Modeling and Analysis of Dynamic Assignment in Scale-Free Networks: An Agent-based Simulation Exercise

Yu Teng, Nan Kong*

Weldon School of Biomedical Engineering, Purdue University

206 S. Martin Jischke Dr., West Lafayette, IN 47906

ABSTRACT

This paper addresses stochastic dynamic assignment of perishable goods in a scale-free network. We develop an agent-based simulation model in which the supplies and demands for a single perishable commodity are generated at each node (agent) in the network. We match supplies and demands with several pre-specified assignment rules and given various commodity decay durations. A utility is associated with each assignment based on the spatial proximity between the supplier and the recipient. We assess various rules in terms of overall network utility. Our computational results show that 1) it is beneficial to use hybrid assignment rules that integrate the spirit of system-wide assignment and that of local preferred assignment; 2) it may be beneficial to allow agent behavior changes with certain degree of autonomy.

Keywords: agent-based modeling and simulation, dynamic assignment, resource allocation, scale-free network, Repast Symphony.

INTRODUCTION

The assignment problem is one of the fundamental combinatorial optimization problems. It is stated, in the most general form, as follows. There are a number of resources and a number of tasks. Any resource is assigned to perform any task. Each resultant resource-task assignment generates some benefit. The objective is to exactly assign each resource to perform one task in such a way that the total benefit can be maximized. For a comprehensive treatise on assignment problems, we refer the interest readers to Rainer *et al* (2009).

Since most of the real-world assignment problems need to deal with the dynamic nature of the system, there has been considerable recent interest in the problem of dynamically assigning resources (container, vehicle, or organ) to tasks (load, passenger, or patient) over time. The dynamic assignment problem is a fundamental problem in routing and scheduling. It arises in a number of application domains. In the field of freight transportation, truckload motor carriers, railroads, and shipping companies all have to manage the fleets of containers that move one load at a time, with orders arriving continuously over time (Powell, 1996). It is common to assume that the arrival of orders is random (e.g., known only through a probability distribution). This key feature is also

prevalent in the passenger arena (Yang *et al*, 2003) and many applications in telecommunication (Everitt and Manfield, 1989).

The dynamic assignment problem provides a natural framework for modeling the dynamic information processes and comparing myopic models with those that exploit distributional information about the future (Spivey and Powell, 2004). The dynamic assignment problem offers considerable richness compared to its static counterpart.

In this paper, we apply agent-based discrete-event simulation to study a generic stochastic dynamic assignment problem defined on randomly generated scale-free networks. Our purpose is to investigate the effect of alternative spatially based assignment rules that are commonly implemented. We focus on problems where each task is completed with the assignment of one resource. We assume that resources and tasks are each characterized by a set of possibly unique attributes, and the benefit generated by an assignment will depend on the attributes of the resource and the task. It is worth noting that the studied problem is often too complex to obtain analytical solutions, especially for those defined on large complicated networks.

There have been several simulation works for dynamic assignment, most of which appears in traffic control (e.g., Florian *et al*, 2001; Mahmassani, 2004). Many of these works focus on improving real-time traffic management for specific transportation networks. We, on the other hand, compare various assignment rules in a more generic scale-free network. Furthermore, the previous work rarely studies the evolution of agent behaviors even though behaviors are sometime modeled probabilistically as simulation inputs. We, on the other hand, attempt to study the effect of agents changing their assignment behaviors with two well crafted experiments. To the best of our knowledge, our work is among the first to apply agent-based simulation to the dynamic assignment problem.

Agent-based modeling and simulation (ABMS) provides a promising approach to evaluate alternative assignment rules. It can model a system as a collection of autonomous decision-making entities called agents (Bonabeau, 2002). Based on a set of rules, each agent individually assesses its situation, makes decisions, and executes various behaviors. As a computational modeling technique, ABMS has been applied in a variety of application domains.

The remainder of this paper is organized as follows. In Section 2, we state the dynamic assignment problem more formally and use organ allocation as a motivating example. In Section 3, we introduce agent-based simulation and present the conceptual design of our model. In Section 4, we describe the implementation of the simulation model in Repast Symphony, a free and open source agent-based modeling toolkit. In Section 5, we

report our computational experiments. In section 6, we conclude the paper and point out future research directions.

PROBLEM STATEMENT

Given a scale-free network $G = (V, E)$. For each node $i \in V$, we consider two independent stochastic arrival processes for two types of entities (called types A and B). We in this work assume that the two arrival processes are identical between the two entity types and each arrival process is identical across all nodes in V. We associate a time tag t with each generated entity. For each entity either of type A or B, we assign two attributes: location and life span.

Once a type A entity a becomes available, we assign it to a type B entity b based on some assignment rule. For each assignment (a, b), we assign a benefit based on the spatial relationship between the two nodes, denoted by $u(a, b)$. Along a pre-determined time horizon, a number of assignments are made and the overall benefit is accumulated throughout the network. The goal is to maximize the cumulative benefit of the overall system. In Appendix A, we present the mathematical model of the basic stochastic dynamic assignment problem. The optimization problem is, of course, computationally intractable. Instead, we apply simulation to evaluate alternative assignment rules.

A Motivating Example

In this subsection we use an example in organ allocation to motivate the idea of dynamic assignment. Deceased organ transplantation is the only effective therapy for almost all patients with diseases that cause organ dysfunction. Unfortunately, it is hindered in the United States by donor scarcity. Furthermore, many types of organs (e.g., livers, hearts, and lungs) are often underutilized due to various clinical factors. Due to these reasons, organ transplantation and allocation has been a contentious issue for decades. Therefore, a major concern is the efficient allocation of deceased organs.

To facilitate organ allocation, the United Network for Organ Sharing (UNOS) (*www.unos.org*) was established to operate the organ transplantation and allocation system in the U.S. Currently, there are nearly 60 local organ procurement organizations (OPOs) that are responsible for matching organs with patients.

It is clear that organ allocation policy improvement can be formulated as a stochastic dynamic assignment problem. Donors and patients are regarded as types A and B entities. The two arrival processes are stochastic. Each entity is associated with a number of attributes, e.g., location and blood type for both donors and patients,

acceptable life span for donors, etc. The transplant efficiency is dependent upon the donor-patient pair, for which an important factor is the distance between the donor and the patient.

There have been several simulation models that are used to examine how alternative allocation policies can affect system outcomes (Taranto *et al*, 2000; Shechter *et al*, 2005; van den Hout *et al*, 2003). However, to the best of our knowledge, no model is agent based. It prohibits us from understanding the changes of the practices in OPOs and policies in UNOS. We in this paper study a generic dynamic assignment problem. With valuable insight, we will focus on organ allocation policy in future research.

Introduction to Scale-free Network

Scale-free network refers to a network that has a power-law distribution of number of links connecting to a node, i.e., a majority of nodes have less-than-average links while a small fraction of nodes are connected to many other nodes. The power-law distribution can be expressed as:

$$p(k) \sim k^{-\alpha},$$

which means that the probability of a randomly selected node having k links (degree k) decays exponentially with respect to k, where the rate of decay is determined by the exponent α, typically in the range $2<\alpha<3$. The term *scale-free* is coined by Barabási and Albert (1999) when the authors study the topology of the World Wide Web. Many other real-world networks are also scale-free, e.g., the citation network, the protein regulatory networks, and the sexual relationship networks. Barabási and Albert (1999) propose a generation model for scale-free networks (see Appendix B) which captures the network dynamics. The key concepts in this model are *growth* and *preferential attachment* (the likelihood of a new node connecting to an existing node depends on the degree of the existing node). One of the interesting results caused by the topology of the scale-free network is that it has a high error tolerance; however the high tolerance comes at a price of high vulnerability to attacks (Albert *et al*, 2000).

Assignment Rules

In this paper we consider two basic assignment rules as follows. At assignment cycle $t = 1,\ldots,T$, and at node $i \in V$, we rank all type A entities and type B entities separately based on their arrival times. The one with an earlier arrival time is given higher priority.

R0: A type A entity is assigned to a type B entity *at any node* if the type B entity becomes known and available for the assignment.

R1: A type A entity is assigned to a type B entity *at the same node* if the type B entity becomes known and available for the assignment. If no type B entities are known or no one is available for assignment, the type A entity is assigned to a type B entity *at any other node* if the type B entity becomes known and available for the assignment.

Note that if there is a tie between two type B entities for assignment, we break the tie randomly. It is clear that the assignment rule R0 provides a fair assignment scheme in the entire network. It assigns a type A entity to a type B entity regardless of its location. The assignment rule R1 takes local preference into consideration. It grants the type B entities at the same node higher priority than type B entities at other nodes. In some sense, rule R1 complicates rule R0 by granting the node itself higher priority in the assignment.

We further complicate the assignment scheme. For each node, we prioritize the nodes in the network for assignment. We give the node itself the highest priority, then for a subset of nodes the second highest priority, and finally all other nodes in the network. One interesting case is to include the neighbors of the node given the topology of the network. Hence, we have the following rule.

R2: A type A entity is assigned to a type B entity at the same node if the type B entity becomes known and available for the assignment. If no type B entities are known or no one is available for assignment, the type A entity is assigned to a type B entity at one of the nodes that is closest to the node where the type A entity is known. If no type B entities at these nodes are known or no one is available for assignment, the type A entity is assigned to a type B entity at one of the nodes that are two links apart from the node where the type A entity is known. Following the same logic, the type A entity is assigned sequentially to the nodes that are farther away from the node where the type A entity is known.

