery_ratio))==TRUE)),]
94      d=ab
95      lines(y=d$delivery_cost, x=d$delivery_ratio,col=1,pch=1, type="l")
96      #d=al[1:length((which(al$delivery_ratio<max(al$delivery_ratio))==TRUE)),]
97      d=al
98      lines(y=d$delivery_cost, x=d$delivery_ratio,col=2,pch=2, type="l")
99      #d=ar[1:length((which(ar$delivery_ratio<max(ar$delivery_ratio))==TRUE)),]
100     d=ar
101     lines(y=d$delivery_cost, x=d$delivery_ratio,col=3,pch=3, type="l")
102     #d=af[1:length((which(af$delivery_ratio<max(af$delivery_ratio))==TRUE)),]
103     #d=af
104     lines(y=d$delivery_cost, x=d$delivery_ratio,col=4,pch=4, type="l")
105     legend("topleft", c("BUBBLE","LABEL","RANK","FLOOD"),fill=c(1,2,3,4))
106   }
107   PlotDelRatio<-function(ab,al,ar,af){
108     #    Plot delivery ratio
109     #
110     plot(x=af$time,y=af$delivery_ratio,
111          type="n", ylab="Delivery_ratio", xlab="Time",
112          main="Delivery_ratio_in_time")
113     lines(ab$time, y=ab$delivery_ratio,col=1,pch=1, type="l")
114     lines(al$time, y=al$delivery_ratio,col=2,pch=2, type="l")
115     lines(ar$time, y=ar$delivery_ratio,col=3,pch=3, type="l")
116     lines(af$time, y=af$delivery_ratio,col=4,pch=4, type="l")
117     legend("topleft", c("BUBBLE","LABEL","RANK","FLOOD"),fill=c(1,2,3,4))
118   }
119   PlotDelCost<-function(ab,al,ar,af){
120     #    Plot delivery cost
121     #
122     plot(x=af$time,y=af$delivery_cost,
123          type="n", ylab="Delivery_cost", xlab="Time",
124          main="Delivery_cost_int_time")
125     lines(ab$time, y=ab$delivery_cost,col=1,pch=1, type="l")
126     lines(al$time, y=al$delivery_cost,col=2,pch=2, type="l")
127     lines(ar$time, y=ar$delivery_cost,col=3,pch=3, type="l")
128     lines(af$time, y=af$delivery_cost,col=4,pch=4, type="l")
129     legend("topleft", c("BUBBLE","LABEL","RANK","FLOOD"),fill=c(1,2,3,4))
130
131   }
132   AggregateData<-function(path='.', pattern='[0-9]*\\.csv', timeslices=6,maxinterval
          =20000,mininterval=0){
133     #    Computes average delivery cost and delivery ratio aggregated by time slice
134     #    for each simulation data files. Then calculate the average of theese values,
135     #    aggregated by time slice.
136     #
137     # Args:
138     #    path: the base directory where data files are stored
139     #    pattern: regex that match a valid csv data file name
140     #    timeslices: number of slices
141     #    maxinterval: upper bound of the considered time interval
142     #    mininterval: lower bound of the considered time interval
143     #
144     # Returns:
145     #    A data frame which contains avg delivery ratio and average
146     #    delivery cost per time slice
147     #
148     files<-list.files(path=path, pattern=pattern)
149     slist<-data.frame()
150     for (f in files ){
151       s<-AggregateSim(file=paste(path,f,sep=""),timeslices=timeslices,maxinterval=
              maxinterval,mininterval=mininterval)
152       slist<-rbind(slist,s)  # Pretty inefficient
153     }
154     t<-unique(slist$time)
155     dr<-tapply(slist$delivery_ratio,slist$time,mean)
156     dc<-tapply(slist$delivery_cost,slist$time,mean)
157     drv<-tapply(slist$delivery_ratio_var,slist$time,mean)
158     dcv<-tapply(slist$delivery_cost_var,slist$time,mean)
```

```
    return(data.frame(time=t,delivery_ratio=dr,delivery_cost=dc,
                      delivery_ratio_var=drv,delivery_cost_var=dcv))
}
AggregateSim<-function(file, timeslices=6, maxinterval=20000,mininterval=0){
  #    Computes average delivery cost and delivery ratio aggregated by time slice
  #
  # Args:
  #    file: file where simulation data are stored
  #    time.cut: time intervals which are used for aggregation
  #    timeslices: number of slices
  #    maxinterval: upper bound of the considered time interval
  #    mininterval: lower bound of the considered time interval
  #
  # Returns:
  #    A data frame which contains avg delivery ratio and average
  #    delivery cost per time slice
  #
  d<-read.csv(file)
  tround= round(maxinterval/timeslices)
  brseq<-seq(mininterval, maxinterval, by=tround)
  timecut=cut(d$time,breaks=brseq,labels=sprintf("%.0f",brseq[-1]))
