Multiobjective Evolutionary Algorithms for Dynamic Social Network Clustering

Keehyung Kim
School of Computer Science and Engineering
Seoul National University
Seoul, 151-744 Korea
keehyung@snu.ac.kr

RI (Bob) McKay
School of Computer Science and Engineering
Seoul National University
Seoul, 151-744 Korea
rimsnucse@gmail.com

Byung-Ro Moon
School of Computer Science and Engineering
Seoul National University
Seoul, 151-744 Korea
moon@snu.ac.kr

ABSTRACT
The main focus of this paper is to propose integration of dynamic and multiobjective algorithms for graph clustering in dynamic environments under multiple objectives. The primary application is to multiobjective clustering in social networks which change over time. Social networks, typically represented by graphs, contain information about the relations (or interactions) between online materials (or people). A typical social network tends to expand over time, with newly added nodes and edges being incorporated into the existing graph. We reflect these characteristics of social networks based on real-world data, and propose a suitable dynamic multiobjective evolutionary algorithm. Several variants of the algorithm are proposed and compared. Since social networks change continuously, the immigrant schemes effectively used in previous dynamic optimization give useful ideas for new algorithms. An adaptive integration of multiobjective evolutionary algorithms outperformed other algorithms in dynamic social networks.

Categories and Subject Descriptors
H.4 [Information Systems Applications]: Miscellaneous; I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods, and Search—Heuristic methods; I.5.3 [Pattern Recognition]: Clustering—Algorithms

General Terms
Algorithms, Experimentation

Keywords
Dynamic optimization, multi-objective evolutionary algorithm, social network, elitism-based immigrants

1. INTRODUCTION
A social network is a graph representing the interactions (or connections) between individuals (or online materials). Since social networks contain useful information for online marketing, social behavior, and web characteristics, they have attracted much attention in a range of research fields [29, 34] including computer science [33]. One of the most important topics, from the earliest work on social networks, has been finding optimal clusters in a social network, where each cluster represents a small community (or a set of materials) which share more common traits than usual [4, 35]. In this context, clustering involves finding groups of similar objects from often huge data sets [15, 24].

Determining the optimal cluster structure in a social network can help solve real-world problems, extending research in such areas as focused marketing, and allowing economists and statisticians to test economic theories and induced policy changes in relatively self-contained submarkets.1

This form of clustering is equivalent to graph partitioning, an NP-hard problem [5, 17], to which approximation by genetic algorithms (GA) is a suitable approach. See [6, 23] for previous successful work using GAs with reordering and local heuristics. In most clustering algorithms, the number of clusters should be specified beforehand. For example, Bui and Moon [6] used a single objective (minimum edge-cut) optimized under an equal balance constraint, to bi-partition a graph. In general, though, the structure of a social network is not known in advance, and may change constantly in a dynamic social environment. So it is better to determine the number of clusters during the evolutionary process. This may be quite difficult when multiple clustering criteria are used. Thus Multiobjective Evolutionary Algorithms (MOEAs) may be appropriate for social graph clustering [13]. Handl and Knowles [21] used two objectives, the cluster count being automatically determined. In [12, 21, 22], multi-objective clustering problems with an unspecified number of clusters were investigated.

In clustering social networks, one of the most important properties has been largely ignored: that they usually change dynamically. Recently, social network research has begun to focus on dynamic characteristics [27]. Berger-Wolf and Saia [2] proposed a framework for identifying communities in dynamic social networks. They discovered some notable features of online social networks. Communities tend to evolve gradually over time [1], rather than assembling or disbanding spontaneously. Also, people change their home community infrequently [1], and interact with the home community most of the time [34]. Other research on social networks found continuous expansion over time. Whenever the graph

---

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 200X ACM X-XXXX-XX-X/XX/XX ...$10.00.

1Yahoo! Research, http://research.yahoo.com/project/2368
Clustering social networks is a graph partitioning problem; in general, this involves finding \( k \) disjoint clusters \( C_1, \ldots, C_k \subset V \) in a graph \( G = (V, E) \), where the vertices within each cluster are densely connected to one another, but loosely connected to other vertices. The optimization problem is to partition \( G \) into disjoint subgraphs while maximizing certain criteria.