It is worth noting that rule R2 can be regarded as a multi-tier (hierarchical) assignment rule. A generalization is that at each tier, we select a subset of nodes to consider. This subset is not necessary to be all nodes that are of equal shortest path distance to the node where the type A entity becomes known.

Here we use the development of organ allocation policy as a real-world example. With the understanding of the existing inequality and inefficiency in the organ transplantation and allocation network, UNOS has attempted to establish true “national list” with which patients are given priority solely based on their medical needs. However, many OPOs object the principle of *national sharing*, especially those net donor organ suppliers. They prefer retaining organs that are procured within the OPOs for the local recipient candidates before offering the organs

to the entire country. We call this principle *local usage*. Note that national sharing and local usage are manifestations of assignment rules R0 and R1, respectively.

To compromise between national sharing and local usage, UNOS developed a hierarchical allocation system in which organs are allocated based on three levels of geographic proximity to the procurement OPOs. We call the three levels, the local, regional, and national levels. Each region contains a number of OPOs. It is clear that this allocation policy is of the spirit of rule R2. To get more information about organ allocation, we refer to the website of UNOS (*www.unos.org*) and Kong et al. (2009).

In this paper, we also consider two cases that empower nodes with some capability of changing their own assignment rules. First we define two terms, *outflow assignment* and *inflow assignment*. An assignment is an outflow assignment to a node i if a type A entity at node i is assigned to a type B entity at node $j \neq i$. An assignment is an inflow assignment to a node i if a type A entity at node $j \neq i$ is assigned to a type B entity at node i. One case we consider is to have a node change its assignment rule from R0 to R1 if certain outcome of the node is least favorite among all nodes, i.e., the difference between the cumulative outflow assignment cardinality (or utility) and the cumulative inflow assignment cardinality (or utility) is the maximum. We use R0R1 to represent this case that involves agent assignment behavior change. The other case is defined as follows. Given a node i, if certain outcome of the node is least favorite among all nodes, i.e., the cumulative outflow cardinality (or utility) is the lowest, we change the assignment rule of all the neighboring nodes of node i from R1 to R2 and this change only applies to node i. We term this case R1R2.

AGENT-BASED SIMULATION MODELING

Introduction to ABMS

Agent-based modeling and simulation (ABMS) models a system as a collection of autonomous decision-making entities called agents. Based on a set of rules, each agent individually assesses its situation, makes decisions and executes various behaviors (Bonabeau, 2002). ABMS has its historical roots in the study of complex adaptive systems (CAS), which was originally motivated by investigations into the adaptation and emergence of biological systems (North and Macal, 2007). ABMS builds upon proven, highly successful techniques such as discrete-event simulation and object-oriented programming. Discrete-event simulation provides an established mechanism for coordinating the interaction of individual agents within a simulation, while object-oriented programming provides frameworks to organize agents based on their behaviors. ABMS combines these techniques with a focus on bottom-up model construction (North and Macal, 2007). The research field of ABMS is rapidly growing in various fronts. Several examples of the ABMS application areas

are: modeling agent behaviors in the stock market (LeBaron, 2002) and supply chains (Fang *et al*, 2002); modeling the spread of epidemics (Huang *et al*, 2004) and the threat of bio-warfare (Carley, 2006); modeling the growth and decline of ancient civilizations (Kohler *et al*, 2005); and modeling the complexities of the human immune system (Folcik and Orosz, 2006).

An Agent-based Model for Dynamic Assignment

Model Components

In order to study the dynamic assignment problem on a scale-free network, we need to specify: 1) the network, 2) the arrival rate of resources, 3) the assignment rules, and 4) the utility measure. In the ABMS language, we need to specify the model components: the environment, the network, the agents, agent attributes and agent behaviors.

For the convenience of display and analysis, we choose the environment to be a 20 by 20 grid. We design two networks in the model: the initial network (the scale-free network that specifies the relationship between the nodes), the assignment network (which demonstrates the actual assignment process). We have 7 types of agents: model initializer, network builder/loader, network nodes, type A entity, type B entity, AB generator, and monitor. Note that only types A and B entity agents are real agents from the assignment viewpoint. Many others are simulation modules from a simulation model implementation viewpoint.

The key attributes and behaviors of the agents are specified as follows:

1) Model initializer

   Behavior: initialization (generates one network builder/loader agent, one AB generator agent, and one monitor agent.)

2) Network builder/loader

   Behavior: build a scale-free network/load a network

3) AB generator

   Attribute: type A entity arrival probability, type B entity arrival probability

   Behavior: generate type A entities and type B entities

4) Type A entity

   Attribute: generation location (home node), life span

   Behavior: update (life span decreases as the simulation clock moves, remove the entity itself from the system when it gets assigned or its life decreases to 0.)

5) Type B entity (same attribute and behavior as type A entity)

6) Network node

Attribute: id, neighbor list, distance list, important neighbor list, B list, assignment discrepancy (for the two cases R0R1 and R1R2)

Behavior: memorize neighbor and distance, make assignment, and calculate assignment utility

7) Monitor (for R0R1 and R1R2)

   Behavior: find the node with the greatest assignment discrepancy, and notify this node or its nearest neighbors to change the assignment rules

Schedule of Agent Behaviors

At the initialization stage, the following agent behaviors are carried out:

1) The model initializer agent initializes the model.
2) The network builder/loader agent generates a scale-free network (see Appendix B for the generation algorithm), or loads a saved network.
3) Each node agent ranks all the other nodes in the network by the lengths of the "geodesics" (a geodesic is a shortest path) in an ascending order. Here we assume that the length of a geodesic between any two immediate adjacent nodes is 1. Each node records the order in its "neighbor list", and record the length of the geodesics accordingly in its "distance list."

After the initialization stage, we denote the time interval for a complete assignment process as an "assignment round". In each assignment round, the following agent behaviors are carried:

1) The AB generator agent generates a type A entity at each node following a Bernoulli distribution with success probability "type A entity arrival probability", and generates type B entities similarly.
2) Each node assigns local type A entities to type B entities following the given assignment rule at the current assignment round.
3) Each node calculates the assignment utilities.

   For the assignment utilities, we assume self assignment has utility 1, and the utility of assignments to neighbors get discounted as the length of a geodesic increases. Specifically, we assume that assignment utility=(discount factor)^(geodesic length)

   We take the discount factor as 0.8. And the node can find the geodesic length to its recipient node in its "distance list."
4) Each type A or B entity updates itself. Decrease its "life" by 1. If its "life" reaches 0 or it get assigned in this round, remove itself from the system.

5) For R0R1 and R1R2, at the end of each assignment round, the monitor agent will rank the "assignment discrepancy" of all nodes, notify the node with highest discrepancy or its nearest neighbors to change its (or their) assignment rule(s) from the next assignment round.

Model Implementation

The simulation environment we choose for model implementation is Repast Symphony 1.2 (*http://repast.sourceforge.net*). Repast Symphony is a free and open source ABMS toolkit developed by the Repast group in the Division of Decision and Information Science at the Argonne National Laboratory. It is designed to include advanced point-and-click features for agent behavioral specification and dynamic model self assembly. The model components can be developed using any mixture of Java, Groovy, and flowcharts (North and Macal, 2007). The screen shots show the development environment using flowcharts (see Figure 1) and runtime environment of the package (see Figure 2).

For the interest of the space, we skip other details in implementation and only introduce how node agents' assignment behaviors are implemented:

1) R0:

   For each local type A entity, the node uses a query to get all unassigned type B entities in its "B list". If the "B list" is not empty, randomly choose an entity in "B list" as the recipient.

2) R1:

   For each local type A entity, first try to fill its "B list" with all unassigned local type B entities, and randomly choose one as the recipient. If there is no local type B entity available, fill the "B list" with all unassigned type B entities in the network, and make the assignment.

3) R2:

   For each local type A entity, loop through its neighbors following the order in the "neighbor list", and for each neighbor, add all its available type B entities to "B list". Thus the nodes get their "B list" ordered by the shortest path lengths, and then try to assign the type A entity to the nearest type B entity.

4) R0R1:

   Every fraction of the simulation duration, the monitor agent finds the node with the greatest assignment discrepancy, and changes the assignment rule of that node to R1 from the next assignment round.

5) R1R2:

   Every fraction of the simulation duration, the monitor agent finds the node with the greatest assignment discrepancy, and adds this node to its immediate adjacent neighbors' "important neighbor lists".

For each local type A entity, each node will fill its "B list" with the following order: first local type B entities, then type B entities of nodes on the "important neighbor list", and finally all other available type B entities.

COMPUTATIONAL EXPERIMENTS

Simulation Setup

We consider three randomly generated scale-free networks with 10, 20, and 50 nodes, respectively (see Figure 3 for the characteristics of the 50-node network). We assume the arrivals of the two types of entities are independent and identical Bernoulli processes. We specify that the success probability of the process is either 20% or 1 over the number of nodes. We consider that three life spans of each entity of either type. They are 1, 5, and 10 assignment rounds. We consider two types of outcomes. They are the overall network assignment utility and the average utility per assignment at each single node.

For each network, we run the simulation for each configuration that is determined by the two input parameters (success probability of the Bernoulli process and life span). Hence, for each network, we run the simulation for 6 different configurations.