  agr<-aggregate(d$delivery_ratio, list(timecut), mean)
  agrv<-aggregate(d$delivery_ratio, list(timecut), var)
  agc<-aggregate(d$delivery_cost, list(timecut), mean)
  agcv<-aggregate(d$delivery_cost, list(timecut), var)
  return(data.frame(time=agr$Group.1,delivery_ratio=agr$x,delivery_cost=agc$x,
                    delivery_ratio_var=agrv$x,delivery_cost_var=agcv$x))
}
# Load Simulation Data
if ( !exists('loaded') || !loaded){
  # Reading simulations data
  g1b<-GlobalSimStats(path='60_6_16/bubble/')
  g1l<-GlobalSimStats(path='60_6_16/label/')
  g1r<-GlobalSimStats(path='60_6_16/rank/')
  g1f<-GlobalSimStats(path='60_6_16/flood/')
  g2b<-GlobalSimStats(path='60_6_100/bubble/')
  g2l<-GlobalSimStats(path='60_6_100/label/')
  g2r<-GlobalSimStats(path='60_6_100/rank/')
  g2f<-GlobalSimStats(path='60_6_100/flood/')
  # Prepare aggregated data
  timesl<-40
  maxint1<-18000
  a1b<-AggregateData(path='60_6_16/bubble/',timeslice=timesl,maxinterval=maxint1)
  a1l<-AggregateData(path='60_6_16/label/',timeslice=timesl,maxinterval=maxint1)
  a1r<-AggregateData(path='60_6_16/rank/',timeslice=timesl,maxinterval=maxint1)
  a1f<-AggregateData(path='60_6_16/flood/',timeslice=timesl,maxinterval=maxint1)
  maxint2<-23000
  a2b<-AggregateData(path='60_6_100/bubble/',timeslice=timesl,maxinterval=maxint2)
  a2l<-AggregateData(path='60_6_100/label/',timeslice=timesl,maxinterval=maxint2)
  a2r<-AggregateData(path='60_6_100/rank/',timeslice=timesl,maxinterval=maxint2)
  a2f<-AggregateData(path='60_6_100/flood/',timeslice=timesl,maxinterval=maxint2)
  # Global Simulation Statistics Data
  g1<-data.frame()
  g1<-rbind(g1, data.frame(max_delivery_ratio=g1b$max_delivery_ratio,
                           avg_delivery_cost=g1b$avg_delivery_cost,
                           var_delivery_cost=g1b$var_delivery_cost,
                           time_to_deliver=g1b$time_to_deliver,
                           alg='bubble'))
  g1<-rbind(g1, data.frame(max_delivery_ratio=g1l$max_delivery_ratio,
                           avg_delivery_cost=g1l$avg_delivery_cost,
                           var_delivery_cost=g1l$var_delivery_cost,
                           time_to_deliver=g1l$time_to_deliver,
                           alg='label'))
  g1<-rbind(g1, data.frame(max_delivery_ratio=g1r$max_delivery_ratio,
                           avg_delivery_cost=g1r$avg_delivery_cost,
                           var_delivery_cost=g1r$var_delivery_cost,
                           time_to_deliver=g1r$time_to_deliver,
                           alg='rank'))
  g1<-rbind(g1, data.frame(max_delivery_ratio=g1f$max_delivery_ratio,
                           avg_delivery_cost=g1f$avg_delivery_cost,
                           var_delivery_cost=g1f$var_delivery_cost,
                           time_to_deliver=g1f$time_to_deliver,
                           alg='flood'))
  g2<-data.frame()
  g2<-rbind(g2, data.frame(max_delivery_ratio=g2b$max_delivery_ratio,
                           avg_delivery_cost=g2b$avg_delivery_cost,
                           var_delivery_cost=g2b$var_delivery_cost,
                           time_to_deliver=g2b$time_to_deliver,
                           alg='bubble'))
  g2<-rbind(g2, data.frame(max_delivery_ratio=g2l$max_delivery_ratio,
                           avg_delivery_cost=g2l$avg_delivery_cost,
                           var_delivery_cost=g2l$var_delivery_cost,
                           time_to_deliver=g2l$time_to_deliver,
                           alg='label'))
  g2<-rbind(g2, data.frame(max_delivery_ratio=g2r$max_delivery_ratio,
                           avg_delivery_cost=g2r$avg_delivery_cost,
                           var_delivery_cost=g2r$var_delivery_cost,
                           time_to_deliver=g2r$time_to_deliver,
                           alg='rank'))
  g2<-rbind(g2, data.frame(max_delivery_ratio=g2f$max_delivery_ratio,
                           avg_delivery_cost=g2f$avg_delivery_cost,
                           var_delivery_cost=g2f$var_delivery_cost,
                           time_to_deliver=g2f$time_to_deliver,
                           alg='flood'))
  loaded<-TRUE
}
```