Many clustering algorithms use properties of specific classes of graphs to generate efficient heuristics. How to measure cluster quality is an important issue, because simple algorithms such as min-cut [14] may partition into unbalanced subgraphs whose sizes may differ radically. To remedy this, various constraints have been introduced, such as ratio cut [7], normalized cut [32], and min-max cut [14]. In [26], normalized cut, modularity, and silhouette width are assessed independently as fitness functions for graph clustering. In [12, 13], min-max cut and silhouette index are used as multiple objectives.

The min-max cut [12, 14] shown in equation 1 aims to maximize the similarity within each cluster, while minimizing that between clusters.

\[
\text{MinMaxCut} = \sum_{m=1}^{k} \frac{\text{cut}(C_m, V - C_m)}{\text{assoc}(C_m)} \tag{1}
\]

where \( \text{cut}(G_1, G_2) = \sum_{i \in G_1, j \in G_2} w(i,j) \) and \( \text{assoc}(G) = \sum_{i,j \in G} w(i,j) \).

In equation 1, \( k \) is the number of clusters, \( \text{cut}(C_m, V - C_m) \) is the sum of edge weights between the vertices in \( C_m \) and in the rest of the graph \( V - C_m \), and \( w(i,j) \) is the weight of the edge between nodes \( i \) and \( j \). This quantity is to be minimized in the corresponding optimization problem.

The silhouette index proposed in [30], and shown in equation 2, is a score based on the intuition that a cluster should gather similar elements.

\[
S(i) = \frac{a'(i) - b'(i)}{\max(a'(i), b'(i))} \tag{2}
\]

where \( a'(i) = \text{avg}_{j \in C} w(i,j), i \in C \), \( b'(i) = \max_{C \neq C_i} d'(i) \), \( d'(i) = \text{avg}_{j \in C} w(i,j), i \notin C \).

In equation 2, \( S(i) \) is a silhouette value assigned to each vertex \( i \). \( a'(i) \) is the average similarity between \( i \in C \) and other vertices in \( C \). \( b'(i) \) is the maximum average similarity between vertex \( i \) and other clusters \( C(i \notin C) \).

The global silhouette index (equation 3) is an average over all vertices in the graph. Rousseeuw calculated the silhouette based on dissimilarity in his work [30]. The analogous formula for similarity that we use, equation 2, was presented in [26].

The global silhouette is a cluster validation index that can be used to compare the quality of clustering solutions with different numbers of clusters. It takes on values in the range \([-1, 1]\), and is to be maximized.

\[
\text{Global Silhouette} = \frac{\sum_{i \in V} S(i)}{|V|} \tag{3}
\]

The related work section of this paper is to propose an effective framework for dynamic multiobjective evolutionary algorithms (DMOEA) particularly for online social network clustering (i.e., growing with time). We describe and compare several methods for dynamic and/or multiobjective optimization, including a new hybrid approach.

The rest of the paper is organized as follows: Section 2 provides an overview of related work. In section 3, we explain the algorithm and method we propose. Section 4 provides experimental results, followed by discussion in section 5. Section 7 concludes the paper and gives directions for future work.

2. RELATED WORK

2.1 Clustering as an Optimization Problem

Clustering social networks is a graph partitioning problem; in general, this involves finding \( k \) disjoint clusters \( C_1, \ldots, C_k \subset V \) in a graph \( G = (V, E) \), where the vertices within each cluster are densely connected to one another, but loosely connected to other vertices. The optimization problem is to partition \( G \) into disjoint subgraphs while maximizing certain criteria.

Many clustering algorithms use properties of specific classes of graphs to generate efficient heuristics. How to measure cluster quality is an important issue, because simple algorithms such as min-cut [14] may partition into unbalanced subgraphs whose sizes may differ radically. To remedy this, various constraints have been introduced, such as ratio cut [7], normalized cut [32], and min-max cut [14]. In [26], normalized cut, modularity, and silhouette width are assessed independently as fitness functions for graph clustering. In [12, 13], min-max cut and silhouette index are used as multiple objectives.

The min-max cut [12, 14] shown in equation 1 aims to maximize the similarity within each cluster, while minimizing that between clusters.

\[
\text{MinMaxCut} = \sum_{m=1}^{k} \frac{\text{cut}(C_m, V - C_m)}{\text{assoc}(C_m)} \tag{1}
\]

where \( \text{cut}(G_1, G_2) = \sum_{i \in G_1, j \in G_2} w(i,j) \) and \( \text{assoc}(G) = \sum_{i,j \in G} w(i,j) \).