At the beginning of the computational experimentation, we use preliminary tests to determine the simulation duration for each replication and the number of replications. We specify the simulation duration to be 50K assignment rounds for the 10-node and the 20-node networks and 100K assignment rounds for the 50-node network to guarantee the validity of our statistic analyses. We specify the number of replications to be 3.

Centralized Assignment rule evaluation

In our first computational experiment set, we deal with the three centralized assignment rules. In other words, all nodes adapt the same assignment rule and each node does not change its rule throughout the simulation duration. We report the overall system cumulative assignment utility for all input parameter configurations in Table 1. We report the average utility per assignment for each single node on the 10-node network in Figures 4, 5, and 6. We have one plot for each combination of the number of nodes, the success probability, and the life span. We present results against different network characters of each node. For the interest of the space, we do not include all the corresponding charts for the 20-node and 50-node networks. Instead, we in Figure 7 present the comparative studies between the three networks for the case where the success probability is 20% and the life span is 5.

Table 1 shows that R2 is better than R1 and R1 is better than R0, regardless of the simulation input parameters. As the life span increases, the overall system utility increases. Figures 4 – 6 show the following: 1) the utility is higher for nodes that are closer to the center of the network in all cases; 2) R0 is the worst almost surely and R2 is the best in most of the cases for almost all nodes; 3) the gap between the performance of R2 and R0 increases for those nodes that are closer to the center of the network; 4) the same gap seems to be larger for a larger success probability. We make the same observations for the 20-node and 50-node networks. For the interest of space, we do not present those results. Figure 7 shows that the gap between the performance of R2 and R0 seems to increase with the expansion of the network.

Decentralized assignment rule evaluation

In our second computational experiment set, we use two well designed cases to investigate the change of assignment rule at each node. We only test the two cases on the 20-node network. The fraction of the simulation duration we use in both cases is 2.5K assignment rounds. Hence, each node makes one decision on whether to change its assignment rule after every 2.5K assignment rounds. There are two outcome measures based on which each node makes the decision. One is cumulative assignment cardinality. The other is cumulative assignment utility.

We include the overall system cumulative assignment utility for all input parameter configurations in Table 1 as well. In addition, we report the average utility per assignment for each single node in Figures 8 – 11. Table 1 shows that the performance of R0R1 is between those of R0 and R1 regardless of the outcome measure each node uses. This observation also holds when comparing R1R2 with R1 and R2. Figures 8 – 11 present the same performance relationship among R0R1, R0, and R1. In addition, they show that allowing assignment rule changes among nodes may be more beneficial for the case where the life span is larger. Furthermore, Figures 10 and 11 show that it may be beneficial to adapt R0R1 for nodes that are closer to the center of the network, especially when the life span is larger and the success probability of the arrival process is higher.

CONCLUSIONS AND FUTURE RESEARCH

This paper presents an agent-based simulation study for the dynamic assignment problem defined on generic scale-free networks. We test a number of commonly used centralized assignment rules and make attempt to investigate the performance relationship between the centralized and decentralized rules.

The development of the agent-based simulation model offers much space for us to explore in the future on agent behaviors in dynamic assignment. We plan to study the cases where agents apply different assessment (metrics) to the system; the cases where agents conduct assessment periodically with different frequencies or even

constant; the cases where agents have different levels of information regarding the system and other agents' behaviors; and the cases where agents apply optimization in developing optimal assignment rules.

We also plan to further formalize the studied assignment rules along the direction of spatial prioritization. Then we plan to consider simulation optimization tools to develop optimal assignment rules. The key question is how to apply simulation optimization to study systems that allow agent autonomy and evolution.

Since we are dealing with dynamic assignment on a generic scale-free network, we will explore the applicability of various application domains. We also plan to integrate agent-based simulation and modeling with the GIS technology for resource allocation. This will help us understand the information process in a multi-agent environment.

Appendix A

Basic Dynamic Assignment Problem

There are five parts in the description: the basic notation, the information process, the decision process, the system dynamics, and the objective function. To a large extent, we apply the Markov process paradigm for simplicity. This description is adapted from Spivey and Powell (2004).

First we introduce the basic notation. We model the problem in discrete time over the finite horizon $\mathcal{T}$ = {0, 1, …, T} and there are finite numbers of possible types A and B entities available at a set of nodes $\mathcal{I}$. Define $\mathcal{A}$ to be the set of all type A entities that might possibly arrive in the system; and $\mathcal{B}$ to be the set of all type B entities that might possibly arrive in the system.

Next we describe the information process. Let

$$\hat{A}_{tia} = \begin{cases} 1, & \text{if entity } a \in \mathcal{A} \text{ becomes known in assignment cycle } t \text{ and at node } i; \\ 0, & \text{otherwise;} \end{cases}$$

$$\hat{B}_{tib} = \begin{cases} 1, & \text{if entity } b \in \mathcal{B} \text{ becomes known in assignment cycle } t \text{ and at node } i; \\ 0, & \text{otherwise;} \end{cases}$$

Let $\hat{A}_t = (\hat{A}_{tia})_{i\in\mathcal{I},a\in\mathcal{A}}$ and $\hat{B}_t = (\hat{B}_{tib})_{i\in\mathcal{I},b\in\mathcal{B}}$.

To describe the state of our system, we define

$$A_{tia} = \begin{cases} 1, & \text{if entity } a \in \mathcal{A} \text{ becomes known and available to be assigned in assignment cycle } t \text{ and at node } i; \\ 0, & \text{otherwise;} \end{cases}$$

B_{tib}

$= \begin{cases} 1, & \text{if entity } b \in \mathcal{B} \text{ becomes known and available to be assigned in assignment cycle } t \text{ and at node } i; \\ 0, & \text{otherwise.} \end{cases}$

Let $A_{ta} = (A_{tia})_{i \in \mathcal{I}}$ and $B_{tb} = (B_{tib})_{i \in \mathcal{I}}$. Let $A_t = (A_{tia})_{i \in \mathcal{I}, a \in A}$ and $B_t = (B_{tib})_{i \in \mathcal{I}, b \in B}$. Then the state of our system is presented as $S_t = (A_t, B_t)$. Note that we use the vector representation. A more natural representation is to use sets. Let $\mathcal{A}_t = \cup_{i \in \mathcal{I}} \{a \mid A_{tia} = 1\}$ and $\mathcal{B}_t = \cup_{i \in \mathcal{I}} \{a \mid B_{tia} = 1\}$. Hence, the state of the system can also be given by $\mathcal{S}_t = (\mathcal{A}_t, \mathcal{B}_t)$.

The main reason that $\hat{A}_{tia}$ and A_{tia}, $\hat{B}_{tia}$and $B_{tia}$are different is that an entity becomes unavailable once it is assigned. A specific reason in this work is that the entity has passed its life span.

We denote the assignment benefit to be

$$u_{ab} = \text{the benefit of assigning entity } a \text{ to entity } b = u(a, b).$$

The benefit function $u(a, b)$ is a deterministic function of the locations of entities *a* and *b*. We define a function $l : \mathcal{S}_t \to \mathcal{I}$. Hence in our problem $u(a, b) = u\big(l(a), l(b)\big)$. Note that we assume that $u(a, b)$ is independent of assignment cycle *t*.

In the simple model, the information arriving in assignment cycle *t* at node *i* is presented as

$$W_t = \big(\hat{A}_t, \hat{B}_t\big). \qquad (1)$$

We let ω be a sample realization of $(W_t)_{t \in \mathcal{T}}$. We assume that Ω represents the set of elementary outcomes. If $\mathcal{F}$ is the σ-algebra on Ω and $\mathcal{P}$ is a probability measure on Ω, then $(\Omega, \mathcal{F}, \mathcal{P})$ is a probability space. We let $\mathcal{F}_t$ be the σ-algebra generated by $(W_0, W_1, \ldots, W_t)$, where the sequence of increasing sub-σ-algebra $\mathcal{F}_t$ forms a filtration. We assume that our information process satisfies $\sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{I}} \hat{A}_{tia}(\omega) \le 1$ and $\sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{I}} \hat{B}_{tib}(\omega) \le 1$ almost surely (a.s.), which means that every entity either of type A or type B can become known at most once in at most one node, and not every entity will become known in every outcome.

Next we model the decision process. We represent the decision we have made by:

$$x_{tab} = \begin{cases} 1, & \text{if entity } a \in \mathcal{A} \text{ is assigned to entity } b \in \mathcal{B} \text{ at time } t; \\ 0, & \text{otherwise;} \end{cases}$$

$x_t = (x_{tab})_{a \in \mathcal{A}, b \in \mathcal{B}}$

The function that returns a decision can be represented as X_t^π a member of a family of functions $(X^\pi)_{\pi\in\Pi}$ that returns a decision vector x_t at time *t*. We refer to a particular function X^π as a policy, and let the set Π represent our family of policies.

We can write our decision function as taking the general form

$$X_t^\pi = \text{argmax}_x\, C_t^\pi(x|S_t). \qquad (2)$$

X_t^π is an $\mathcal{F}_t$-measureable function providing an assignment at time *t* from a given policy π. The goal is to find a computationally tractable function and the problem (2) must be solved subject to the flow conservation

$$\sum_{b\in\mathcal{B}} x_{tab} \le A_{ta} \text{ and } \sum_{a\in\mathcal{A}} x_{tab} \le A_{tb}. \qquad (3)$$

Next we describe the dynamics of our system. It is presented as

$$A_{t+1} = A_t - x_t^A + \hat{A}_{t+1} \qquad (4)$$

$$B_{t+1} = B_t - x_t^B + \hat{B}_{t+1} \qquad (5)$$

Note that our decision function returns decisions $x_t = x_t^\pi$ that must satisfy Constraint (3). It is clear that from Equations (4) and (5) that the random variables A_t, B_t, and S_t are defined for a given policy π. We, however, drop the superscript π for notational simplicity.

Finally, we describe our objective function. The benefit of an assignment is presented as

$$C_t(x_t) = \sum_{a\in\mathcal{A}_t}\sum_{b\in\mathcal{B}_t} u_{ab}x_{tab}. \qquad (6)$$

For a state S_t and a policy π, define, for each *t*,

$$F_t^\pi(S_t) = E\left\{\sum_{t'=t}^{T} C_{t'}(X_{t'}^\pi)\,\middle|\,S_t\right\}.$$

The optimization problem can now be formally stated as

$$F_t^*(S_t) = \sup_\pi F_t^\pi(S_t).$$

The solution of the dynamic assignment problem can be found by solving

$$F_0^*(S_0) = \sup_\pi F_0^\pi(S_0). \qquad (7)$$

It is computationally intractable for the exact solution of the problem. It is even computationally intractable for many classes of approximations. Therefore, given several policies $\Pi' \subset \Pi$, we in our work evaluate $F_0^{\pi}(S_0)$ for all $\pi \in \Pi'$. Note that the policies we consider in the simulation are topologically based.

Appendix B

Algorithm for the Barabási-Albert Model (Barabási and Albert, 1999)

Step 1: Given an initial small network $G_0 = (V_0, E_0)$ with $|V_0| = n_0$, and a number K. Let G_1 be G_0.

Step 2:

2a: (Growth) Generate a new node v_k and $n_k(\leq n_0)$ edges that connect v_k and n_k different nodes in V_k. Let the set of the selected n_k nodes be $V_k' \subseteq V_k$ and let the set of generated edges be E_k', i.e., $E_k' = \{(v_k, v) \mid v \in V_k'\}$.

2b: (Preferential attachment) To generate V_k', we check each node in V_k. For each node $v_i \in V_k$, let

$p_i = \dfrac{d_i}{\sum_{v_j \in V_k} d_j}$, where d_i is the degree of node v_i. We generate a random number p, if $p \leq p_i$, we include v_i in V_k' and include (v_k, v_i) in E_k'.

Step 3: Let $V_k \leftarrow V_k \cup \{v_k\}$ and $E_k \leftarrow E_k \cup E_k'$. If $k \leq K$, Set $k \leftarrow k + 1$ and Go back to Step 2. Otherwise, STOP.

After K steps this algorithm likely results in a scale-free network with $N = K + n_0$ and nt edges.

Appendix C

Network Measures

Centrality is the extent to which a node is central to a network (Brass, 1995). In this paper, we use 4 centrality measures to describe the positions of individual nodes in the network: degree, closeness, betweenness and eigenvector. We use a social network analysis software, UCINET (*www.analytictech.com*), to calculate the measures. The definitions of the 4 measures are as follows (Bonacich, 1972; Freeman, 1979):

1. Degree (Freeman, 1979):

   The number of nodes adjacent to a given node in a symmetric graph is the degree of that vertex. Since we use symmetric networks and the links are not weighted, the degree of a node is just the number of links the node possesses.

2. Closeness (Freeman, 1979):

The farness of a node is the sum of the lengths of the geodesics (shortest path between two nodes) to every other node. The reciprocal of farness is closeness. The normalized closeness of a node is the closeness of the node divided by the maximum possible closeness and expressed as a percentage.

3. Betweenness (Freeman, 1979):

   Let b_{jk} be the proportion of all geodesics linking node *j* and node *k* which pass through node *i*. The betweenness of node *i* is the sum of all b_{jk} where *i*, *j* and *k* are distinct. Betweenness is therefore a measure of the number of times a vertex occurs on a geodesic. The normalized betweenness is the betweenness divided by the maximum possible betweenness and expressed as a percentage.

4. Eigenvector (Bonacich, 1972):

   Given an adjacency matrix *A*, the centrality of node *i* (denoted by c_i), is given by $c_i = \alpha \sum A_{ij} c_j$,where α is a parameter. The centrality of each node is therefore determined by the centrality of the nodes it is connected to. The parameter α is required to give the equations a non-trivial solution and is therefore the reciprocal of an eigenvalue. It follows that the centralities will be the elements of the corresponding eigenvector. The normalized eigenvector centrality is the scaled eigenvector divided by the maximum possible difference and expressed as a percentage.

REFERENCES

Albert R, Jeong H and Barabási AL (2000). Error and attack tolerance in complex networks. *Nature* **406**: 378.

Barabási AL and Albert R (1999). Emergence of scaling in random networks. *Science* **286**:509-512.

Bonabeau E (2002). Agent-based modeling: Methods and techniques for simulating human systems. In: *Sackler Colloquium on Adaptive Agents, Intelligence, and Emergent Human Organization: Capturing Complexity through Agent-Based Modeling 10*. National Academy of Sciences: Washington DC, pp 7280-7287.

Bonacich P (1972). Factoring and weighting approaches to status scores and clique identification. *Journal of Mathematical Sociology* **2**: 113-120.

Brass DJ (1995). A social network perspective on human resource management. In: Rowland KM and Ferris GR (eds). *Research in Personnel and Human Resource Management 13*. JAI Press: Greenwich, pp. 39-79.

Carley K (2006). BioDefense through city level multi-agent modeling of bio and chemical threats. In: *Arizona Spring Biosurveillance Workshop*: Tucson, AZ.

Everitt D and Manfield D (1989). Performance analysis of cellular mobile communication systems with dynamic channel assignment. *IEEE Journal on Selected Areas in Communications* **7(8)**: 1172-1180.

Fang C, Kimbrough SO, Pace S, Valluri A and Zheng Z (2002). On adaptive emergence of trust behavior in the game of stag hunt. *Group Decision and Negotiation* **11(6)**:449-467.

Florian M, Mahut M and Tremblay N (2001). A hybrid optimization-mesoscopic simulation dynamic traffic assignment model. In: *Proceedings of 2001 IEEE Intelligent Transportation Systems*, pp 118-121.

Folcik V, and Orosz CG (2006). An agent-based model demonstrates that the immune system behaves like a complex system and a scale-free network. In: *SwarmFest 2006*: University of Notre Dame, South Bend, IN.

Freeman LC (1979). Centrality in social networks: Conceptual clarification. *Social Networks* **1**: 215-239.

Huang CY, Sun CT, Hsieh JL and Lin H (2004). Simulating SARS: Small-world epidemiological modeling and public health policy assessments. *Journal of Artificial Societies and Social Simulation* **7(4)**.

Kohler TA, Gumerman GJ and Reynolds RG (2005). Simulating ancient societies. *Scientific American* **293(1)**:76-84.

LeBaron, B (2002). Short-memory traders and their impact on group learning in financial markets. In: *Sackler Colloquium on Adaptive Agents, Intelligence, and Emergent Human Organization: Capturing Complexity though Agent-Based Modeling 10*. National Academy of Sciences: Washington DC, pp. 7201-7206.

Powell WB (1996). A stochastic formulation of the dynamic assignment problem, with an application to truckload motor carriers. *Transportation Science* **30(3)**: 195-219.

Kong N, Schaefer AJ, Hunsaker B and Roberts MS (2009). Maximizing the efficiency of the U.S. liver allocation system through region design. Under second review in *Management Science*.

Mahmassani HS (2001). Dynamic Network Traffic Assignment and Simulation Methodology for Advanced System Management Applications. *Networks and spatial economics* **1(3-4)**: 267-292.

LeBaron B (2002). Short-memory traders and their impact on group learning in financial markets. Supplement 3: Arthur M. Sackler Colloquium of the National Academy of Sciences. In: *Sackler Colloquium on Adaptive Agents, Intelligence, and Emergent Human Organization: Capturing Complexity though Agent-Based Modeling 10*. National Academy of Sciences: Washington DC, pp 7201-7206.

North MJ and Macal CM (2007). *Managing Business Complexity: Discovering Strategic Solutions with Agent-Based Modeling and Simulation*. Oxford University Press: New York, NY.

Rainer B, Dell'Amico M and Martello S (2009). *Assignment Problems*. Society of Industrial and Applied Mathematics: Philadelphia.

Shechter SM, Bryce CL, Alagoz O, Kreke JE, Stahl JE, Schaefer AJ, Angus DC and Roberts MS (2005). A clinically based discrete event simulation of end-stage liver disease and the organ allocation process. *Medical Decision Making* **25(2)**: 199-209.

Spivey MZ and Powell WB (2004). The dynamic assignment problem. *Transportation Science* **38(4)**: 399-419.

Taranto SE, Harper AM, Edwards EB, Rosendale JD, McBride MA, Patrick Daily O, Murphy D, Poos B, Reust J and Schmeiser B (2000). Developing a national allocation model for cadaveric kidneys. In: *Proceedings of the 32nd Winter Simulation Conference, Orlando, FL*, pp 1971-1977.

van den Hout WB, Smits JMA, Deng MC, Hummel M, Schoendube F, Scheld HH, Persijn GG and Laufer G (2003). The heart-allocation simulation model: a tool for comparison of transplantation allocation policies. *Transplantation* **76(10)**: 1492-1497.