In equation 1, \( k \) is the number of clusters, \( \text{cut}(C_m, V - C_m) \) is the sum of edge weights between the vertices in \( C_m \) and in the rest of the graph \( V - C_m \), and \( w(i,j) \) is the weight of the edge between nodes \( i \) and \( j \). This quantity is to be minimized in the corresponding optimization problem.

The silhouette index proposed in [30], and shown in equation 2, is a score based on the intuition that a cluster should gather similar elements.

\[
S(i) = \frac{a'(i) - b'(i)}{\max(a'(i), b'(i))} \tag{2}
\]

where \( a'(i) = \text{avg}_{j \in C} w(i,j), i \in C \), \( b'(i) = \max_{C \neq C_i} d'(i) \), \( d'(i) = \text{avg}_{j \in C} w(i,j), i \notin C \).

In equation 2, \( S(i) \) is a silhouette value assigned to each vertex \( i \). \( a'(i) \) is the average similarity between \( i \in C \) and other vertices in \( C \). \( b'(i) \) is the maximum average similarity between vertex \( i \) and other clusters \( C(i \notin C) \).

The global silhouette index (equation 3) is an average over all vertices in the graph. Rousseeuw calculated the silhouette based on dissimilarity in his work [30]. The analogous formula for similarity that we use, equation 2, was presented in [26].

The global silhouette is a cluster validation index that can be used to compare the quality of clustering solutions with different numbers of clusters. It takes on values in the range \([-1, 1]\), and is to be maximized.

2.1.1 Dynamic Clustering

Dynamic graph clustering seems to have received only very limited previous study [31, 19], with all work we are aware of using classic graph techniques; there does not seem to be any previous work on heuristic methods for dynamic graph clustering.

2.2 Multiobjective Evolutionary Algorithms

The goal of multiobjective optimization is to find a population of solutions approximating the pareto front. Thus it is important to track a variety of elite individuals, while single objective algorithms need only find one best individual.

Many real-world problems have more than one objective, often requiring a trade-off among multiple conflicting objectives. Multi-Objective Evolutionary Algorithms (MOEA) are designed to approximate the pareto front, and many kinds have been proposed [11, 16, 37]. We base our work on one of the best-known, NSGA-II [11]. Further details on this and other MOEAs can be found in [9, 13].

Algorithm 1 NSGA-II

1: Randomly initialize population \( P_0 \)
2: Compute fitness values of individuals in \( P_0 \)
3: Perform non-dominated sorting on \( P_0 \)
4: Apply tournament selection on \( P_0 \)
5: Generate child population \( Q_0 \) by crossover and mutation
6: while max no. of generations not reached do
7: Generate \( R_t = P_t \cup Q_t \)
8: Perform non-dominated sorting on \( R_t \)
9: Copy individuals from non-dominated fronts to \( P_{t+1} \)
10: Apply tournament selection on \( P_{t+1} \)
11: Generate child population \( Q_{t+1} \) by crossover and mutation
12: end while

NSGA-II uses a layered classification of the individuals. At each generation, all non-dominated individuals are given rank 1 and removed from the population. The newly non-dominated individuals are given rank 2 and removed. This process continues until no individuals are left. The individuals are sorted and assigned fitness values based on rank.
crowding comparison operator helps to maintain diversity. The pseudocode of NSGA-II is given in Algorithm 1.

2.3 Immigrant Schemes in Dynamic Optimization

Most real-world problems involving social networks are dynamic. Thus they require different methods from Static Optimization Problems (SOPs). For example, a certain amount of diversity should be maintained, to support adaptation in a dynamically changing environment. While maintaining diversity, it is also important to converge to optima fast enough after a change to provide good solutions consistently over time.

A standard GA (SGA) works well for SOPs, converging relatively rapidly to near-optimal solutions. In contrast, over-convergence is a major problem for Dynamic Optimization Problems (DOPs) since it reduces exploration of new optima arising in the dynamic environment. Thus it is important to maintain a certain level of population diversity, to permit adaptation to changes in the overall landscape. The random immigrants scheme (RIGA) [20] replaces a proportion of the population with randomly generated individuals, thus increasing diversity.