Yang H, Ye M, Tang WH and Wong SC (2003). Modeling urban taxi services: A literature survey. In: Lam WHK and Bell MGH (eds). *Advanced Modeling for Transit Operations and Service Planning*. Pergamon, pp 257-286.

Table 1 Overall System Cumulative Assignment Utility

Size	Success probability	Life Span	R0	R1	R2	R0R1(c)*	R0R1(u)*	R1R2(c)	R1R2(u)
10 node	0.1	1	18,033	18,431	18,764				
		5	31,734	33,252	34,042				
		10	33,557	35,772	36,496				
	0.2	1	47,248	49,506	50,886				
		5	67,127	72,448	74,388				
		10	69,289	76,391	77,519				
20 node	0.05	1	15,927	16,142	16,522	16,001	16,007	16,272	16,281
		5	28,555	29,482	30,646	28,920	28,883	29,892	29,971
		10	30,270	31,875	33,131	30,804	30,673	32,254	32,193
	0.2	1	98,230	104,645	111,408	100,970	99,870	106,951	106,921
		5	125,041	138,295	145,319	130,082	127,073	140,878	140,844
		10	127,603	143,981	149,620	133,847	130,312	146,567	145,916
50 node	0.02	1	27,227	27,466	28,479				
		5	49,369	50,412	53,377				
		10	52,443	53,951	57,820				
	0.2	1	476,702	520,528	596,072				
		5	553,340	628,845	708,768				
		10	559,860	653,570	720,087				

* c represents cardinality; and u represents utility

Figure 1 Model Developing Enviroment (a portion of node agent behaviors, using flowchart)

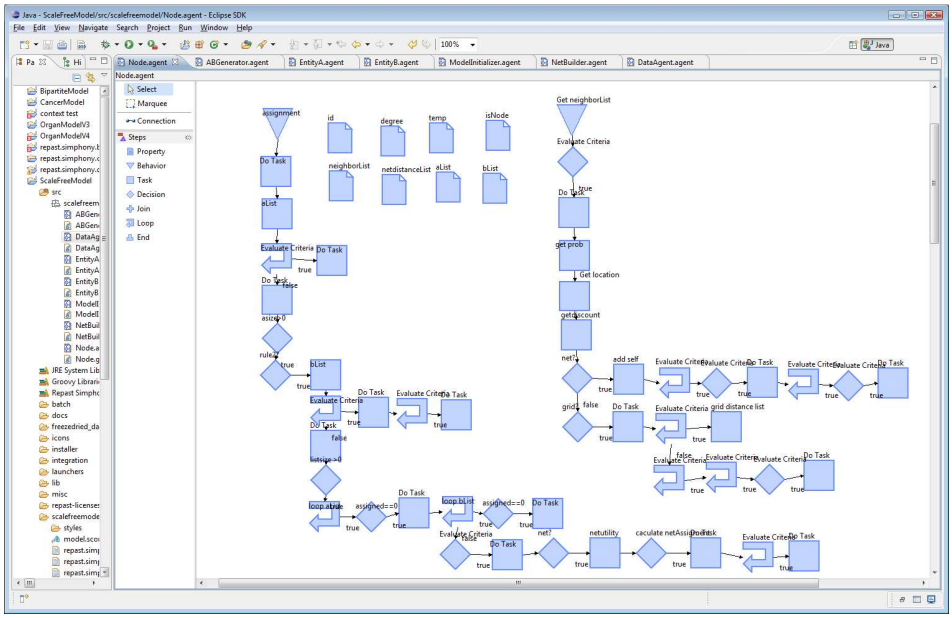

Figure 2: Model Runtime Environment (square represents node; circle represents entity A; cross represents entity B; blue line represents the initial network; yellow arrow represents the assignment network)

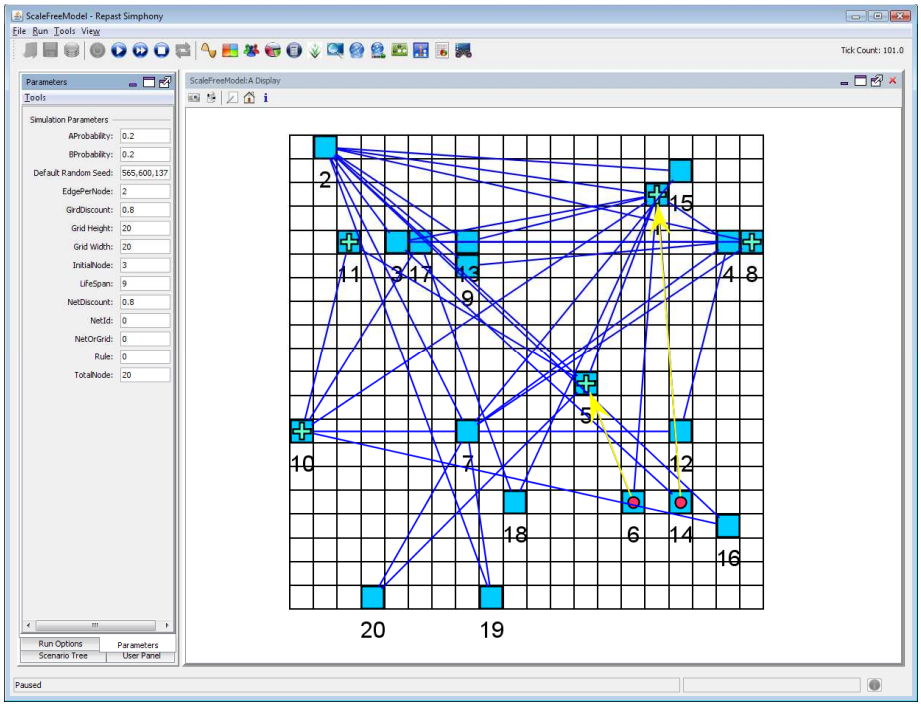

Figure 3 Characteristics of the 50-node Network

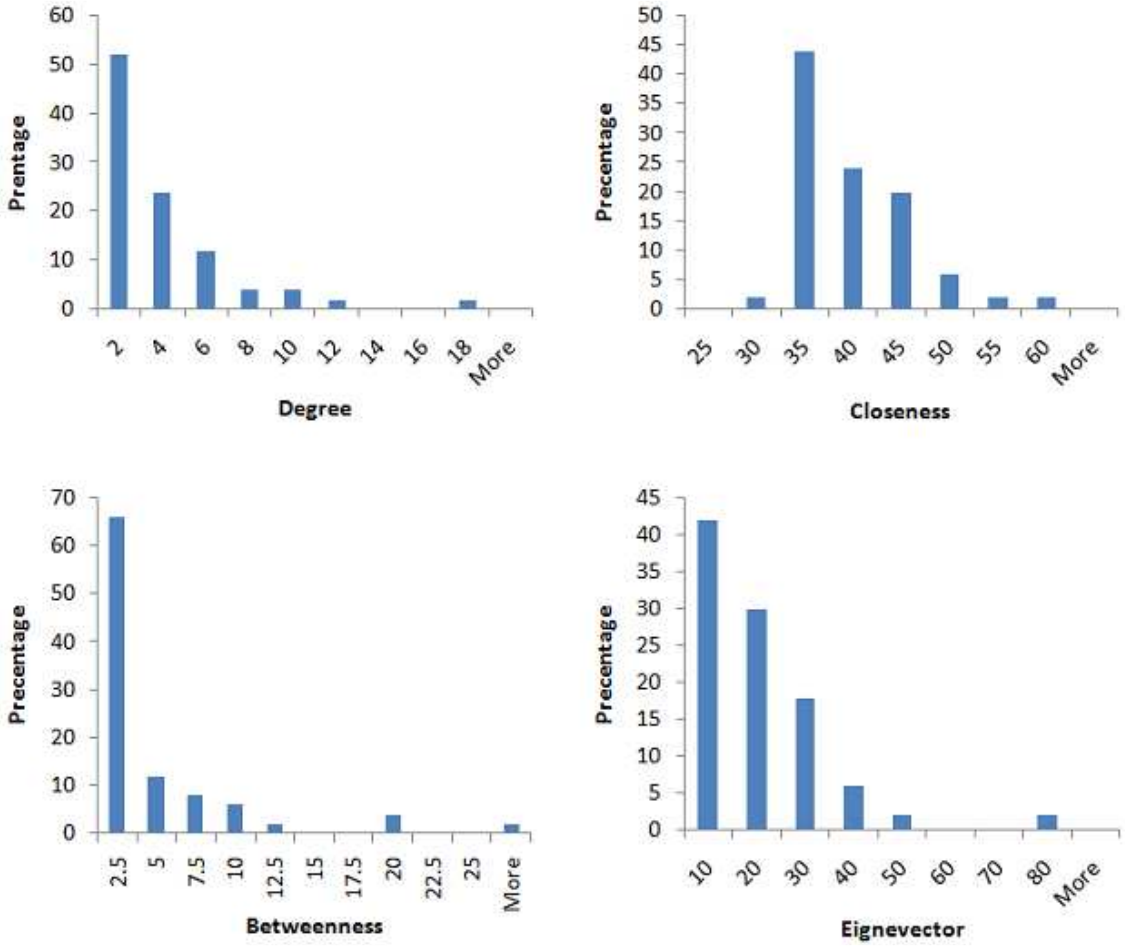

Figure 4 Average Utility per Assignment for the 10-node Network (Life Span = 1 Assignment Round)