RIGA is more effective than SGA in highly dynamic environments. However it suffers if the rate of change is low, because of poor exploitation of information gained in previous generations. Yang [36] proposed variants of RIGA known as EIGA (Elitism-based Immigrants GA) and HIGA (Hybrid Immigrants GA). EIGA and HIGA outperformed RIGA for several DOPs at realistic levels of change.

Algorithm 2 HIGA

1: Randomly initialize population $P_0$
2: Evaluate fitness values on $P_0$
3: while max no. of generations not reached do
4:   Select $P'_t$ from $P_t$
5:   Apply crossover and mutation on $P'_t$
6:   Evaluate the intermediate population $P'_t$
7:   Denote the elite in $P'_t$ by $E_t$
8:   Generate $r_{ei} \times n$ immigrants by mutating $E_t$
9:   Generate $r_{ri} \times n$ immigrants randomly
10:  Evaluate fitness value of immigrants
11:  Replace the worst individuals in $P'_t$ with the immigrants
12:  Copy individuals in $P'_t$ to $P_{t+1}$
13: end while

EIGA introduces elitism-based immigrants by mutating the elite individuals, and replaces some proportion of the population as in RIGA. HIGA combines RIGA and EIGA: the aim is to maintain both exploration and exploitation, to perform consistently in all levels of change.

Algorithm 2 presents the general idea of HIGA: $r_{ei}$ and $r_{ri}$ are the proportions of random and elite immigrants. In HIGA, we usually set $r_{ei} = r_{ri}$; the algorithm degenerates to RIGA if $r_{ei} = 0$, and to EIGA if $r_{ri} = 0$.

3. COMBINING MOEA WITH ADAPTIVE IMMIGRANTS

3.1 Hybrid Immigrants NSGA-II

We use NSGA-II as the core algorithm of our Dynamic MOEA (DMOEA). To deal with dynamicity, we investigate three immigrant schemes: RI-NSGA-II (Random Immigrants NSGA-II) and EI-NSGA-II (Elitism-based Immigrants NSGA-II), and our preferred scheme, Hybrid Immigrants NSGA-II (HI-NSGA-II) which combines random and elite immigrant schemes.

Algorithm 3 HI-NSGA-II

1: Randomly initialize population $P_0$
2: Compute fitness values of individuals in $P_0$
3: Perform non-dominated sorting on $P_0$
4: Apply binary tournament selection on $P_0$
5: Generate child population $Q_0$
6: Apply recombination and mutation
7: while max no. of generations not reached do
8:   Generate $R_t = P_t \cup Q_t$
9:   Perform non-dominated sorting on $R_t$
10: Copy individuals from non-dominated fronts to $P_{t+1}$
11: Apply binary tournament selection on $P_{t+1}$
12: Generate child population $Q_{t+1}$
13: Apply recombination and mutation
14: Denote the pareto front in $P_t$ by $E_t$
15: Generate $r_{ei} \times n$ immigrants by mutating $E_t$
16: Generate $r_{ri} \times n$ immigrants randomly
17: Evaluate fitness value of immigrants
18: Combine child population $Q_{t+1}$ with immigrants
19: end while

HI-NSGA-II follows the same general procedure as NSGA-II, only differing in how the child population $Q$ is generated. In Algorithm 3, part of $Q$ is generated by crossover and mutation from $P$ just as in NSGA-II; the remainder is introduced by an immigration scheme. This differs in detail from the HIGA (Algorithm 2), where it is necessary to use the ‘replacement’ part of the population for immigrants; here, $Q$ is an external child population. Thus in HI-NSGA-II, $r_{ei}$ and $r_{ri}$ refer to the proportions of random and elitism-based immigrants in population $Q$. $r_e$ is the proportion generated by crossover and mutation.

3.2 Adaptive Immigrants NSGA-II

HI-NSGA-II suffers from an important limitation: it lacks any means to predict or observe the changing fitness landscape, so as to control the trade-off between exploration and exploitation, since the ratio of immigrants is statically fixed during the evolution. AI-NSGA-II (Adaptive Immigrants NSGA-II) is designed to impart dynamic adaptability into the algorithm, eliminating this weakness by controlling the ratio of immigrants through observing and predicting the fitness landscape.