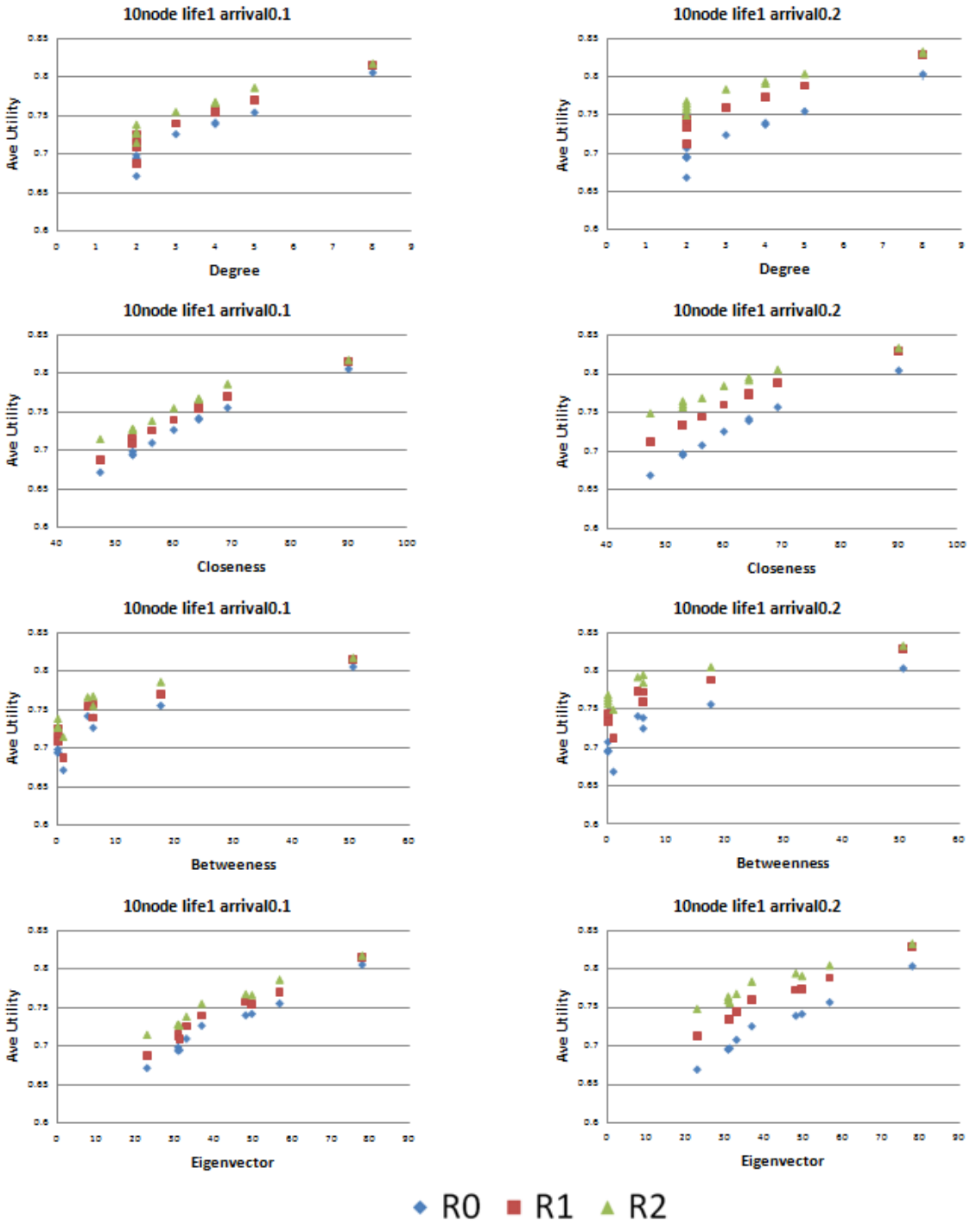

Figure 5 Average Utility per Assignment for the 10-node Network (Life Span = 5 Assignment Rounds)

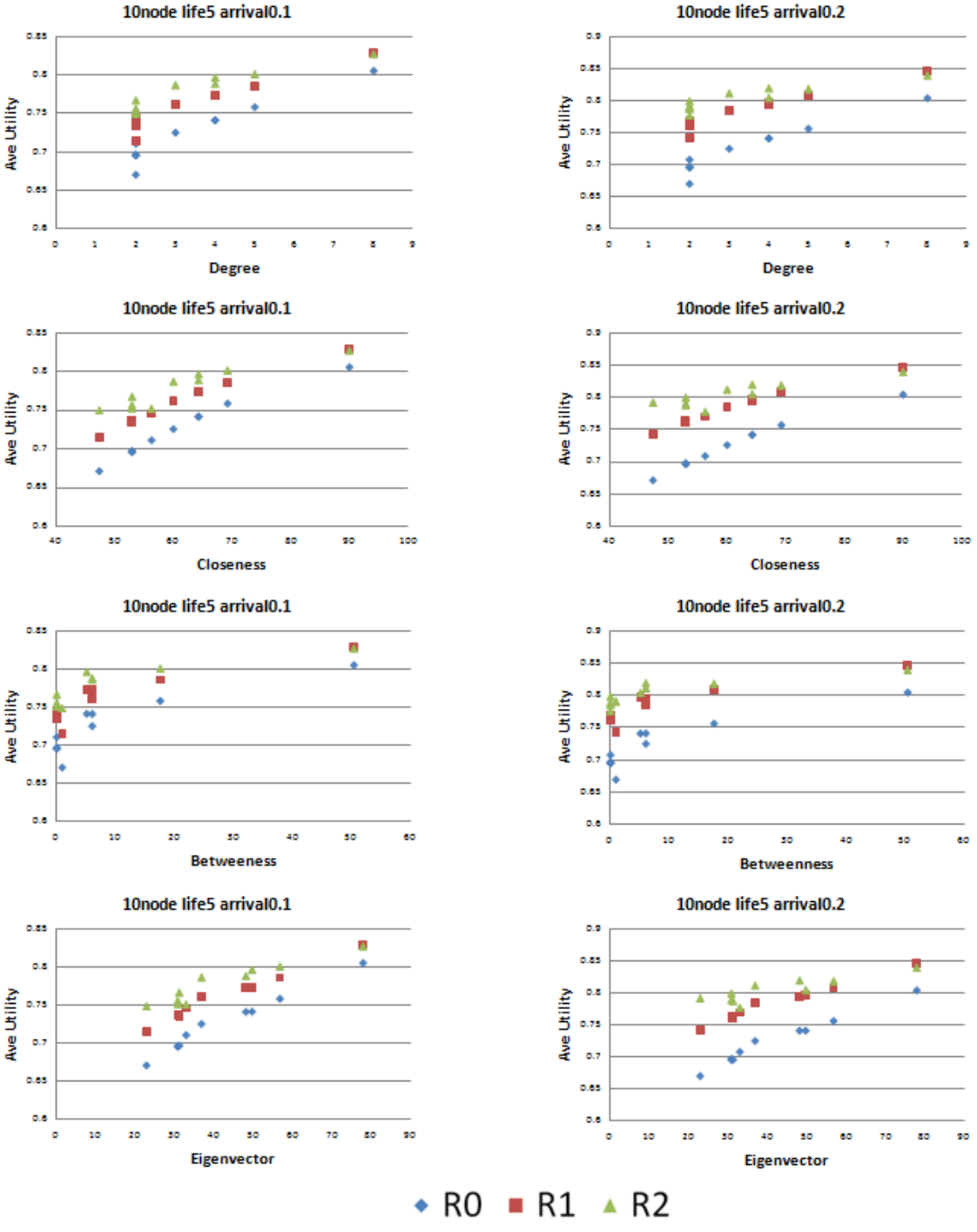

Figure 6 Average Utility per Assignment for the 10-node Network (Life Span = 10 Assignment Rounds)