Two opposing situations could arise while searching for the optimal pareto front in a dynamic multiobjective environment. These extremes are illustrated in Figure 1. The left figure shows a population located compactly inside a small range on a fitness landscape. In EI-NSGA-II, and also HI-NSGA-II, the strong effect of elitism might lead the population to a compact region despite the crowding distance diversity mechanism of NSGA-II. The other extreme situation, seen in Figure 1 (b), maintains a broad diversity on the fitness landscape, but fails to find high quality solutions. RI-NSGA-II’s strong diversification mechanism can raise this
problem. Thus, it is important to observe the distribution of the population on the fitness landscape, and encourage the population to evolve in a favorable direction.

Pseudo code for AI-NSGA-II is given in Algorithm 4. AI-NSGA-II determines whether it will generate a new child by crossover, random immigrants or elite immigrants, based on selected parents. If both parents are located outside the pareto front, the algorithm assumes exploitation is required, generating a new child from the elite. If exactly one parent is on the pareto front, but the distance between the parents is less than 10% of the longest distance between solutions, it assumes exploration is required, so a random immigrant is generated. Otherwise, the new child is generated by crossover and mutation. AI-NSGA-II uses a heuristic approach, in observing the overall fitness landscape as reflected in the selected parents’ relation in the fitness landscape.

3.3 Genetic Mechanisms

![Figure 1: Two possible distributions of a population in the fitness landscape](image)

**Algorithm 4 AI-NSGA-II**

1: Randomly initialize population $P_0$
2: Compute fitness values of individuals in $P_0$
3: Perform non-dominated sorting on $P_0$
4: Apply binary tournament selection on $P_0$
5: Generate child population $Q_0$
6: Apply recombination and mutation
7: while max no. of generations not reached do
8:    Generate $R_t = P_t \cup Q_t$
9:    Perform non-dominated sorting on $R_t$
10:   Copy individuals from non-dominated fronts to $P_{t+1}$
11: while no of generated childs less than $n$ do
12:   Select two parents $p_1, p_2$ with binary tournament selection on $P_{t+1}$
13:   if $p_1, p_2 \not\in$ pareto front then
14:     Randomly select $p$ on $P_{t+1}$
15:     Generate immigrant $c$ by mutating $p$
16:   else if $p_1 \in$ pareto front and $p_2 \not\in$ pareto front and $\text{dist}(p_1, p_2) \leq 0.1 \times \max\text{dist}(P_{t+1})$ then
17:     Generate immigrant $c$ randomly
18:   else
19:     Generate child $c$ by crossover and mutation
20: end if
21: Evaluate fitness value of $c$
22: Generate child population $Q_{t+1} = Q_{t+1} \cup c$
23: end while
24: end while

**Figure 2:** Locus adjacency based representation of chromosome

**Representation**

We use a locus-based adjacency representation [28] as shown in Figure 2. In this graph representation, each individual $g$ consists of $|V|$ genes and each gene $g_i$ can take an integer value $j$ between 1 and $|V|$. A value $j$ assigned to the gene $g_i$ is interpreted as a link from vertex $i$ to $j$ [22]. After decoding the genotype, we can determine the cluster structure, as shown in Figure 2. To figure out cluster structures, decoding of this representation is required, based on identification of all connected components. The same connected components are then assigned to one cluster. This decoding step takes linear time [10].

The locus-based adjacency encoding scheme has several major advantages; most important, there is no need to fix the number of clusters in advance, as it is automatically determined in the decoding step. Hence, it is possible to evolve and compare solutions without setting the number of clusters statically.

**Selection operator**

Tournament selection is used. Previous research on the effect of selection has shown that tournament selection outperforms roulette-wheel selection in many DOPs [25]. It gives more chance for poor solutions to be selected, leading to higher diversity.

When choosing a parent from population $P$, two individuals are selected, and compared using dominance ranking [16]. Depending on probability $p_s$, one parent is finally chosen from them.

**Crossover operator**

Uniform crossover [18] is chosen because it is unbiased with respect to the ordering of genes, and can generate any combination of alleles from the two parents. Several researchers have shown that one-point or two-point crossover perform well for reordering of vertices in graph partitioning [6, 23]. However, in a dynamic environment, the change in social network structures may mislead the evolutionary process.

**Fitness function**

There is a wide variety of methods to measure graph-based clustering quality, each of which could potentially be used as a fitness function [26]. We decided to focus on two conflicting cluster quality scores as objectives to determine the optimal graph clustering.
Min-max cut and global silhouette index are used as two very different objectives.

\[
f_1 = \frac{1}{1 + \text{MinMaxCut}} \quad (4)
\]

\[
f_2 = \text{Global Silhouette} \quad (5)
\]

To apply the MOEA, both of the objectives are taken as maximizing. In equation 4, the first objective min-max cut is converted as $1/(\text{MinMaxCut} + 1)$. This is maximized if all vertices are in the same cluster. The second objective from equation 5, global silhouette, is maximized if each vertex is a separate cluster.

4. EXPERIMENTAL EVALUATION

4.1 Dynamic Multiobjective Environment

Our experiments aim to compare different algorithms, observing their performance in a typical social networking dynamic multiobjective environment. Specifically, we compare our proposed algorithm, tailored to this role—namely AE-NSGA-II—with NSGA-II, RI-NSGA-II, EI-NSGA-II and HI-NSGA-II (the parameter settings are detailed in Table 1) on real-world social network data. We use the well-known data of Cheng et al [8], crawled from YouTube. It treats the YouTube videos as forming a directed graph, in which each video forms a node. If video b is in the top 20 related videos of video a, then there is a directed edge from a to b. The crawler uses breadth-first search (BFS) to find videos in the graph. The videos were crawled daily from Feb. 22 to May. 18, 2007. Each time a video was crawled, a new time-stamped record was created.

Among the various datasets, we illustrate our experimental result based on the data from Mar. 5 on as an example. To construct the initial network, we started with a randomly selected set of 366 videos, and the 420 relation between these videos. Each time a video was crawled, at most 20 edges and 20 nodes could be newly added to the existing social network. Intuitively, we expect the social network to increase over time. We followed the network for a total of 20 changes (the network did not change every day). After these 20 changes, the graph consisted of 562 nodes with 840 edges. Between two consecutive changes, the algorithms are permitted 20 generations to converge. After 20 generations, the next change occurs, and in general, the social network grows.

4.2 Performance Evaluation

Each variant is run 10 times, and the quality of the solution populations is estimated using dominance ranking [16] of the pareto approximation sets. For each approximation set, we count the number of other sets which dominate it. This was done across all runs for each variant for a given generation. Thus we could identify the best, worst and median set, for each variant and each generation. The median set in each generation was selected for performance comparison with other variants.

To compare the approximation sets, we used the binary $\epsilon$-indicator [38]. It relies on binary $\epsilon$-dominance, which may be defined as follows: In a maximization problem with $n$ positive objectives, an objective vector $z^1 = z^1_1, z^1_2, ..., z^1_n \geq \epsilon$ dominates another objective vector $z^2 = z^2_1, z^2_2, ..., z^2_n \geq \epsilon$ if $\forall i \in [1, n], z^1_i \geq \epsilon, z^2_i$ if $\forall i \in [1, n], \exists \epsilon > 0$. Binary $\epsilon$-indicator $I_\epsilon$ can then defined as in equation 6.

\[
I_\epsilon(A, B) = \inf_{z^1 \in A} \sup_{z^2 \in B} I(z^1, z^2) \quad (6)
\]

\[
F(A, B, t) = (I_\epsilon(A_t, B_t) \geq 1) \land (I_\epsilon(B_t, A_t) < 1) \quad (7)
\]

where $A_t$ and $B_t$ are two approximation sets at time $t$.

\[
F(A, B) = \sum_{t=1}^{400} I(A, B, t) \quad (8)
\]

The boolean function $F(A, B, t)$ of equation 7 compares $A$ and $B$ at time $t$: if $F(A, B, t)$ returns true, $A$ is a better approximation than $B$ at time $t$. Otherwise it is not possible to determine whether $A$ is better (of course, it might be worse – i.e. $F(B, A, t)$ might return true). Function $F$ in equation 8 calculates how many times $A$ performs better than $B$ over the total 400 generations. This indicator is used in this study to compare different variants using the same set of objectives.