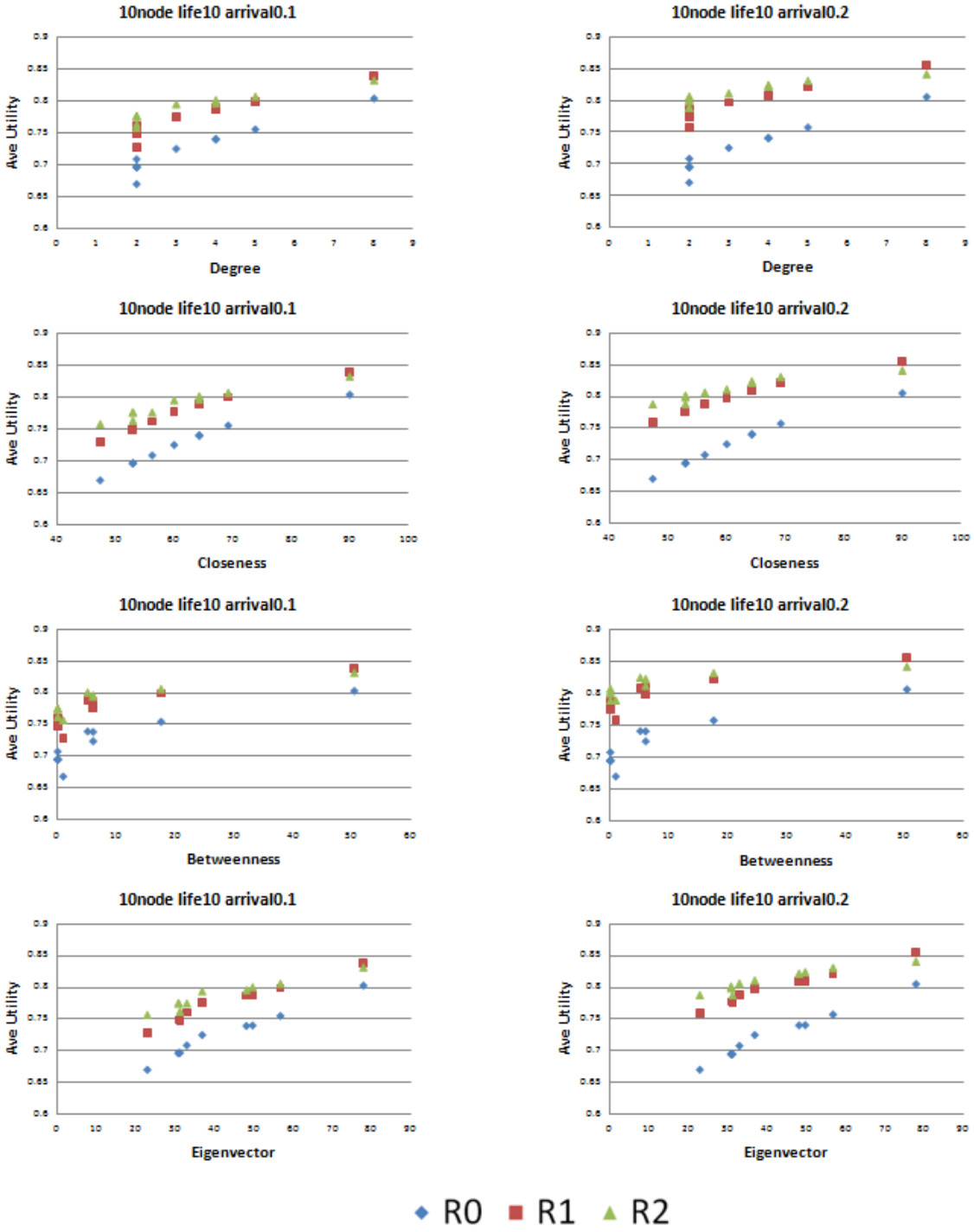

Figure 7 Average Utility per Assignment Success Probability = 20%; Life Span = 5 Assignment Rounds)

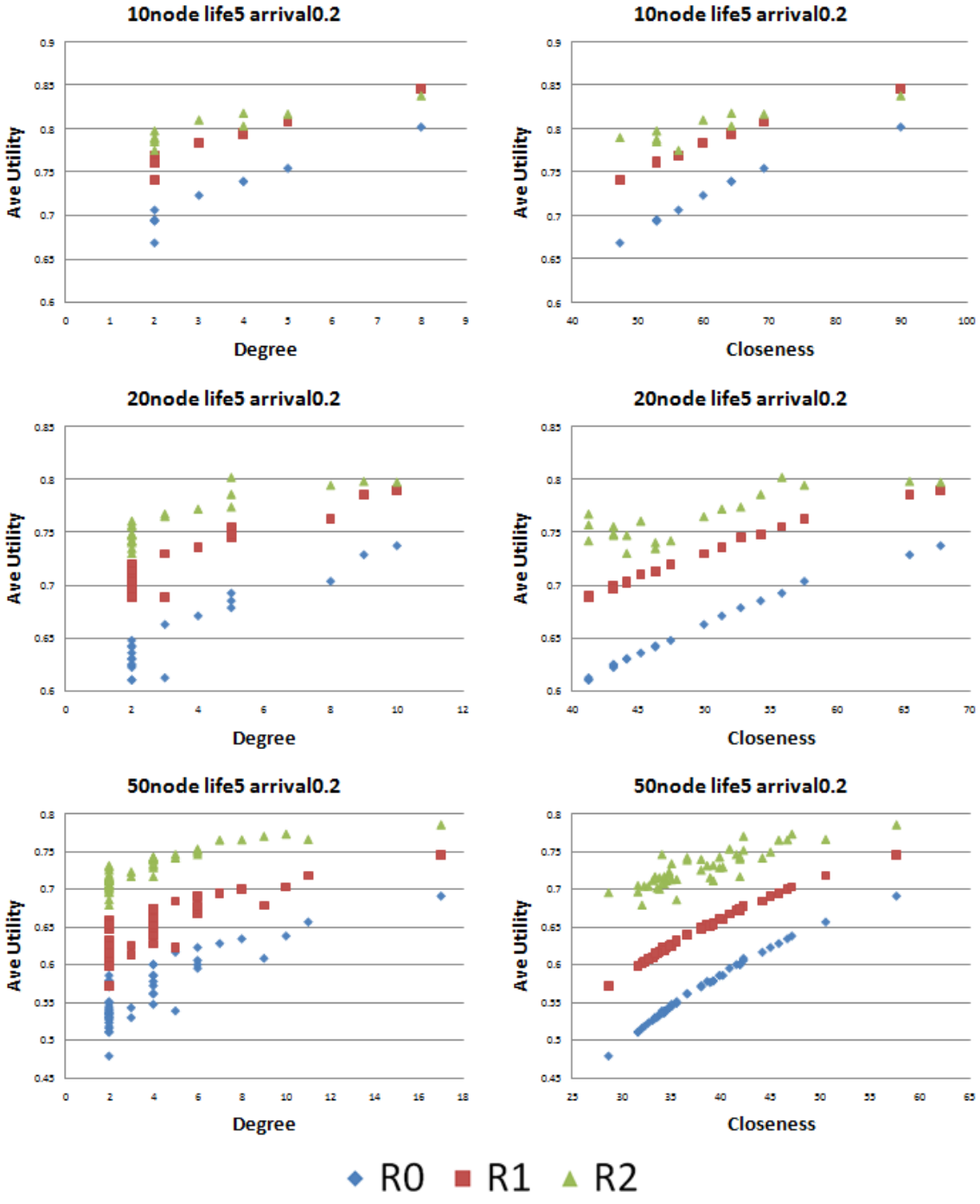

Figure 8 Comparison of R0, R1, and R0R1 in Average Utility per Assignment (Outcome Measure is Cumulative Assignment Cardinality)

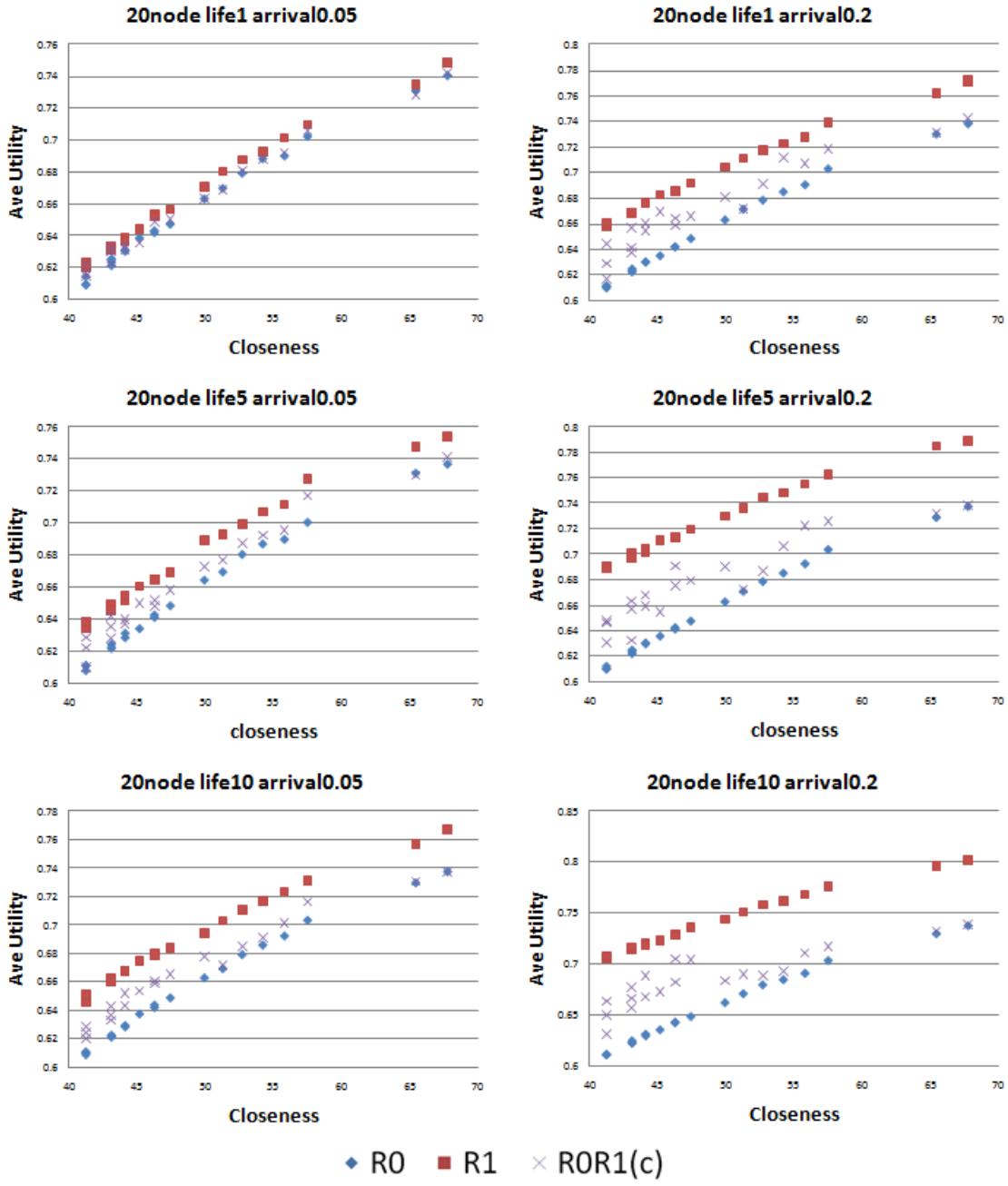

Figure 9 Comparison of R0, R1, and R0R1 in Average Utility per Assignment (Outcome Measure is Cumulative Assignment Utility)