5. EXPERIMENTAL RESULTS

In Figure 3, we display details from the median-rank run of each variant for four selected times (immediately before the 5th, 10th, 15th, and 20th changes in network structure). The plots show the pareto approximations sets for each variant at the relevant time. The results are largely as our previous discussion foreshadowed. RI-NSGA-II is much more exploratory on the pareto front in comparison with EI-NSGA-II, but is not exploitative enough to approximate the pareto optima well. RI-NSGA-II’s pareto front is distributed broadly on the fitness landscape, whereas EI-NSGA-II’s is densely bunched. But, RI-NSGA-II never pareto dominates EI-NSGA-II, while EI-NSGA-II pareto dominates part of the population from RI-NSGA-II at all times. This characteristic is consistent across all generations, not merely those plotted: RI-NSGA-II maintains a broad pareto front, but never dominates other variants; EI-NSGA-II maintains a high quality pareto front, but only in a densely-bunched region. Thus RI-NSGA-II typifies the the left behavior we described in Figure 1, while EI-NSGA-II typifies the right.

In fact, the original core MOEA, NSGA-II, pareto dominates RI-NSGA-II some of the time, so it is clear that, at least for this kind of problem, RI-NSGA-II is too exploratory. HI-NSGA-II, combining the exploratory power of random immigrants and the exploitative power of elite immigrants, maintains a broader pareto front than EI-NSGA-II, with better quality than RI-NSGA-II. However it is always partly dominated by EI-NSGA-II. Conversely, it maintains a narrower pareto front than NSGA-II, though always dominating part. Thus it seems that combining random and elite immigrants statically results in an intermediate pareto front quality.
Table 1: General Settings for each Variant

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Population P size</th>
<th>Child population Q size</th>
<th>crossover (r_c)</th>
<th>RI (r_RI)</th>
<th>EI (r_EI)</th>
<th>Selection</th>
<th>Crossover</th>
<th>Mutation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSGA-II</td>
<td>30</td>
<td>60</td>
<td>1.0</td>
<td>0</td>
<td>0</td>
<td>Binary</td>
<td>Uniform</td>
<td>Adjacency</td>
</tr>
<tr>
<td>RI-NSGA-II</td>
<td>30</td>
<td>60</td>
<td>0.5</td>
<td>0.5</td>
<td>0</td>
<td>tournament</td>
<td>crossover</td>
<td>mutation</td>
</tr>
<tr>
<td>EI-NSGA-II</td>
<td>30</td>
<td>60</td>
<td>0.5</td>
<td>0</td>
<td>0.5</td>
<td>(p_x = 0.6)</td>
<td>(p_c = 1.0)</td>
<td>(p_m = 0.01)</td>
</tr>
<tr>
<td>HI-NSGA-II</td>
<td>30</td>
<td>60</td>
<td>0.5</td>
<td>0.25</td>
<td>0.25</td>
<td>(p_s = 0.6)</td>
<td>(p_c = 1.0)</td>
<td>(p_m = 0.01)</td>
</tr>
<tr>
<td>AI-NSGA-II</td>
<td>30</td>
<td>60</td>
<td>Determined by adaptive heuristic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

On the other hand, AI-NSGA-II achieved what we want. It pareto dominated the other four variants almost always. While AI-NSGA-II kept a broad pareto front comparable to HI-NSGA-II, the quality of pareto approximation outperformed those of EI-NSGA-II, and hence all the others.

The above results are summarized in Table 2, showing the value of function $F$ for each variant’s median pareto approximation, and in Table 3, in which the variant in the column is compared with the variant in the row. The sign “+” indicates that the column variant outperforms the row

Table 2: Performance Comparison between Approaches Based on Function $F$

<table>
<thead>
<tr>
<th>Approach</th>
<th>NSGA-II</th>
<th>RINS</th>
<th>EINS</th>
<th>HINS</th>
<th>AINS</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSGA-II</td>
<td>-</td>
<td>94</td>
<td>0</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>RI-NSGA-II</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>EI-NSGA-II</td>
<td>13</td>
<td>15</td>
<td>-</td>
<td>129</td>
<td>0</td>
</tr>
<tr>
<td>HI-NSGA-II</td>
<td>18</td>
<td>25</td>
<td>2</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>AI-NSGA-II</td>
<td>211</td>
<td>174</td>
<td>88</td>
<td>244</td>
<td>-</td>
</tr>
</tbody>
</table>
variant slightly over the 400 generations, while the sign “−” indicates the reverse. The “=” indicates that neither variant shows any improvement over the other. The sign “+ +” (“++”) indicates that the column variant outperforms the row more than 20% (40%) overall. As before, “−−” (”− −”) have the reverse meaning.

6. DISCUSSION

Based on our results, for real-world data in a dynamic multiobjective setting, AI-NSGA-II produced easily the best pareto approximation among the five approaches, outperforming the other algorithms most of the time. Based on these results, it is possible to rank the performance of the algorithms in fitting the pareto front as: AI-NSGA-II, EI-NSGA-II, HI-NSGA-II, NSGA-II, and RI-NSGA-II. Among the other algorithms, RI-NSGA-II showed the worst performance, apparently since it focuses too much on exploring the search space, with too little attention to quality of solutions. EI-NSGA-II partially dominated the other three most of the time, but suffered from a lack of exploration – it only maintains a high quality pareto approximation over a small range. It is hard to distinguish between HI-NSGA-II and NSGA-II, both performing very similarly.

Summing up, we believe that an integrated approach, controlling the distribution of the pareto front approximation, is a key issue in dynamic multiobjective problems. Unlike in DOPs, a static approach to immigration schemes does not appear to work well in DMOPs. We can see this effect in Figure 4, where we plot the ratios $r_c$, $r_r$, $r_e$ in AI-NSGA-II over the process of evolution. The adaptive scheme of AI-NSGA-II tries to balance between exploration and exploitation by adapting these ratios.

To the best of our knowledge, this is the first attempt to combine dynamic and multi-objective techniques for clustering. The results show that it is feasible and useful to do so, and that among a range of possible techniques, AI-NSGA-II performed best.

7. CONCLUSIONS

In this paper, we considered an optimization problem arising in social networking, namely clustering under multiple objectives on a dynamically changing (but as with many social networks, mostly expanding) network. We used real-world data to generate a suitable benchmarking test for comparing different multi-objective algorithms for this purpose. We showed that it was possible to perform such dynamic clustering. We investigated a number of suitable algorithms; and we proposed a new algorithm, AI-NSGA-II, which substantially outperformed other approaches.

Our framework is somewhat restricted in its forms of dynamics, since only new nodes and edges are added into previous social networks in our experiments – no nodes or edges are deleted. Real-world social networks not only grow over time, but also see change in a small proportion of relations in existing social networks. Such scenarios make it more difficult to model the problem meaningfully in graph-theoretic contexts, but our definition and experiments still capture important features of real-world social networks. Further study of more realistic dynamic graph clustering poses an interesting direction for future research.

8. REFERENCES

of vesicular selection in dynamic environments. In
Proceedings of the 11th Annual conference on Genetic
evolutionary computation, pages 1867–1868.

algorithm for clustering in social networks. In GECCO
'09: Proceedings of the 11th Annual Conference on
Genetic and Evolutionary Computation, pages

[27] H. Ohtsuki, C. Hauert, E. Lieberman, and M. A.
Nowak. A simple rule for the evolution of cooperation
on graphs and social networks. Nature,

clustering problems. In Proceedings of the 3rd Annual
Conference on Genetic Programming, pages

[29] J. M. Read and M. J. Keeling. Disease evolution on
networks: The role of contact structure. The Royal
Society Proceedings. B, Biological Sciences,

interpretation and validation of cluster analysis.
Journal of Computational and Applied Mathematics,

[31] B. Saha and P. Mitra. Dynamic algorithm for graph
clustering using minimum cut tree. In Proceedings of

Conference on Computer Vision and Pattern
Recognition, page 731, 1997.

[33] C. Tantipathananandh, T. Berger-Wolf, and
D. Kempe. A framework for community identification
in dynamic social networks. In KDD '07: Proceedings of the 13th ACM SIGKDD International Conference
on Knowledge Discovery and Data Mining, pages

[34] S. Wasserman and K. Faust. Social Network Analysis:
Methods and Applications. Cambridge University

[35] S. Wasserman and K. Faust. Social network analysis:
Methods and applications. Cambridge Univ Pr, 1994.

immigrants for changing optimization problems. In
Proceedings of the 2007 EvoWorkshops 2007 on
EcoCoMnet, EvoFIN, EvoJASP,EvoINTERACTION,
EvoMUSART, EvoSTOC and EvoTransLog, pages

using evolutionary algorithms - a comparative case
study. In PPSN V: Proceedings of the 5th
International Conference on Parallel Problem Solving from

[38] E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, and
V. da Fonseca. Performance assessment of
multiobjective optimizers: an analysis and review.
IEEE Transactions on Evolutionary Computation,