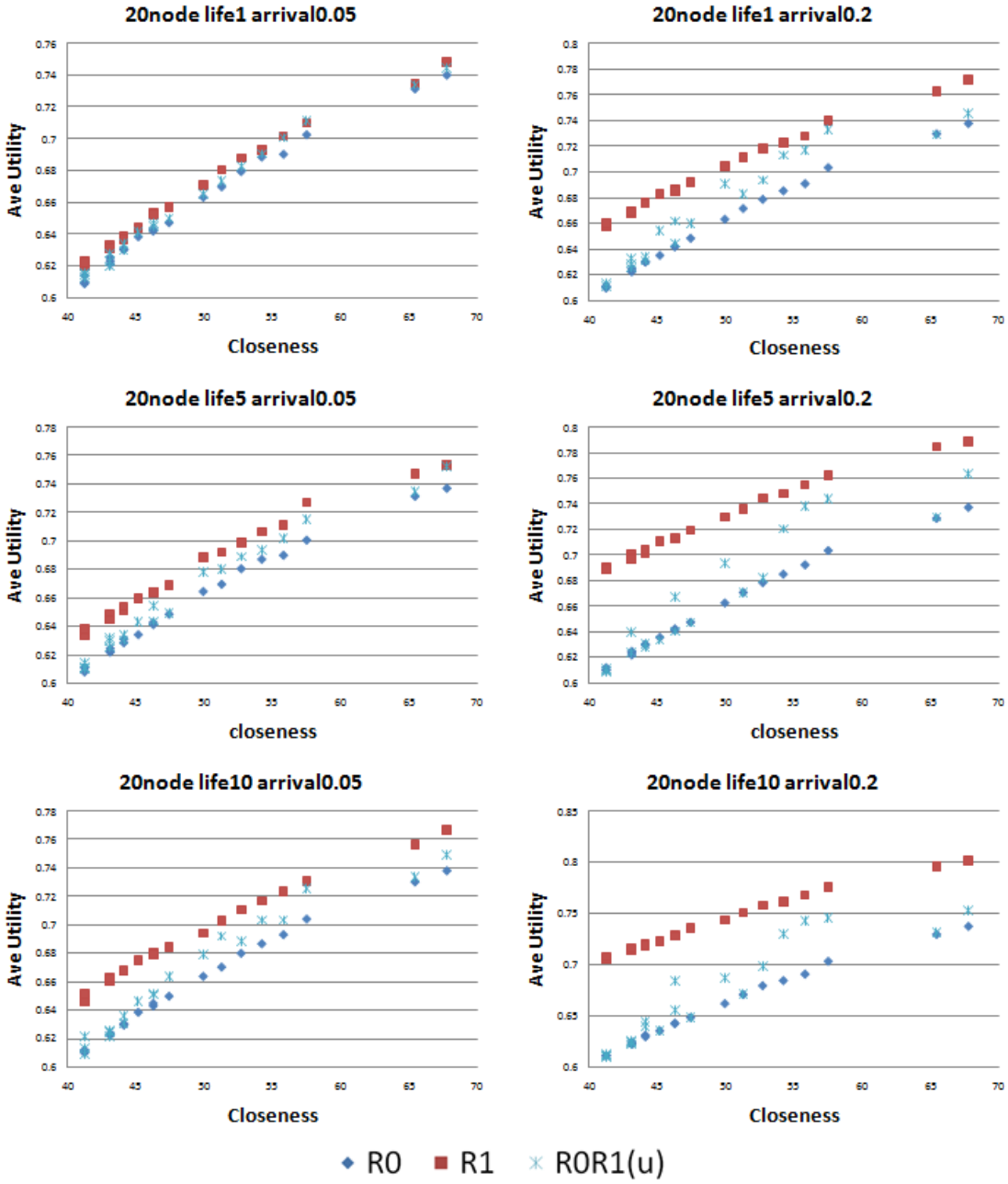

Figure 10 Comparison of R1, R2, and R1R2 in Average Utility per Assignment (Outcome Measure is Cumulative Assignment Cardinality)

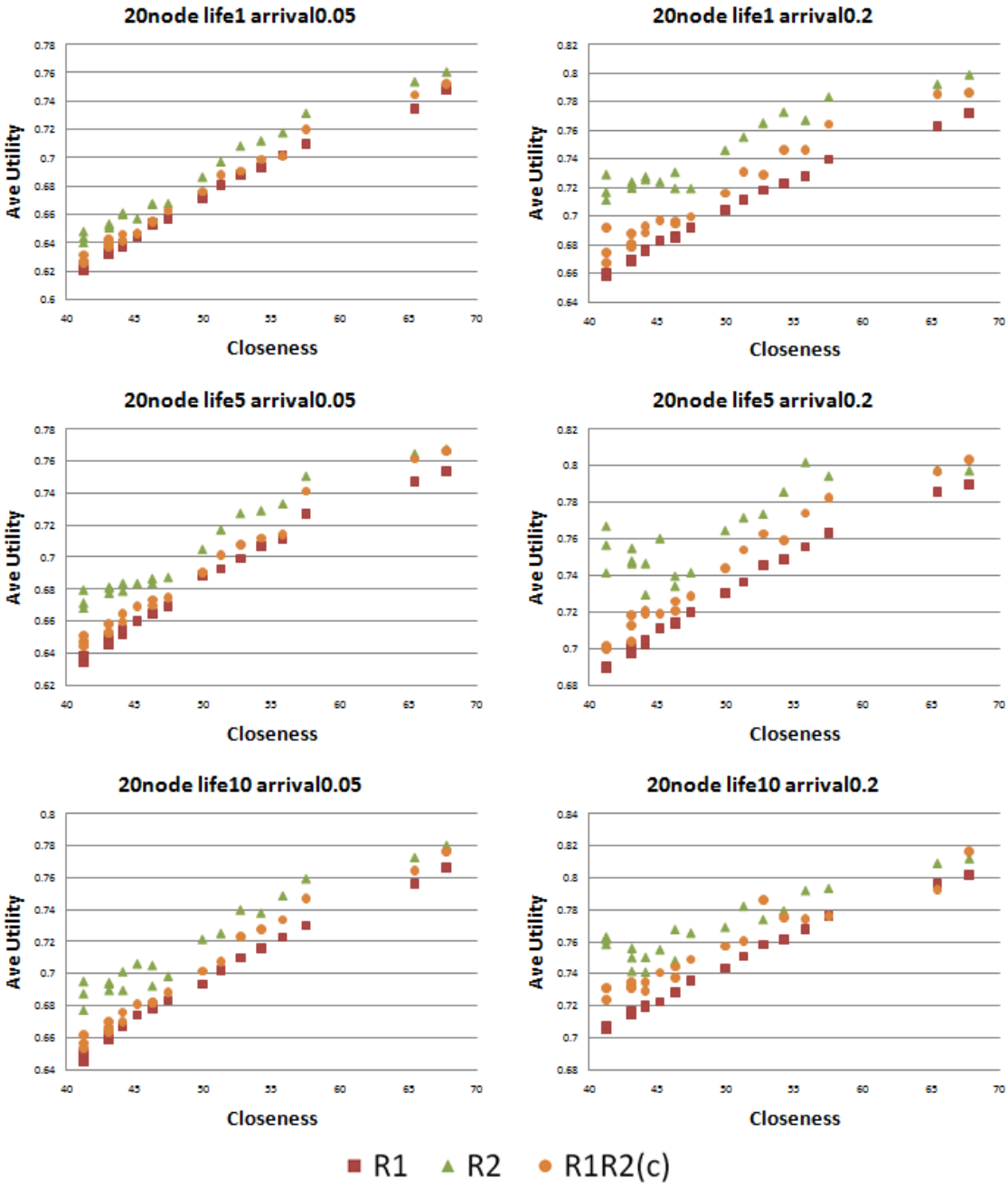

Figure 11 Comparison of R1, R2, and R1R2 in Average Utility per Assignment (Outcome Measure is Cumulative Assignment Utility)